In-Memory Computing SUMMIT | EUROPE 2018

BIG DATA FOR SMALL DOLLARS.

NEIL STEVENSON

11:55, 25TH JUNE

ABOUT ME – NEIL STEVENSON

- neil@hazelcast.com

- Solution architect for Hazelcast
- Started in IT in 1989
- Has maintained programs written before he was born
- Fond of coffee , beer, and coffee
- Mainly a Java person, some GoLang
- Remembers the launch of C++
- Knows what IEFBR14 is

BIG DATA

- Who remembers the ”Y2K Problem“ ?

- Data records looked like “`SW1V1EQ 1155180625`”.

- POSTCODE, byte[8]
- TIME, byte[4]
- DAY, byte[6]
  - This was BIG data! We could not afford 8 bytes for day

BIG DATA

- BIG DATA == Data we cannot afford to store
- Storage costs money
  - $$$$$
  - £££££
- Storage is cheaper and bigger than Y2K days
  - But data is bigger too, increasing at a faster rate, so the problem isn't going away

BIG DATA

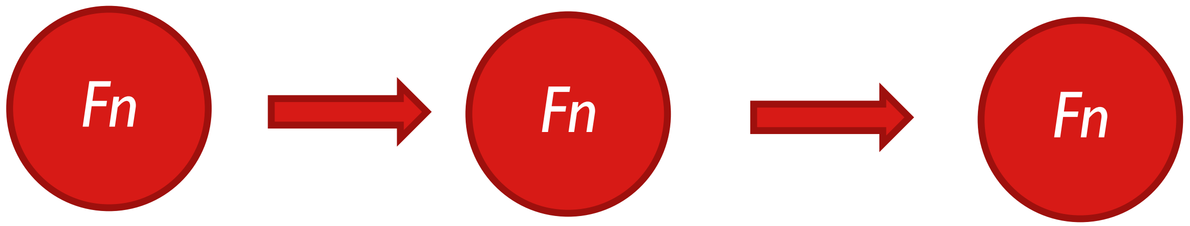
- BIG DATA == Data we cannot afford to store
- Storage costs time
  - Store then compute, results arrive too late, for some applications
  - Even with in-memory storage!
    - So we *need* in-memory computing!

UNIX

- This is a Unix command "`ls | grep neil | wc -l`".
  - "`ls`" == no input, output is list of files
    - Discrete, output is produced then command ends
  - "`grep neil`" == filter for input containing the word *neil,* output the matches
    - Continuous, output produced as input arrives
  - "`wc -l`" == count the input, output the count
    - Discrete, output produced when input exhausted
- It's a simple chain of processing, no intermediate storage

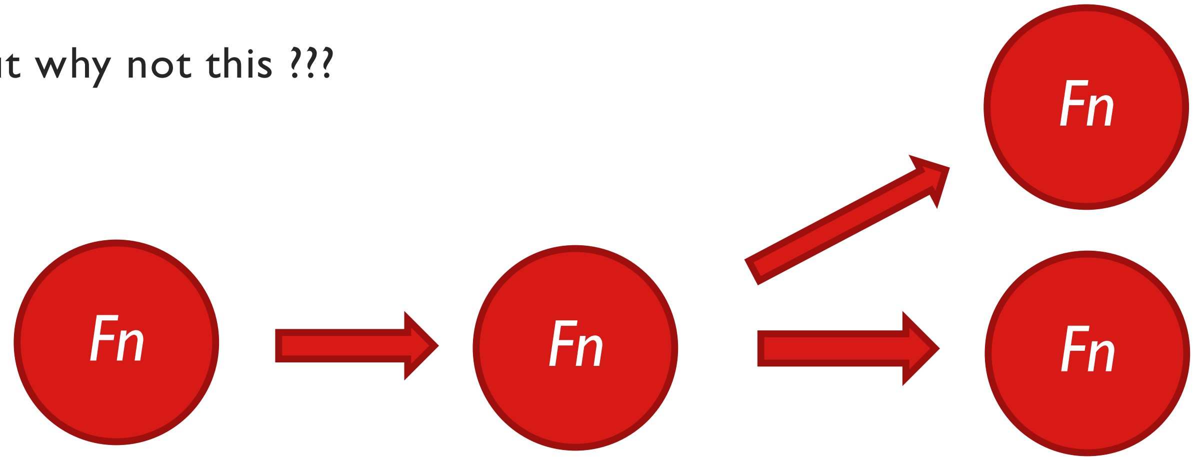
"LS | GREP NEIL | WC -L"

- Really it's this:



"LS | GREP NEIL | WC -L"

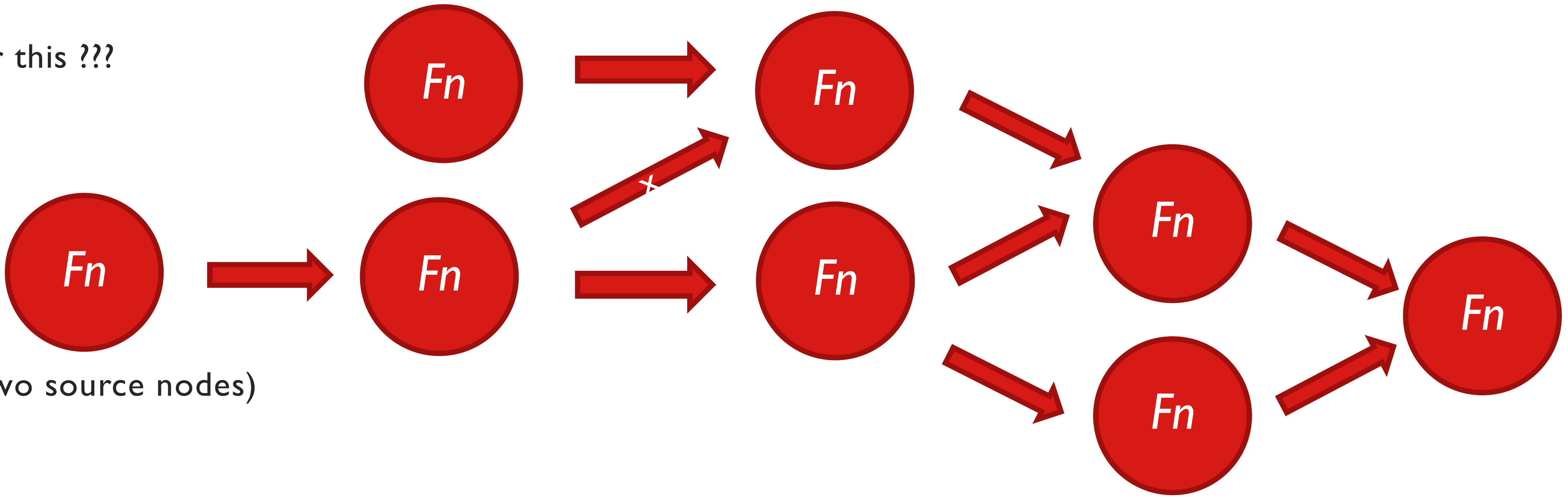
- But why not this ???



The "`tee`" command ??

”LS | GREP NEIL | WC -L”

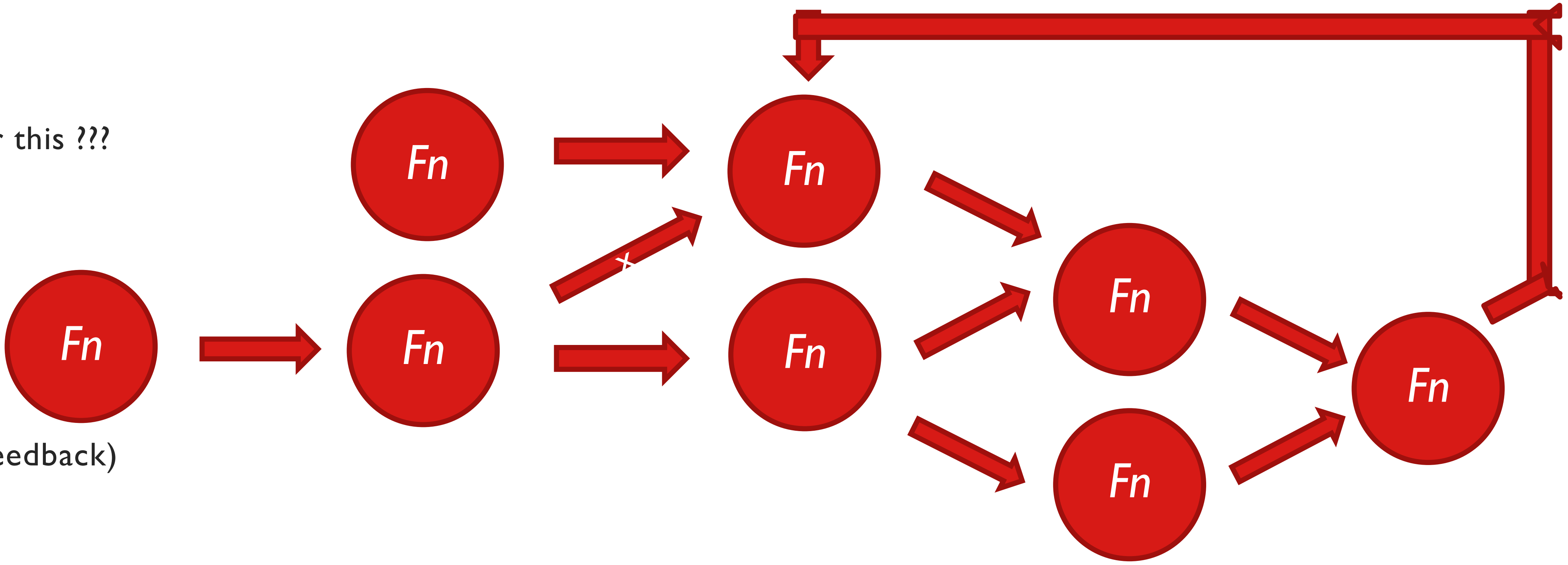
- Or this ???



- (Two source nodes)

" LS | GREP NEIL | WC -L"

- Or this ???



- (Feedback)

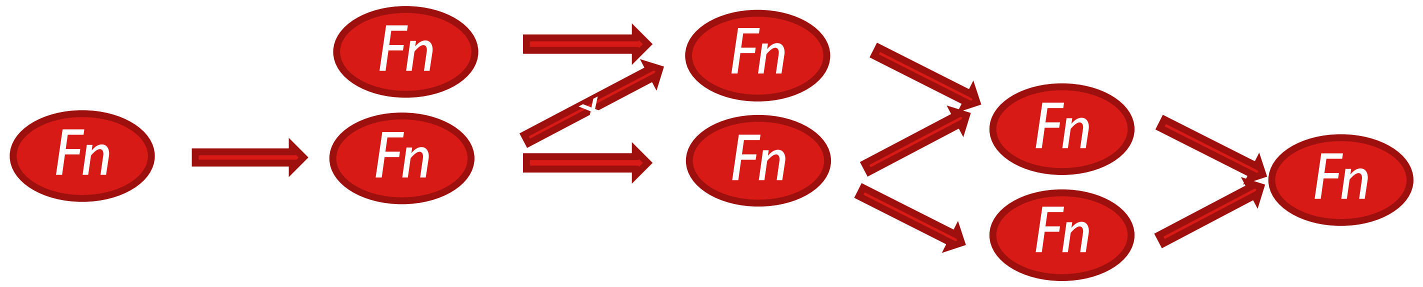
ENTER HAZELCAST JET!

- Java based
- Open source
- Apache 2 licensed
- Distributed Streaming Analytics Engine
- Integrates trivially with Hazelcast IMDG
- Really good, says Neil that works for Hazelcast ☺

ENTER HAZELCAST JET!

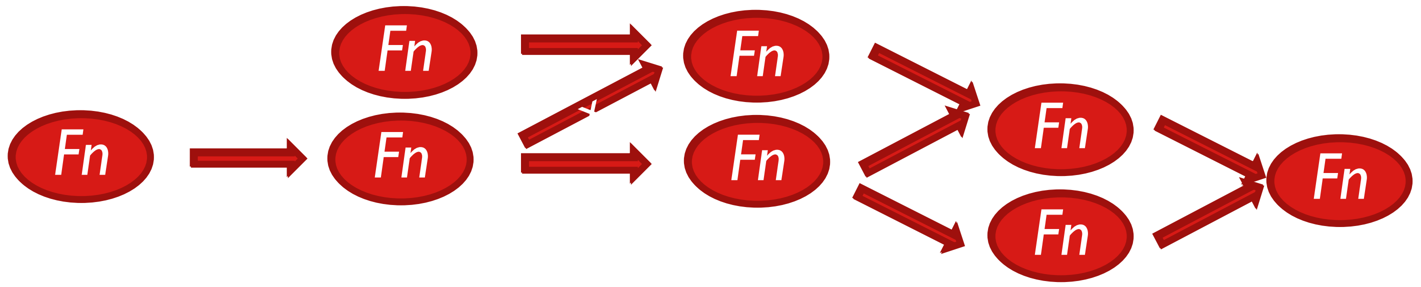
hazelcast JET

- Based around **acyclic graphs**.
  - No feedback loops



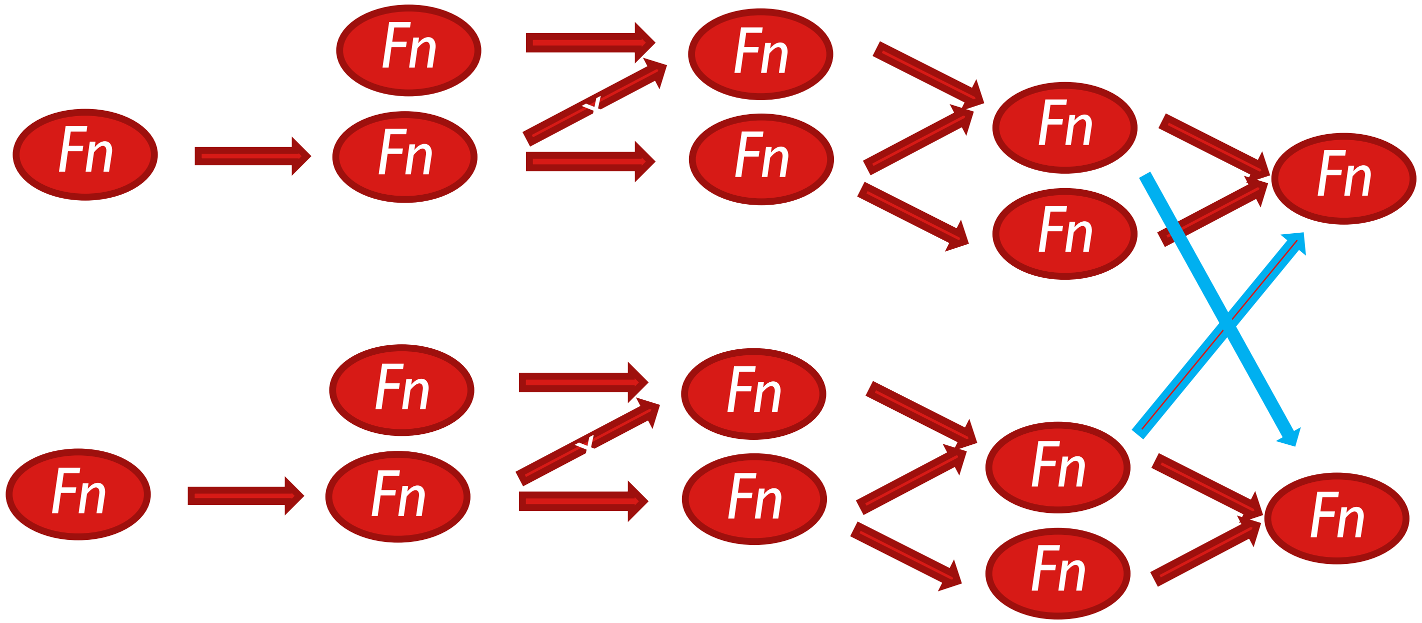
ENTER HAZELCAST JET!

- But **distributed** acyclic graphs.
  - If you have 2 CPUs, run it twice
  - Different JVM or same JVM



ENTER HAZELCAST JET!

- But **distributed** acyclic graphs.
  - If you have 2 CPUs, run it twice
  - Different JVM or same JVM
  - Data can cross instances



THE UBIQUITOUS “WORD COUNT”

```
Pipeline pipeline = Pipeline.create();

pipeline.drawFrom(Sources.<Integer, String>map("hamlet"))
flatMap(entry -> Traversers.traverseArray(Pattern.compile("\\W+").split(entry.getValue())))
.map(String::toLowerCase)
.filter(s -> s.length() > 3)
.groupingKey(DistributedFunctions.wholeItem())
.aggregate(AggregateOperations.counting())
drainTo(Sinks.map("count"));
```

- Quiz time: Can you spot the mistake ?????

THE UBIQUITOUS “WORD COUNT”

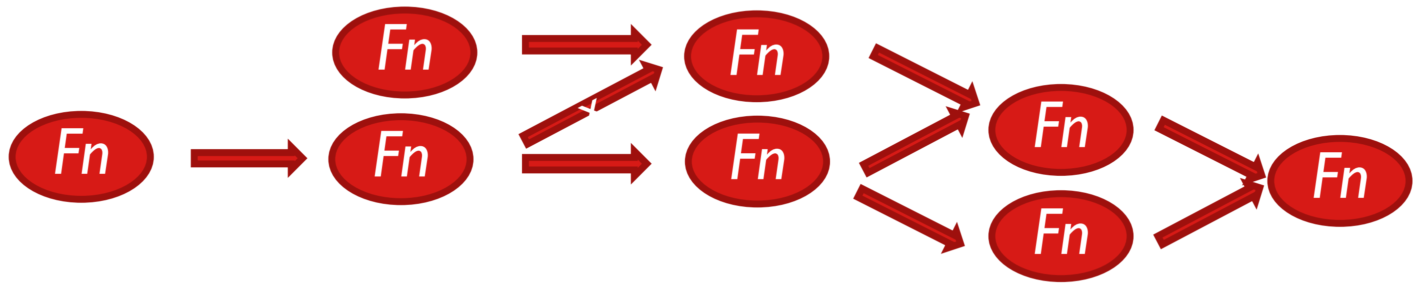
```
Pipeline pipeline = Pipeline.create();


pipeline.drawFrom(Sources.<Integer, String>map("hamlet"))
flatMap(entry -> Traversers.traverseArray(Pattern.compile("\\W+").split(entry.getValue())))
.map(String::toLowerCase)
.filter(s -> s.length() > 3)
.groupingKey(DistributedFunctions.wholeItem())
.aggregate(AggregateOperations.counting())
drainTo(Sinks.map("count"));
```

- Answer: Filter on length is more efficient if it precedes “`toLowerCase()`”. Performance cost!!! Not trivial

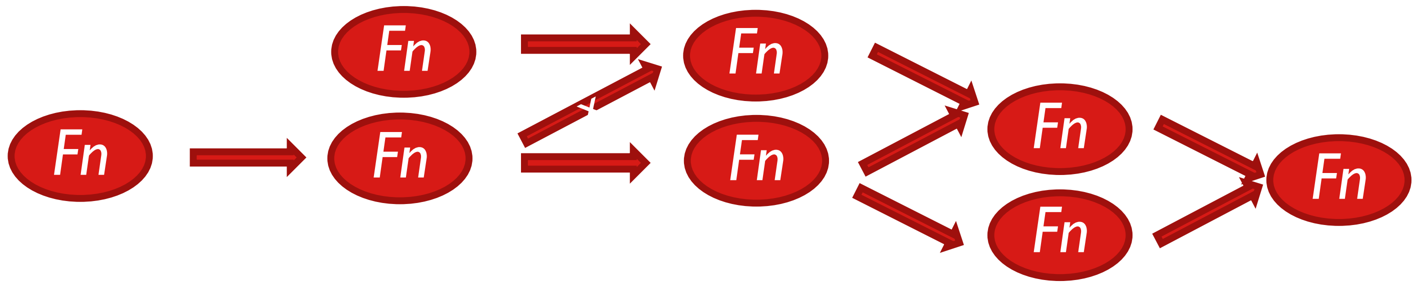
TO BE OR NOT TO BE, THAT IS THE QUESTION

- Data ingest is in parallel

To be

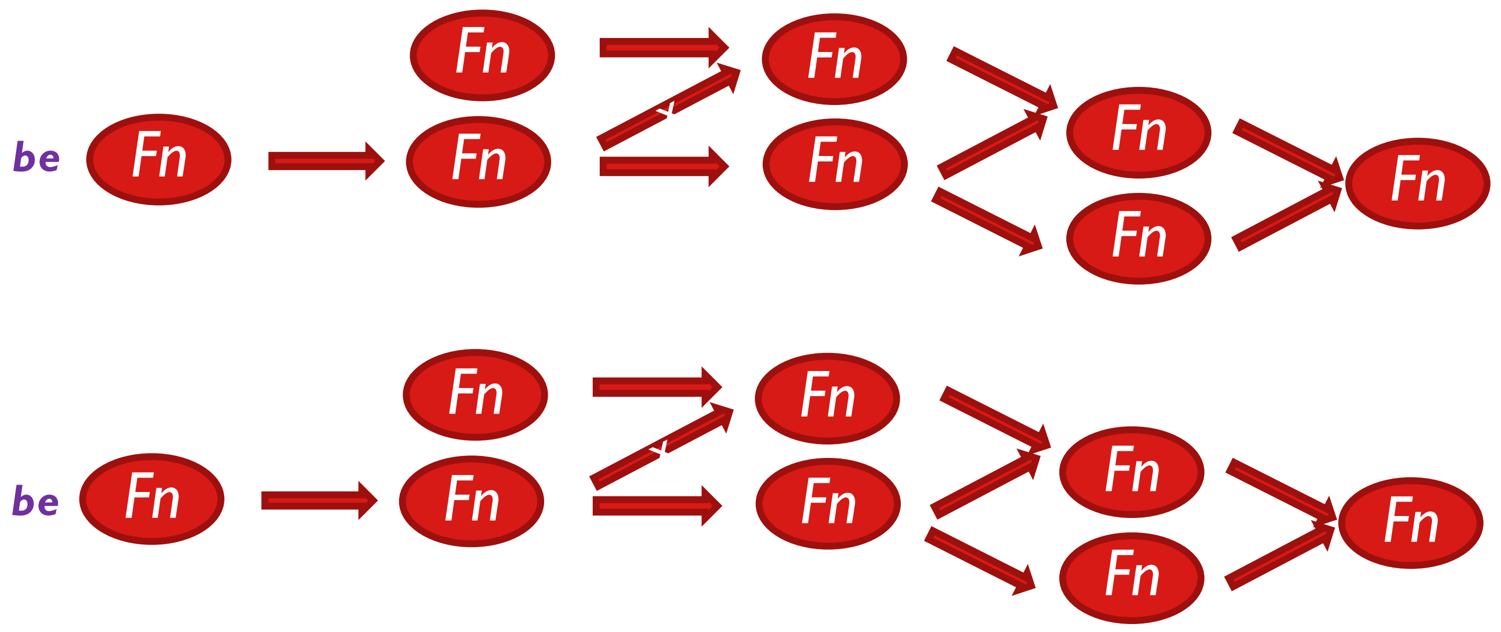


Or not to be



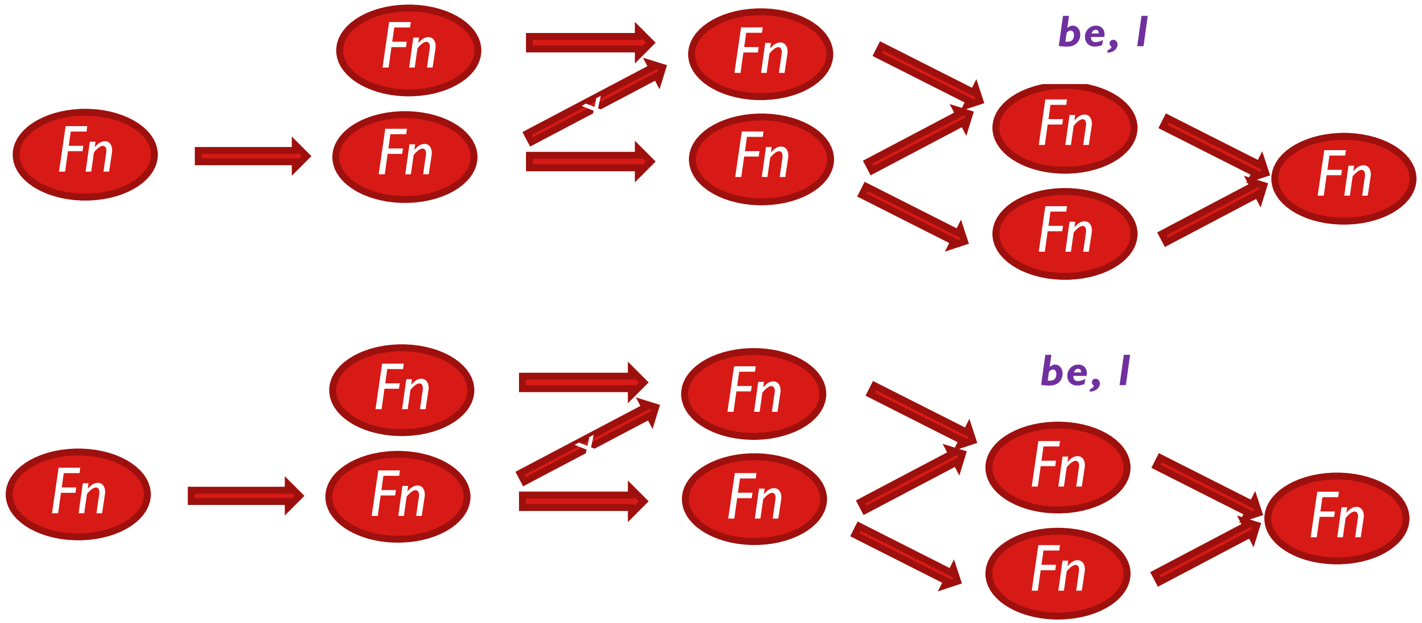
TO BE OR NOT TO BE, THAT IS THE QUESTION

- Data ingest is in parallel



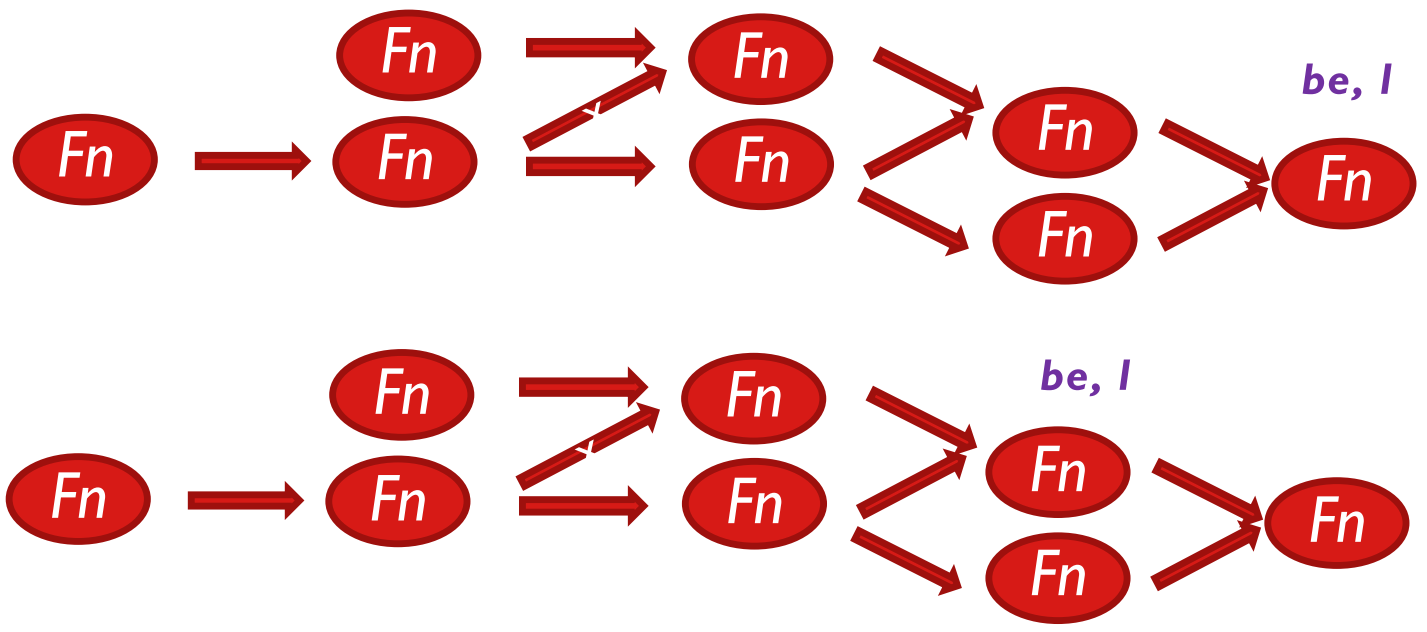
TO BE OR NOT TO BE, THAT IS THE QUESTION

- Data ingest is in parallel
- Data egest is in parallel
- ..if you want



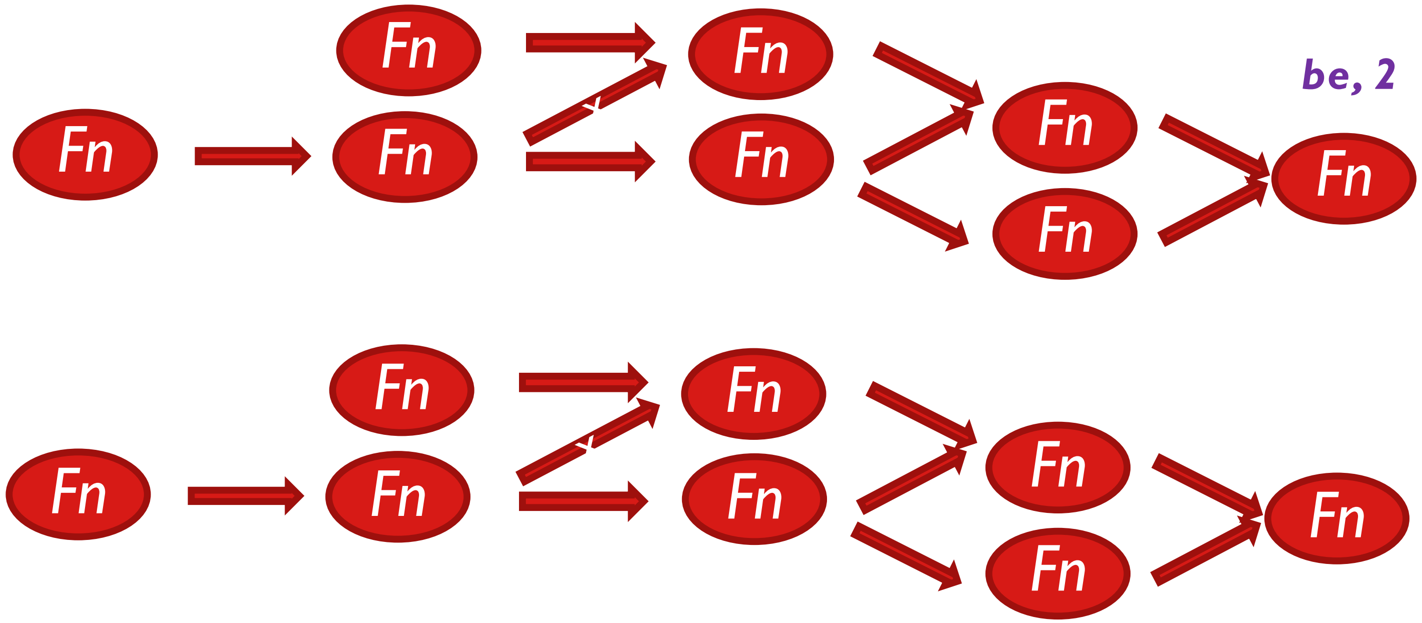
TO BE OR NOT TO BE, THAT IS THE QUESTION

- Data ingest is in parallel
- Data egest is in parallel
- ..if you want



TO BE OR NOT TO BE, THAT IS THE QUESTION

- Data ingest is in parallel
- Data egest is in parallel
- ..if you want



MEANWHILE

- Ok, we have fast streaming processing….
- Next we need some data, BIG data

WHAT IS BIG

- Superbowl 2018
  - Eagles v Patriots, 103.4 million viewers
    - https://www.cbsnews.com/news/super-bowl-lii-tv-ratings/
- Superbowl 2018 Half-Time Show
  - Justin Timberlake, 106.6 million viewers
    - http://money.cnn.com/2018/02/05/media/super-bowl-ratings/index.html
- World Cup 2014
  - Argentina v Germany final, 1.013 billion viewers
    - https://www.fifa.com/worldcup/news/2014-fifa-world-cuptm-reached-3-2-billion-viewers-one-billion-watched--2745519

THE 2014 WORLD CUP FINAL

- The final had 280 MILLION ONLINE viewers

- Many of these have Twitter accounts and will be tweeting
  - 674 million tweets about the final, before, during and after
  - Peak at 618,000 a minute (when Germany scored)

SO….

- Twitter is already storing the tweets, but we'd like to analyse them
- We want to do sentiment analysis
  - Who do the fans think will win before the game starts ?
  - Who do the fans think will win while the game is in progress ?
  - Why do we want to do this ?
    - Place a bet on the winner ! Make SMALL DOLLARS

THE PIPELINE

- Twitter firehose, tweets by hashtag <= could be parallel input across multiple JVMs
- | Filter out if not ASCII
- | Enrich by locating a named team
- | Filter out if no team named
- | Filter out if team named not playing in this game
- | Enrich with sentiment
- | Increment running totals <= possible contention point, unless routing is used

THE PIPELINE

- Twitter firehose, tweets by hashtag
- | Filter out if not ASCII
- | Enrich by locating a named team
- | Filter out if no team named <= Route here on team name
- | Filter out if team named not playing in this game
- | Enrich with sentiment <= Or is here better ?
- | Increment running totals

DEMO TIME

- Let’s see code
  - `java -jar target/worldcup-0.0.1-SNAPSHOT.jar`

- Uruguay v Russia is today at 3pm

DEMO TIME

- Join in!!!
- Uruguay v Russia is today at 3pm
  - ***Hashtag "#URURUS"***

DOES THIS WORK ?

- No
- ….. Or not yet, the business logic is too naïve

- But the idea is sound
- Download the code and fix it yourself ☺

DOES THIS WORK ?

- Some successes!

- Argentina v Croatia, after 18 minutes the sentiment at 0-0 was Argentina to lose. Final score 0-3
- Iran v Spain, at half-time and 0-0 the sentiment was for draw. Final score was 0-1, but Iran had a goal disallowed
- Uruguay v Saudi Arabia, at half-time and 0-0 the sentiment was for Uruguay. Final score was 1-0.

- But most of the others were wrong, so I'm not betting any money on the "predictions"

SUMMARY

- Stream processing == processing before storage
  - Someone else has stored already, eg. an IMDG
  - Can’t afford cost of storage
  - Can’t afford time for storage

- Distributed pipeline is a way to think about processing as a chain of simpler steps
  - Can benefit from machine parallisation

SUMMARY

- neil@hazelcast.com
- https://github.com/neilstevenson/worldcup
  - You will need your own Twitter credentials
- Questions ?