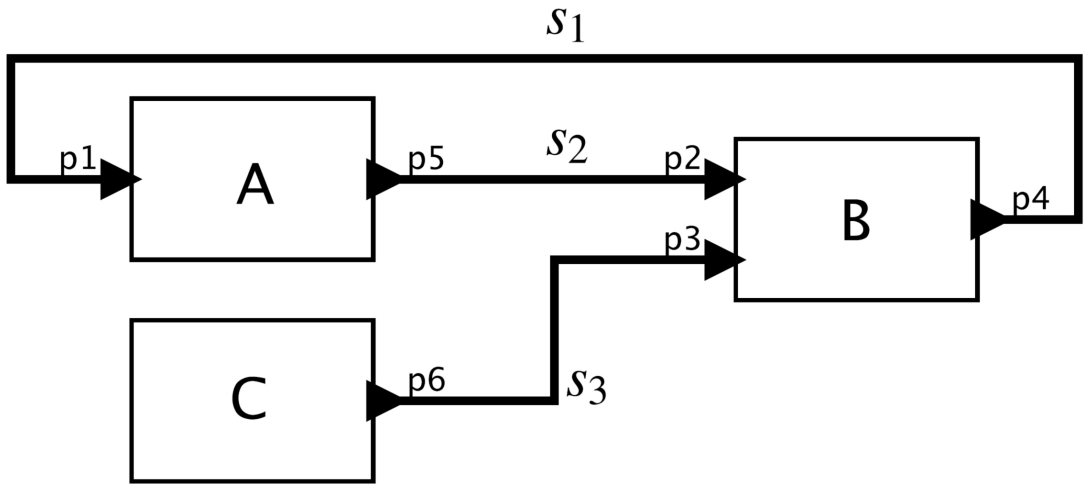
Kahn Process Networks

- A set of computational components, called *actors*.
- Each representing a sequential procedure (process unit).
  - receiving messages from other actors
  - performing local operations
  - sending messages to other actors
- Messages are communicated asynchronously with unbounded buffers.
- A procedure can always send a message. It does not need to wait for the recipient to be ready to receive.
- Messages are delivered reliably and in order.
- When a procedure attempts to receive a message, that attempt blocks the procedure until a message is available.

Example 1



Example of a process network

- Kahn process networks
  - least fixed point semantics, deterministic
  - unbounded buffer
  - grid databases: one request/one response
- No bunbounded buffer

Desired Properties of concurrent, parallel, distributed, event-driven systems

- asynchronism
- determinism
- dead lock free
- scalability

MapReduce

- Model for processing large data sets
- Contains Map and Reduce functions
- Runs on a large cluster of machines
- A lot of MapReduce programs are executed on Google's cluster everyday

Motivation of MapReduce

- Input data is large
  - the whole web, billions of pages
- Lots of machines
  - how to use them efficiently

Programming model

- Input and output
  - each a set of key/value pairs
- Programmer specifies two functions
  - map(in-key, in-value) $\longrightarrow$ list(out-key, intermediate-value)
    - processing input key/value pair
    - process set of intermediate pairs
  - reduce(out-key, list(intermediate-value)) $\longrightarrow$ list(out-value)
    - Combines all intermediate values for a particular key
    - Produces a set of merged output values (usually just one)

Example Word counter

Input: a document with 3 pages

- Page 1: the weather is good
- Page 2: today is good
- Page 3: good weather is good

Map Output

- Worker 1 (one processor)
  - (the 1), (weather 1), (is 1), (good 1)
- Worker 2
  - (today 1), (is 1), (good 1)
- Worker 3
  - (good 1), (weather 1), (is 1), (good 1)

Reduce Input

- Worker 1
  - (the 1),
- Worker 2
  - (is 1), (is 1), (is 1)
- Worker 3
  - (weather 1), (weather 1)
- Worker 4
  - (today 1)
- Worker 5
  - (good 1), (good 1), (good 1), (good 1)

Reduce Output

- Worker 1
  - (the 1),
- Worker 2
  - (is 3)
- Worker 3
  - (weather 2)
- Worker 4
  - (today 1)
- Worker 5
  - (good 4),

Map Input

- Page 1

  the weather is good

- Page 2

  today is good

- Page 3

  good weather is good

Map Output

- Worker 1 (one processor)

  – (the 1), (weather 1), (is 1), (good 1)

- Worker 2

  – (today 1), (is 1), (good 1)

- Worker 3

  – (good 1), (weather 1), (is 1), (good 1)

Reduce Input

- Worker 1

  – (the 1),

- Worker 2

  – (is 1), (is 1), (is 1)

- Worker 3

  – (weather 1), (weather 1)

- Worker 4

  – (today 1)

- Worker 5

  – (good 1), (good 1), (good 1), (good 1)

Reduce Output

- Worker 1

  – (the 1),

- Worker 2

  – (is 3)

- Worker 3

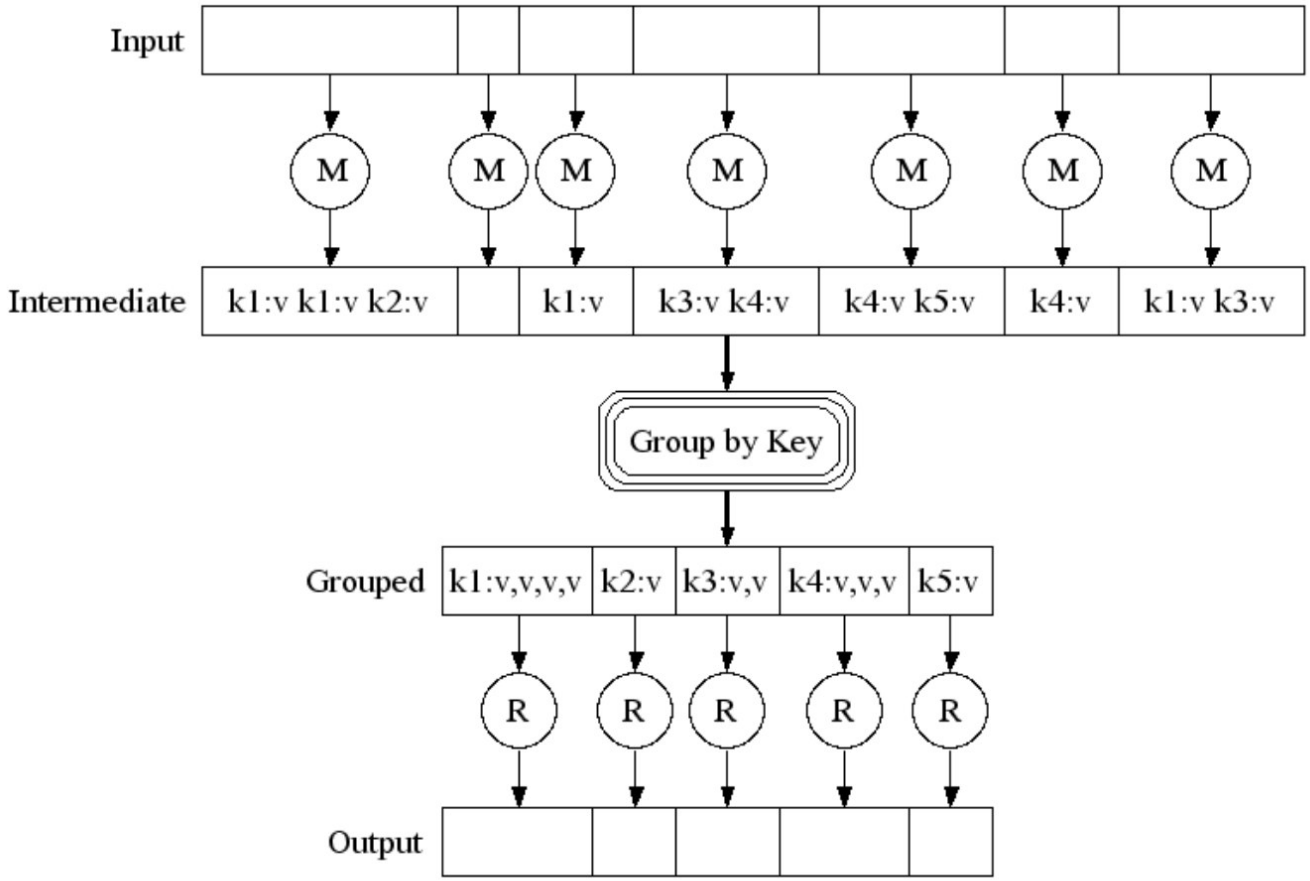
  – (weather 2)

- Worker 4

  – (today 1)

- Worker 5

  – (good 4),


Input
M
M
M
M
M
M
M
Intermediate
k1:v k1:v k2:v
k1:v
k3:v k4:v
k4:v k5:v
k4:v
k1:v k3:v
Group by Key
Grouped
k1:v,v,v,v
k2:v
k3:v,v
k4:v,v,v
k5:v
R
R
R
R
R
Output

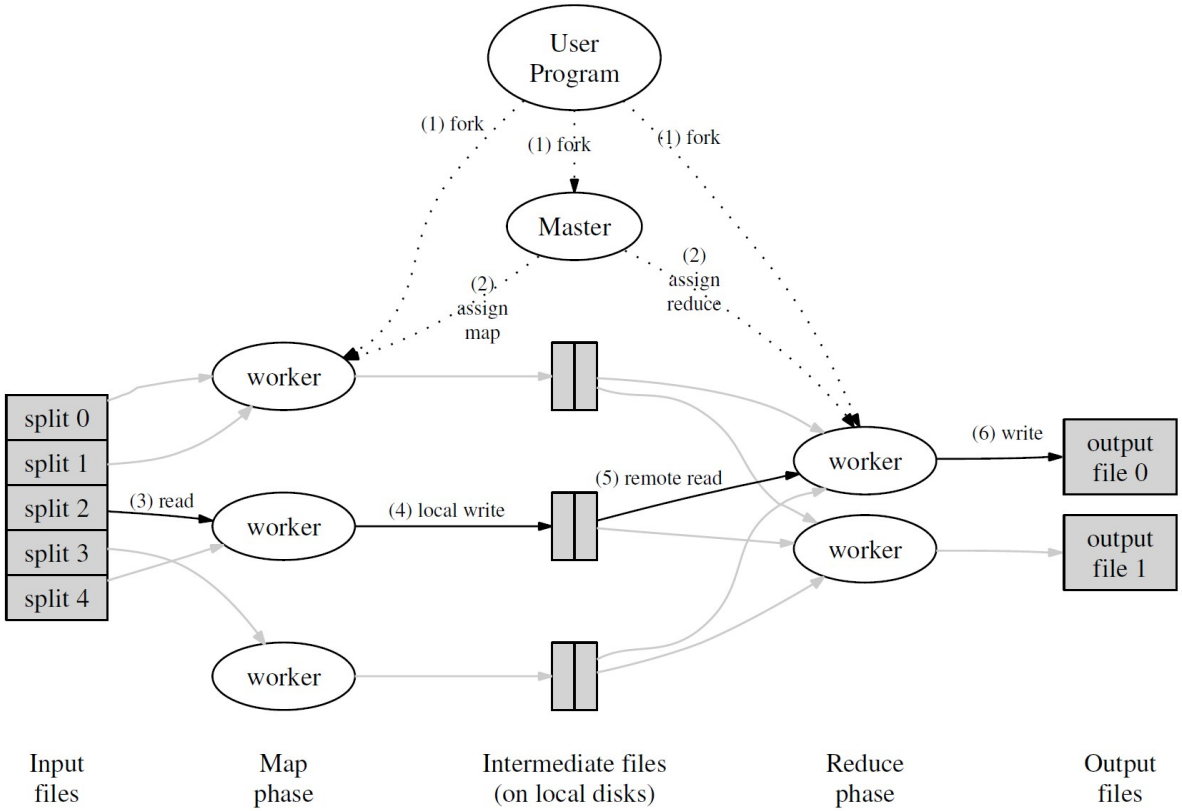
Other Examples

- Inverted Index
  - Input: list of <docId, document>
  - Output: list of <Word, list(docId, frequency)>
  - Map: parse each document and outputs a sequence of <word, docId>
  - Reduce: accept all pairs for a given work, sorts docId according its frequency, and generate the output
- Count of URL access frequency
  - Map function

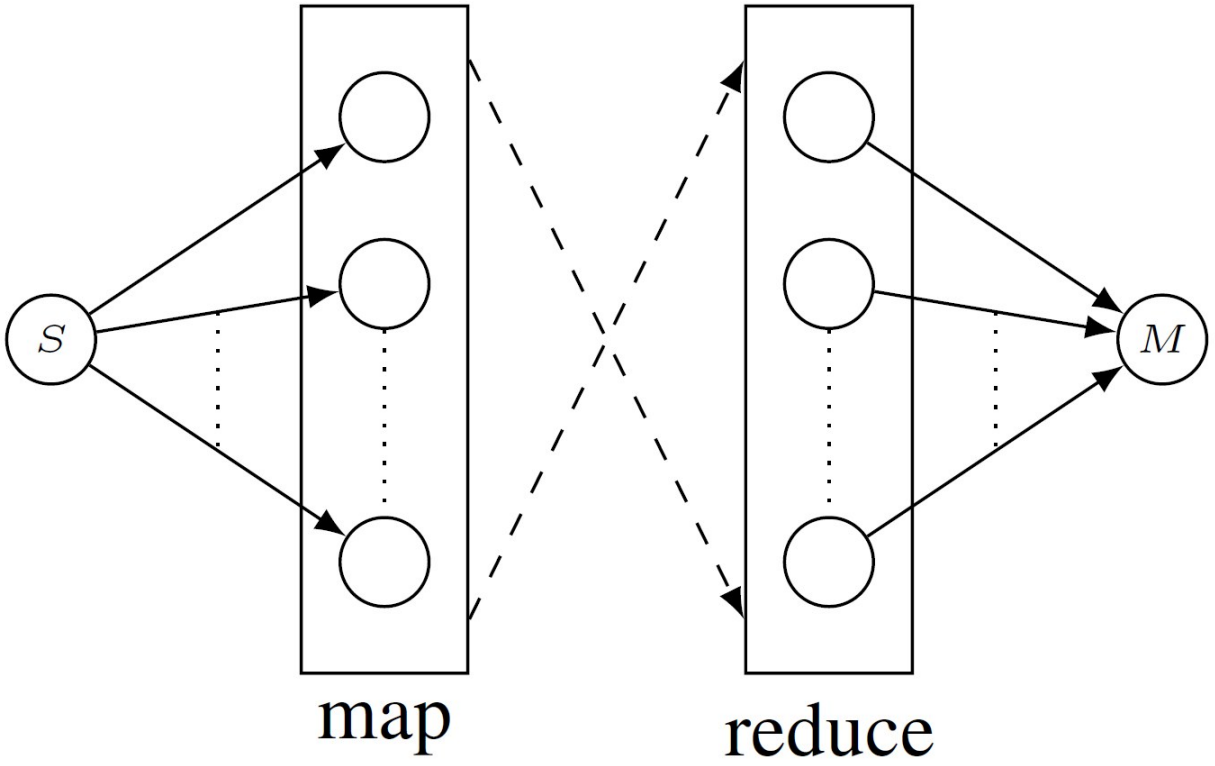
    process logs of web page requests and outputs <URL, 1>
  - Reduce function

    add all values for the same URL and outputs a <URL, total count>

Execution Overview

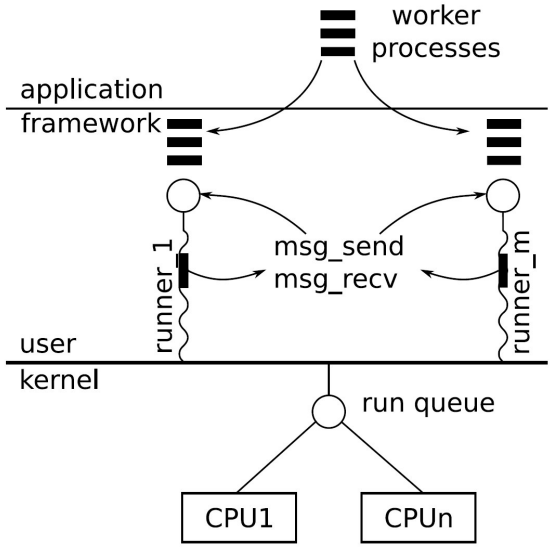


MapReduce vs Kahn Process Network



Implementation of Kahn process networks

- KPN scheduling



- Message transport
- Deadlock detection and resolution

Experimental

Word count Count the number of occurrences of each word in a given text document and outputs them sorted in the order of decreasing frequency.
Experimental

- File size: 10, 50, and 100 (MBs)
- Machine: a single server with 8 CPUs

Three solutions

- Phoenix

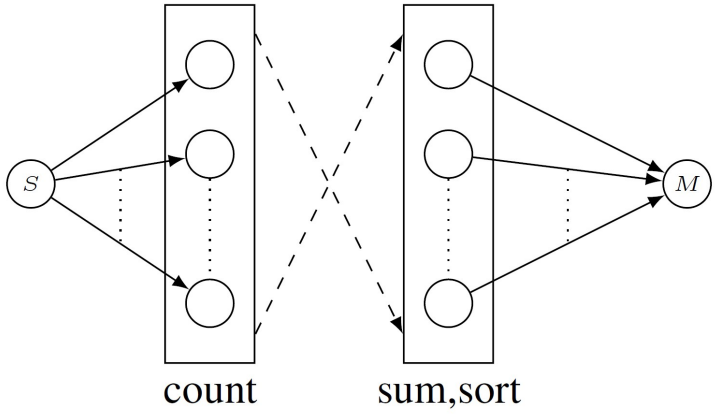
  A MapReduce implementation designed specifically for multi-core machines.

- KPN-MR

  A Kahn process network configured for MapReduce using two instantiations connected in series

$$S \longrightarrow MR_1 \longrightarrow MR_2 \longrightarrow M$$

- KPN-FLEX



Results

Doc Size (MB)	**Phoenix** (s)	**KPN-MR** (s)	**KPN-FLEX** (s)
10	0.30	0.32	0.11
50	0.98	1.86	0.37
100	2.04	4.23	0.74

Experimental for K-means

Points/Group	**Phoenix** (s)	**KPN-MR** (s)	**KPN-FLEX** (s)
100K/100	2.39	1.83	1.51
200K/50	3.92	3.20	2.24
100K/100	5.43	5.19	4/26