

Distributed Systems Seminar

On

“MapReduce: Simplified Data
Processing on Large
Clusters”^[1]

Christoph Pinkel

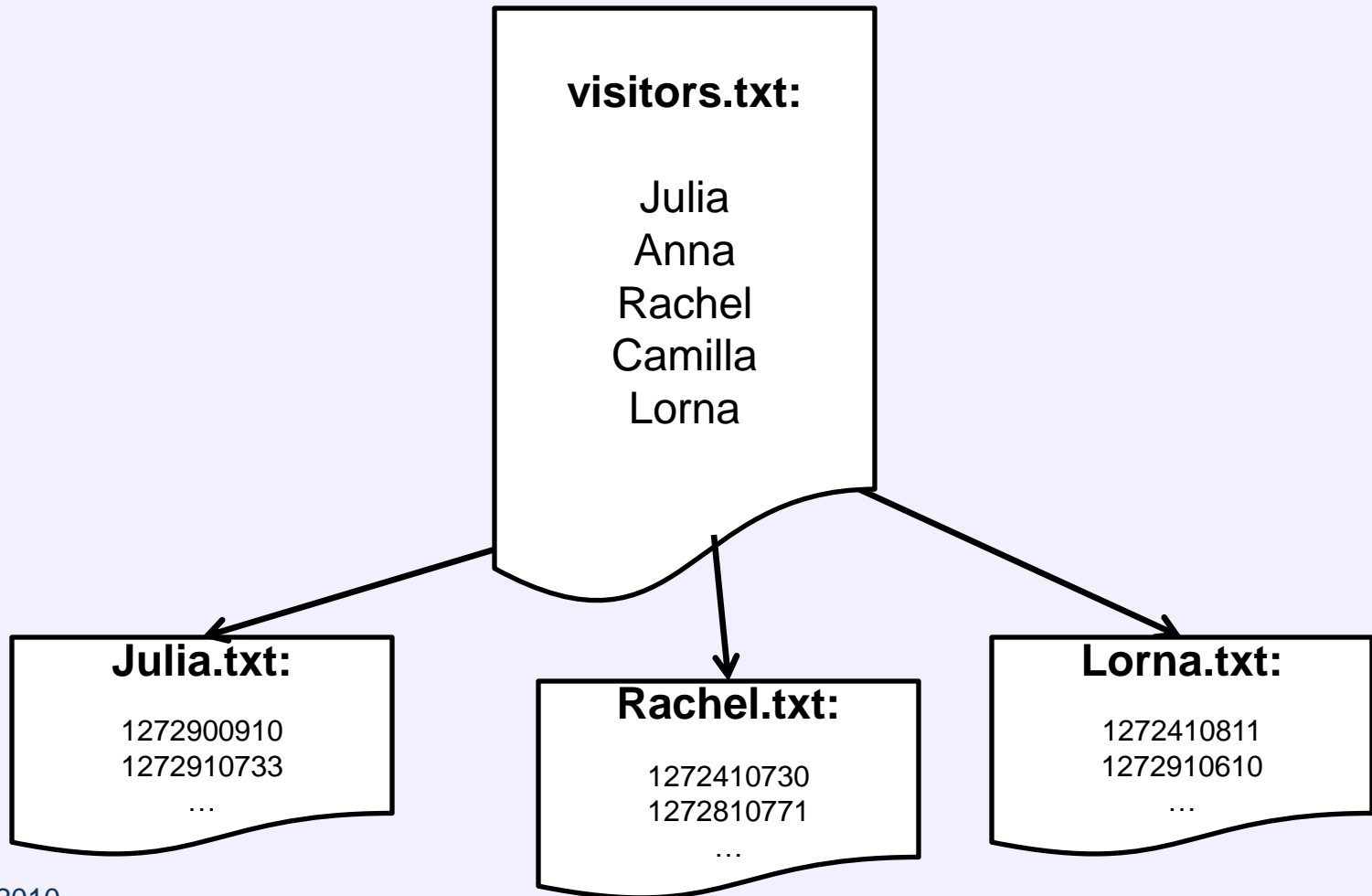
[1] Jeffrey Dean and Sanjay Ghemawat: “MapReduce: Simplified Data Processing on Large Clusters” in *OSDI 2004*

Data Processing

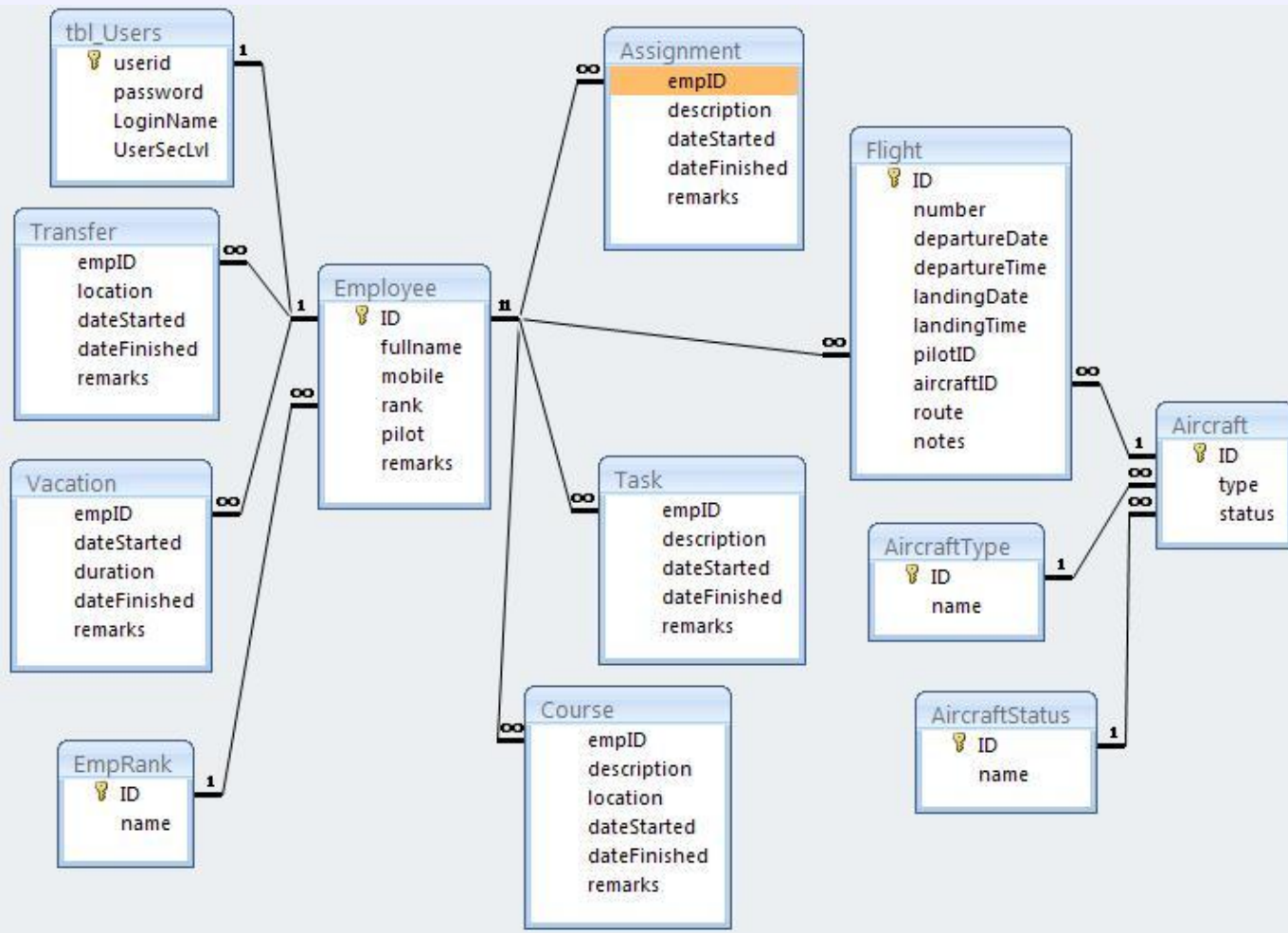
visitors.txt:

Julia
Anna
Rachel
Camilla
Lorna

Data Processing



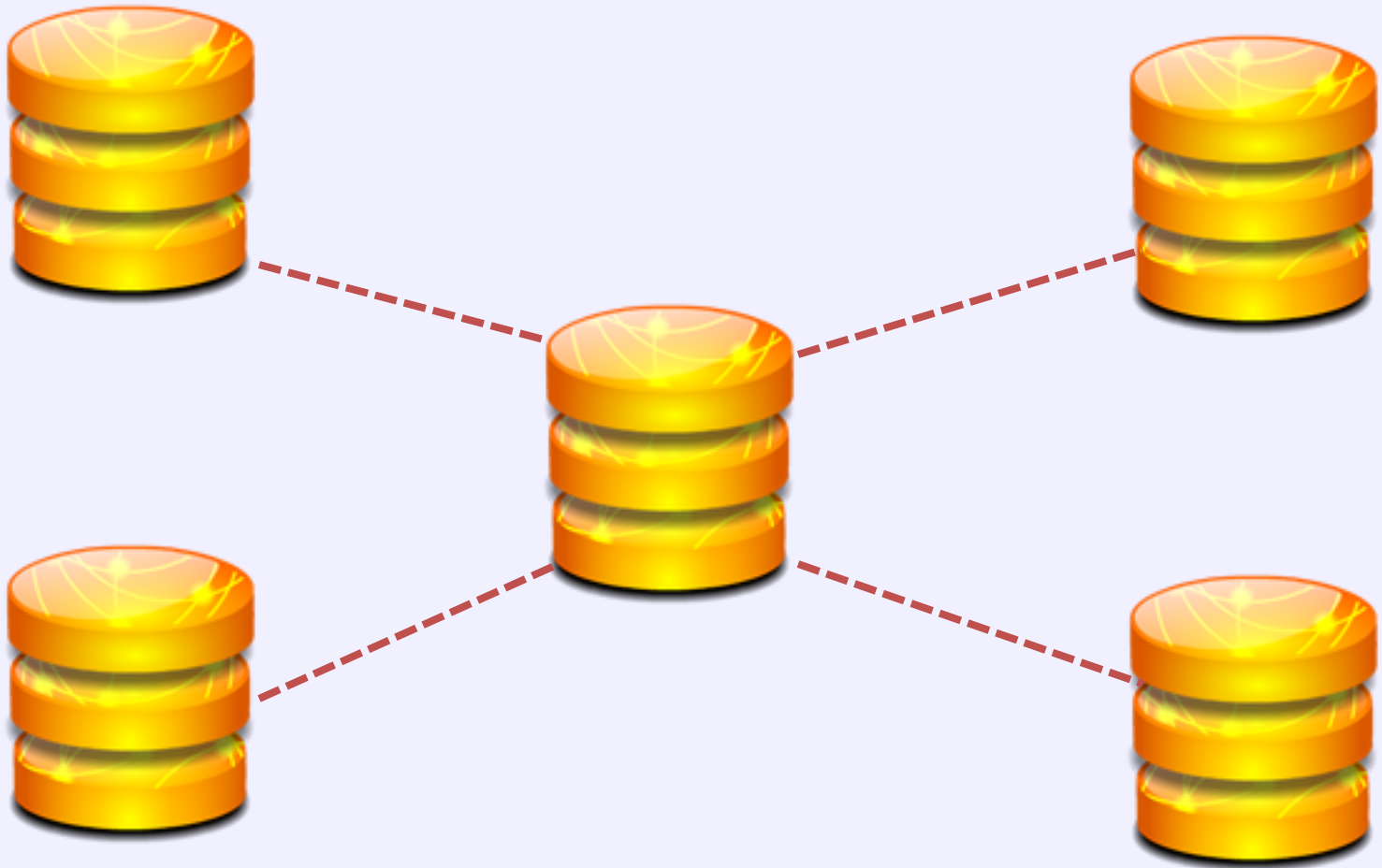
Data Processing



Data Processing



Data Processing



Large-scale Data Processing

- Very large data sets
 - Often not in DBMS
 - Distributed file system
 - Many disks/nodes
 - Several sources
 - Heterogeneous
 - Hard to process

Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process

Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process

Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process

Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process

Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process

Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process

Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process

```
228.228.245.152 -- [05/May/2010:20:52:47 +0200] - GET /url3
228.228.245.152 -- [05/May/2010:20:52:37 +0200] - GET /url2
228.228.245.152 -- [05/May/2010:20:52:51 +0200] - GET /url5
179.20.32.1 -- [05/May/2010:20:52:58 +0200] - GET /url0
179.20.32.1 -- [05/May/2010:20:52:6 +0200] - GET /url5
179.20.32.1 -- [05/May/2010:20:52:7 +0200] - GET /url2
179.20.32.1 -- [05/May/2010:20:52:57 +0200] - GET /url0
179.20.32.1 -- [05/May/2010:20:52:19 +0200] - GET /url2
179.20.32.1 -- [05/May/2010:20:52:58 +0200] - GET /url2
216.63.71.128 -- [05/May/2010:20:52:28 +0200] - GET /url6
216.63.71.128 -- [05/May/2010:20:52:0 +0200] - GET /url3
216.63.71.128 -- [05/May/2010:20:52:47 +0200] - GET /url1
216.63.71.128 -- [05/May/2010:20:52:53 +0200] - GET /url4
216.63.71.128 -- [05/May/2010:20:52:46 +0200] - GET /url1
216.63.71.128 -- [05/May/2010:20:52:16 +0200] - GET /url0
253.224.155.151 -- [05/May/2010:20:52:51 +0200] - GET /url6
253.224.155.151 -- [05/May/2010:20:52:11 +0200] - GET /url1
253.224.155.151 -- [05/May/2010:20:52:54 +0200] - GET /url2
253.224.155.151 -- [05/May/2010:20:52:7 +0200] - GET /url1
253.224.155.151 -- [05/May/2010:20:52:58 +0200] - GET /url1
253.224.155.151 -- [05/May/2010:20:52:59 +0200] - GET /url7
241.38.63.13 -- [05/May/2010:20:52:22 +0200] - GET /url5
241.38.63.13 -- [05/May/2010:20:52:6 +0200] - GET /url0
241.38.63.13 -- [05/May/2010:20:52:16 +0200] - GET /url2
241.38.63.13 -- [05/May/2010:20:52:57 +0200] - GET /url2
241.38.63.13 -- [05/May/2010:20:52:10 +0200] - GET /url4
241.38.63.13 -- [05/May/2010:20:52:8 +0200] - GET /url3
179.250.120.23 -- [05/May/2010:20:52:57 +0200] - GET /url6
179.250.120.23 -- [05/May/2010:20:52:40 +0200] - GET /url6
179.250.120.23 -- [05/May/2010:20:52:22 +0200] - GET /url7
179.250.120.23 -- [05/May/2010:20:52:36 +0200] - GET /url2
179.250.120.23 -- [05/May/2010:20:52:10 +0200] - GET /url2
179.250.120.23 -- [05/May/2010:20:52:58 +0200] - GET /url2
8.129.236.60 -- [05/May/2010:20:52:13 +0200] - GET /url2
8.129.236.60 -- [05/May/2010:20:52:58 +0200] - GET /url2
8.129.236.60 -- [05/May/2010:20:52:58 +0200] - GET /url2
8.129.236.60 -- [05/May/2010:20:52:58 +0200] - GET /url2
```

Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process

```
228.228.245.152 -- [05/May/2010:20:52:47 +0200] - GET /url3
228.228.245.152 -- [05/May/2010:20:52:37 +0200] - GET /url2
228.228.245.152 -- [05/May/2010:20:52:51 +0200] - GET /url5
179.20.32.1 -- [05/May/2010:20:52:58 +0200] - GET /url0
179.20.32.1 -- [05/May/2010:20:52:6 +0200] - GET /url5
179.20.32.1 -- [05/May/2010:20:52:7 +0200] - GET /url2
179.20.32.1 -- [05/May/2010:20:52:57 +0200] - GET /url0
179.20.32.1 -- [05/May/2010:20:52:19 +0200] - GET /url2
179.20.32.1 -- [05/May/2010:20:52:58 +0200] - GET /url2
216.63.71.128 -- [05/May/2010:20:52:28 +0200] - GET /url6
216.63.71.128 -- [05/May/2010:20:52:0 +0200] - GET /url3
216.63.71.128 -- [05/May/2010:20:52:47 +0200] - GET /url1
216.63.71.128 -- [05/May/2010:20:52:53 +0200] - GET /url4
216.63.71.128 -- [05/May/2010:20:52:16 +0200] - GET /url1
216.63.71.128 -- [05/May/2010:20:52:16 +0200] - GET /url0
253.224.155.151 -- [05/May/2010:20:52:51 +0200] - GET /url0
253.224.155.151 -- [05/May/2010:20:52:51 +0200] - GET /url1
253.224.155.151 -- [05/May/2010:20:52:4 +0200] - GET /url2
253.224.155.151 -- [05/May/2010:20:52:4 +0200] - GET /url1
253.224.155.151 -- [05/May/2010:20:52:58 +0200] - GET /url1
253.224.155.151 -- [05/May/2010:20:52:59 +0200] - GET /url7
241.38.63.13 -- [05/May/2010:20:52:22 +0200] - GET /url5
241.38.63.13 -- [05/May/2010:20:52:6 +0200] - GET /url0
241.38.63.13 -- [05/May/2010:20:52:16 +0200] - GET /url2
241.38.63.13 -- [05/May/2010:20:52:57 +0200] - GET /url2
241.38.63.13 -- [05/May/2010:20:52:10 +0200] - GET /url4
241.38.63.13 -- [05/May/2010:20:52:8 +0200] - GET /url3
179.250.120.23 -- [05/May/2010:20:52:57 +0200] - GET /url6
179.250.120.23 -- [05/May/2010:20:52:40 +0200] - GET /url6
179.250.120.23 -- [05/May/2010:20:52:22 +0200] - GET /url7
179.250.120.23 -- [05/May/2010:20:52:36 +0200] - GET /url2
179.250.120.23 -- [05/May/2010:20:52:10 +0200] - GET /url2
179.250.120.23 -- [05/May/2010:20:52:58 +0200] - GET /url2
8.129.236.60 -- [05/May/2010:20:52:13 +0200] - GET /url2
8.129.236.60 -- [05/May/2010:20:52:58 +0200] - GET /url2
8.129.236.60 -- [05/May/2010:20:52:58 +0200] - GET /url2
8.129.236.60 -- [05/May/2010:20:52:58 +0200] - GET /url2
```

2TB

Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process



Large-scale Data Processing

- Very large data sets
- Often not in DBMS
- Distributed file system
- Many disks/nodes
- Several sources
- Heterogeneous
- Hard to process



SELECT url, COUNT(visits) FROM log

One Size Fits...?

- Traditional DBMS mantra:
“One Size Fits All”
- DBMS won't do (not even PDBMS)
- Need custom solutions
- Often based on FS type layer

One Size Fits...?

- Traditional DBMS mantra:
“One Size Fits All”
- DBMS won't do (not even PDBMS)
- Need custom solutions
- Often based on FS type layer

One Size Fits...?

- Traditional DBMS mantra:
“One Size Fits All”
- DBMS won't do (not even PDBMS)
 - Need custom solutions
 - Often based on FS type layer

Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
 - Distribution of data
 - Data parallelism
 - Fault tolerance
 - ...

Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
 - Distribution of data
 - Data parallelism
 - Fault tolerance
 - ...

Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
 - Distribution of data
 - Data parallelism
 - Fault tolerance
 - ...

Custom Systems

- Use distributed storage layer
 - Build “custom query”
 - Implement data processing
 - Take care of...
 - Distribution of data
 - Data parallelism
 - Fault tolerance
 - ...
- 2 TB Weblog
 - `SELECT url, COUNT(visits)`
 - Split in parts
 - Hash partition on URL
 - Distribute parts
 - On each: sort by URL
 - Count visits
 - Output partial results

Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
 - Distribution of data
 - Data parallelism
 - Fault tolerance
 - ...
- 2 TB Weblog
- SELECT url, COUNT(visits)
- Split in parts
- Hash partition on URL
- Distribute parts
- On each: sort by URL
- Count visits
- Output partial results

Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
 - Distribution of data
 - Data parallelism
 - Fault tolerance
 - ...
- 2 TB Weblog
- SELECT url, COUNT(visits)
- Split in parts
- Hash partition on URL
- Distribute parts
- On each: sort by URL
- Count visits
- Output partial results

Custom Systems

- Use distributed storage layer
 - Build “custom query”
 - Implement data processing
 - Take care of...
 - Distribution of data
 - Data parallelism
 - Fault tolerance
 - ...
- 2 TB Weblog
 - SELECT url, COUNT(visits)
 - Split in parts
 - Hash partition on URL
 - Distribute parts
 - On each: sort by URL
 - Count visits
 - Output partial results

Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
 - Distribution of data
 - Data parallelism
 - Fault tolerance
 - ...
- 2 TB Weblog
- SELECT url, COUNT(visits)
- Split in parts
- Hash partition on URL
- Distribute parts
- On each: sort by URL
- Count visits
- Output partial results

Custom Systems

- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
 - Distribution of data
 - Data parallelism
 - Fault tolerance
 - ...
- 2 TB Weblog
- SELECT url, COUNT(visits)
- Split in parts
- Hash partition on URL
- Distribute parts
- On each: sort by URL
- Count visits
- Output partial results

Custom Systems

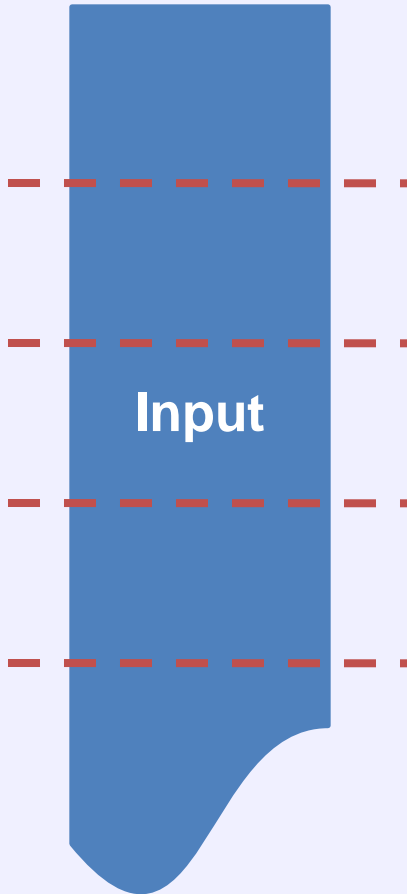
- Use distributed storage layer
- Build “custom query”
- Implement data processing
- Take care of...
 - Distribution of data
 - Data parallelism
 - Fault tolerance
 - ...
- 2 TB Weblog
- `SELECT url, COUNT(visits)`
- Split in parts
- Hash partition on URL
- Distribute parts
- On each: sort by URL
- Count visits
- Output partial results

A Programmer's Nightmare

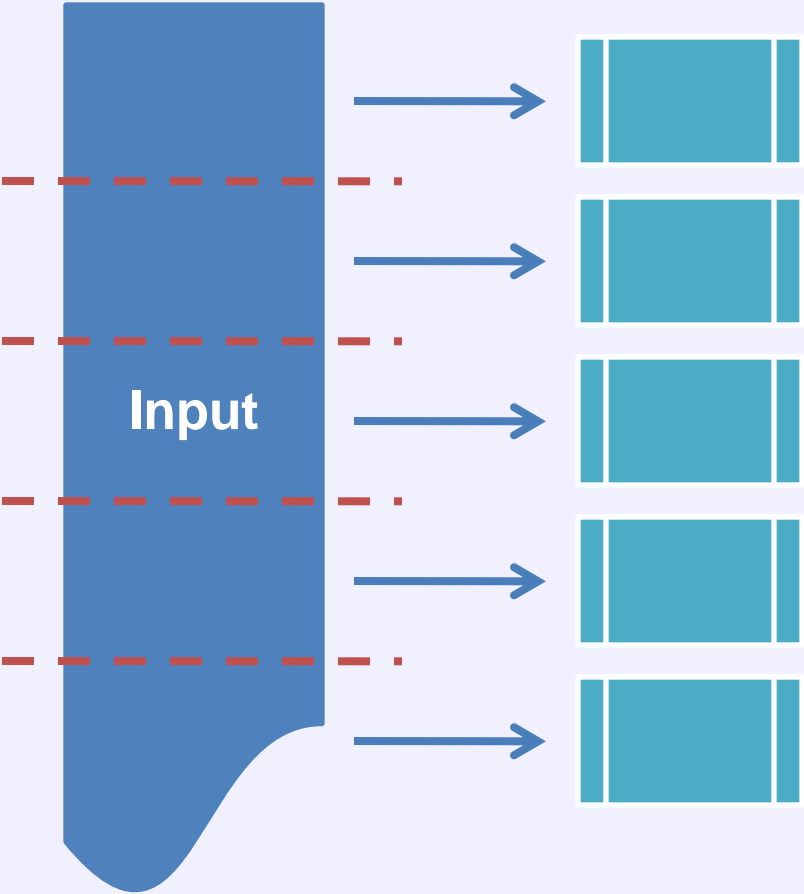


Input

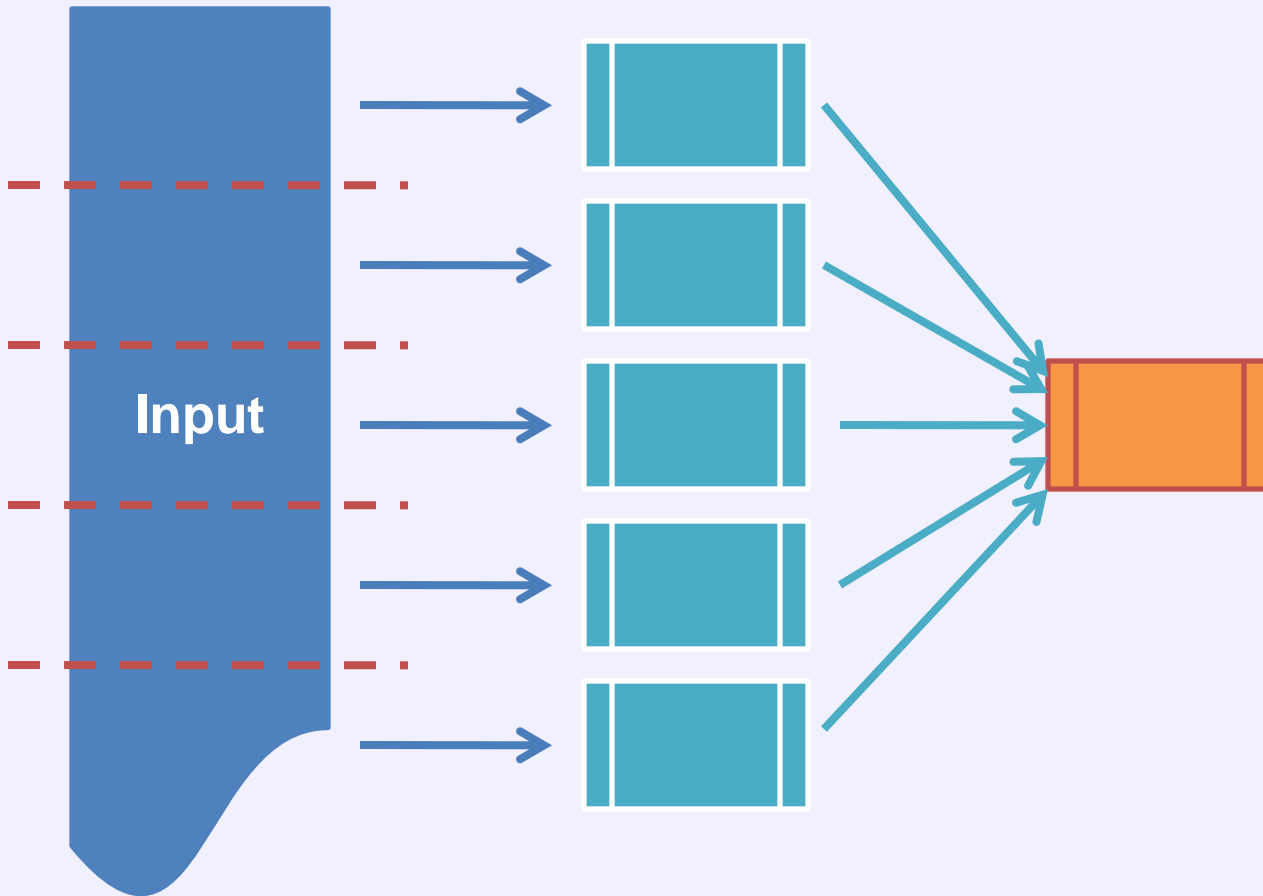
A Programmer's Nightmare



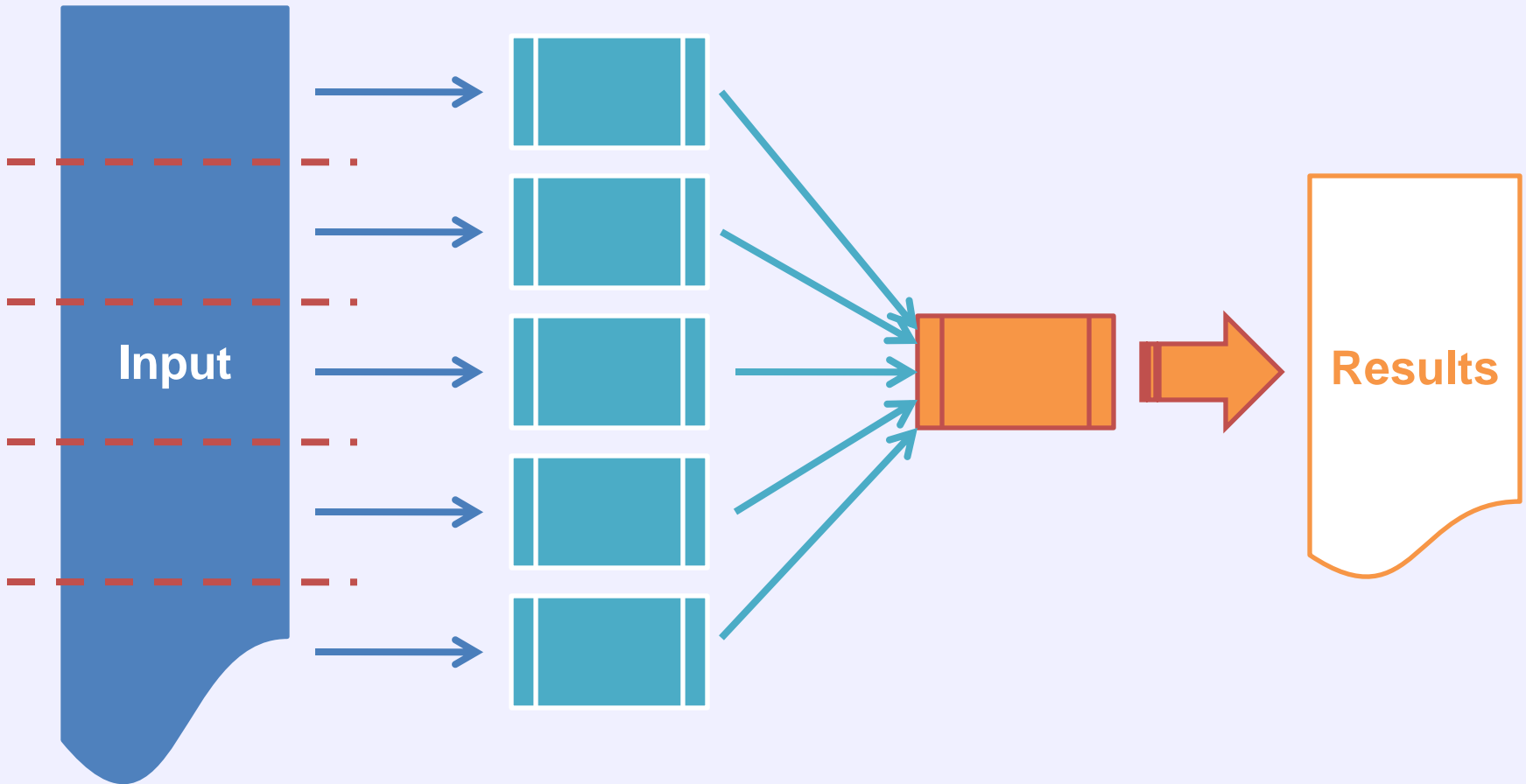
A Programmer's Nightmare



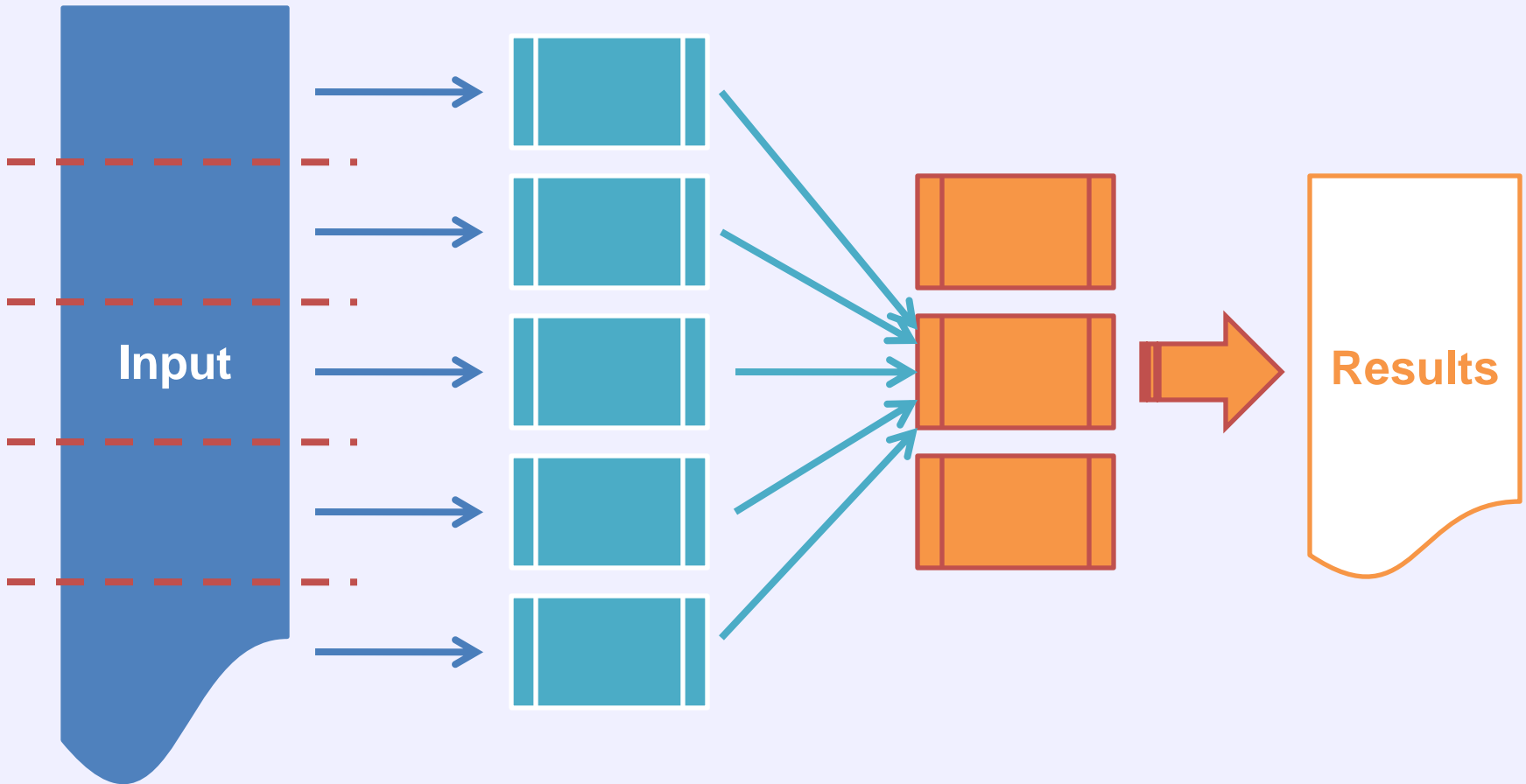
A Programmer's Nightmare



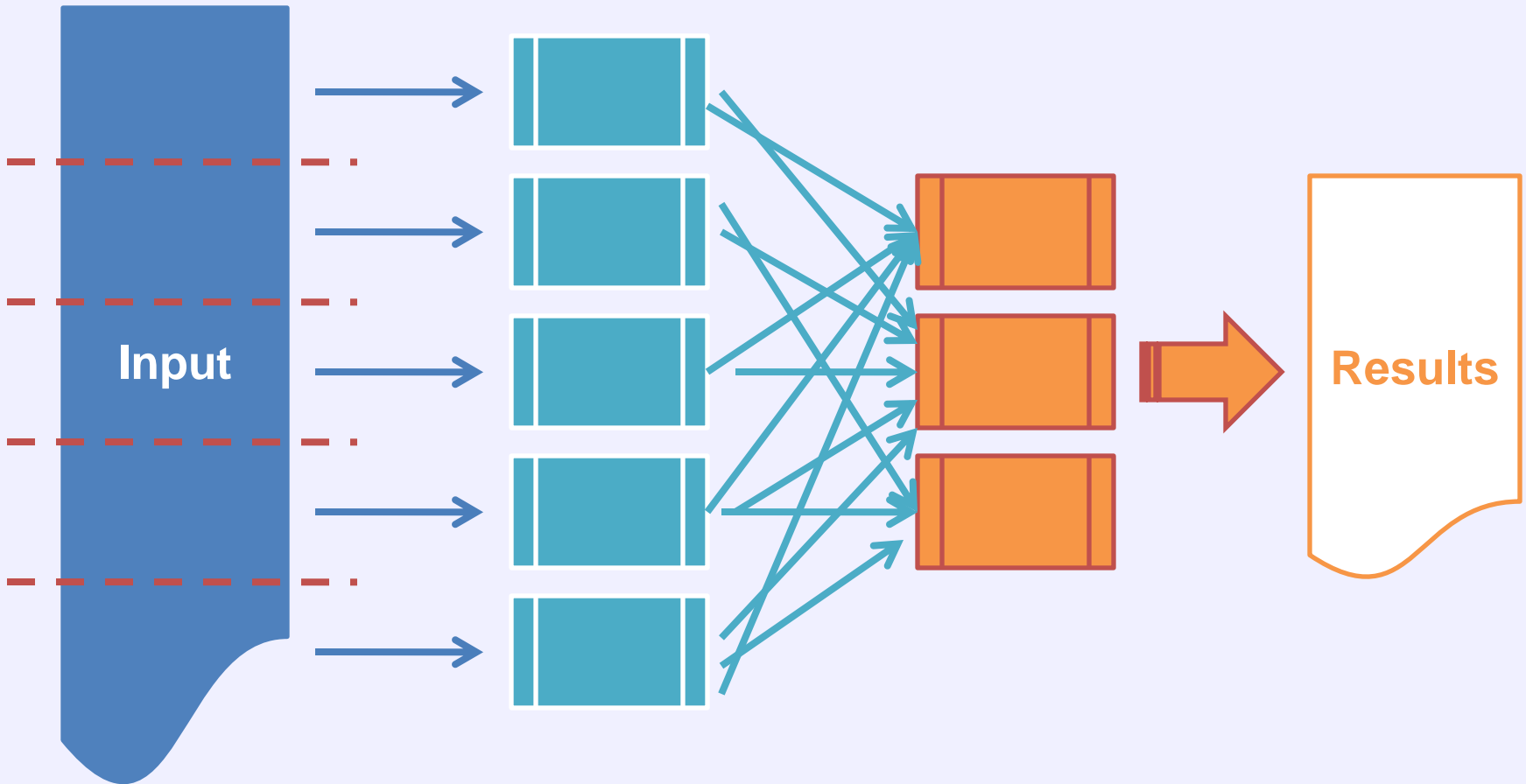
A Programmer's Nightmare



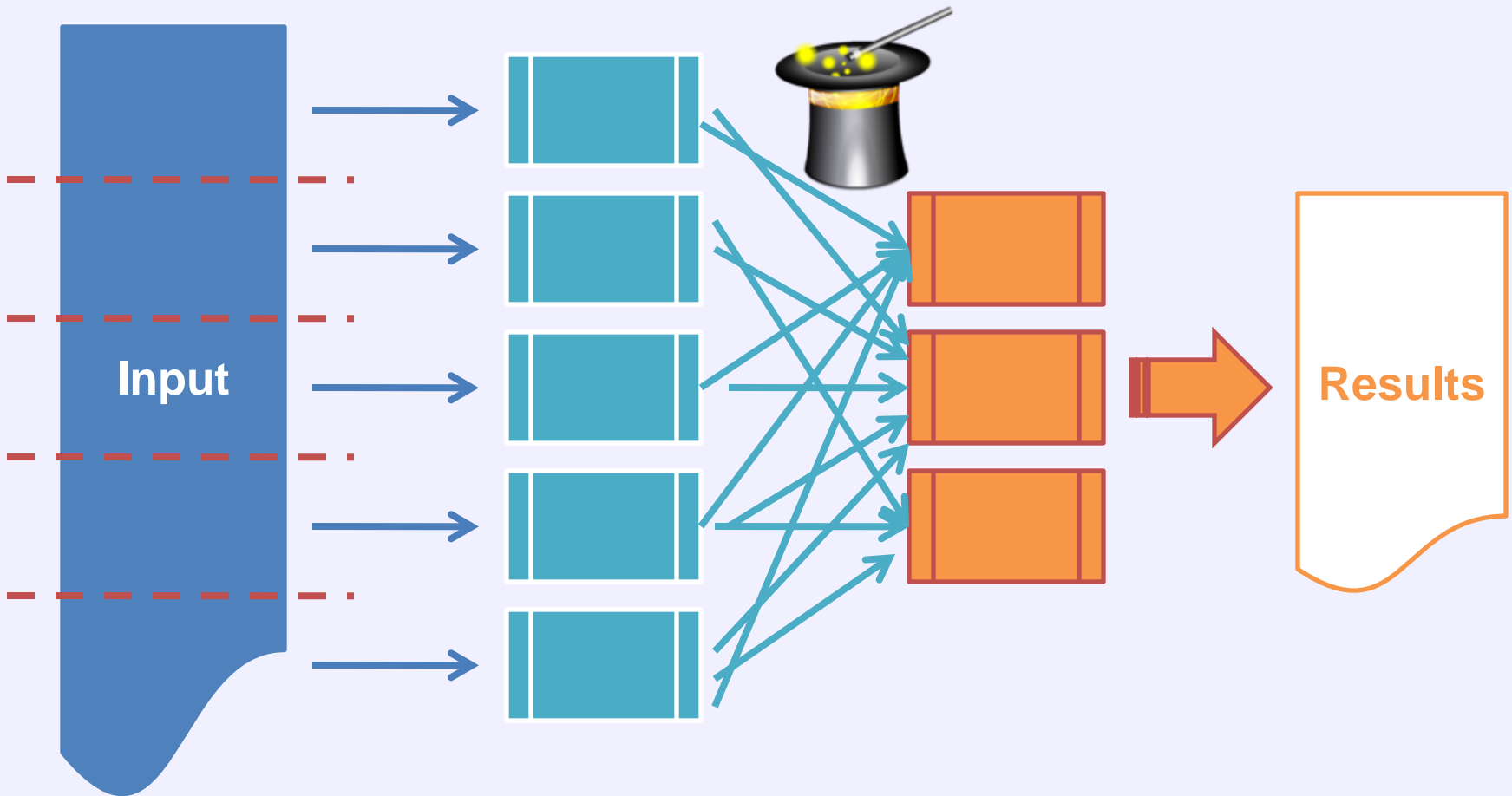
A Programmer's Nightmare



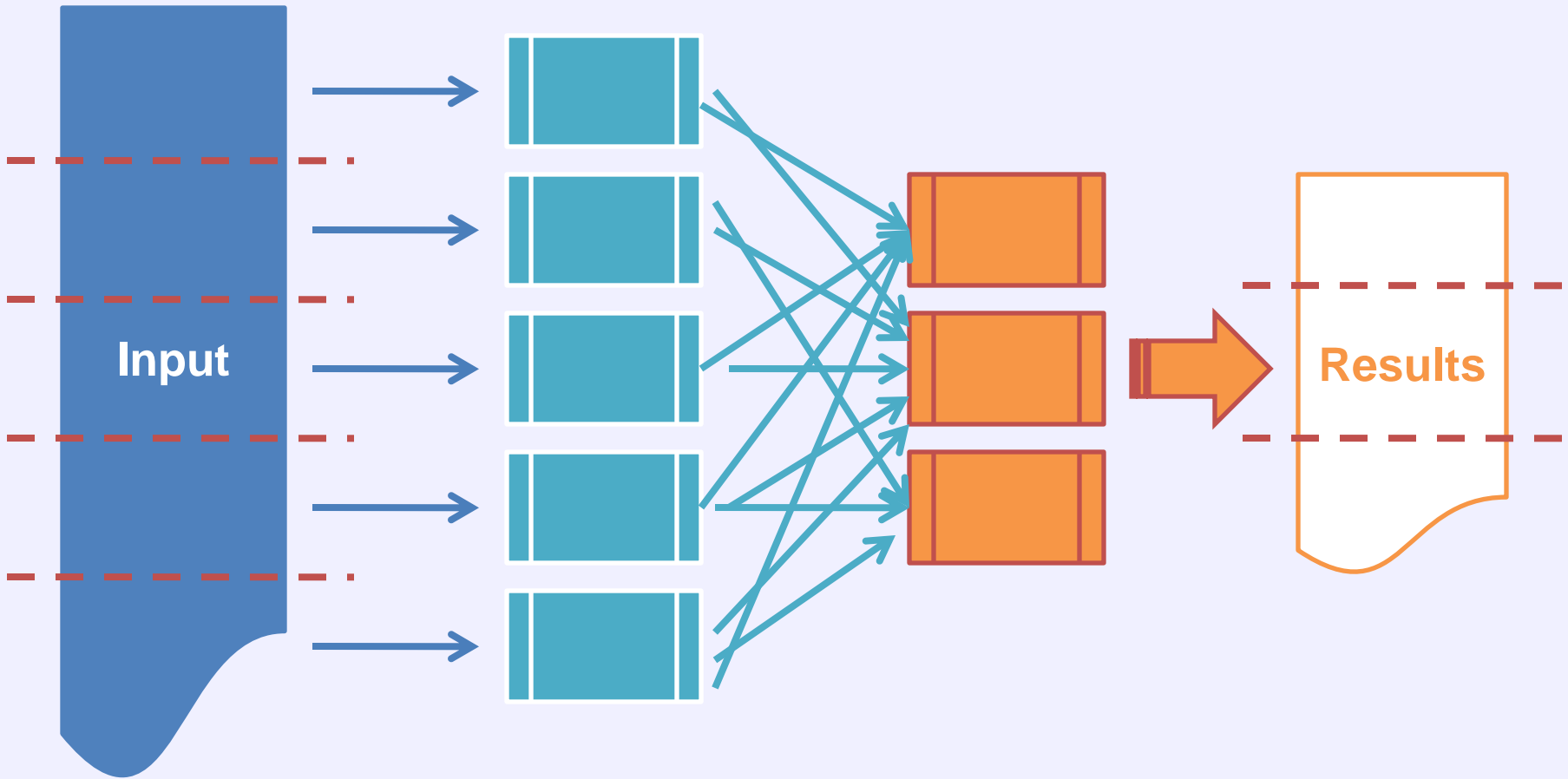
A Programmer's Nightmare



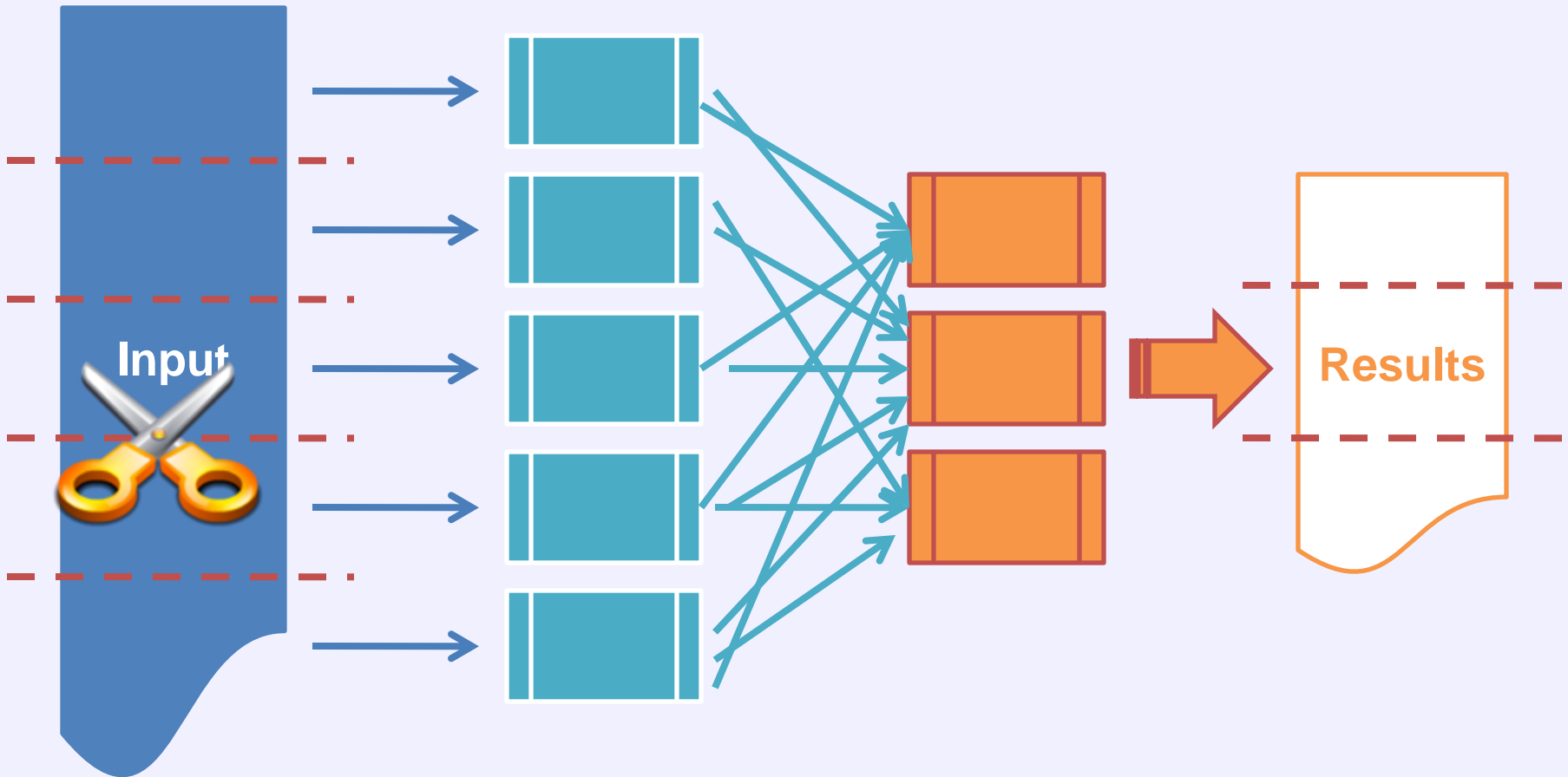
A Programmer's Nightmare



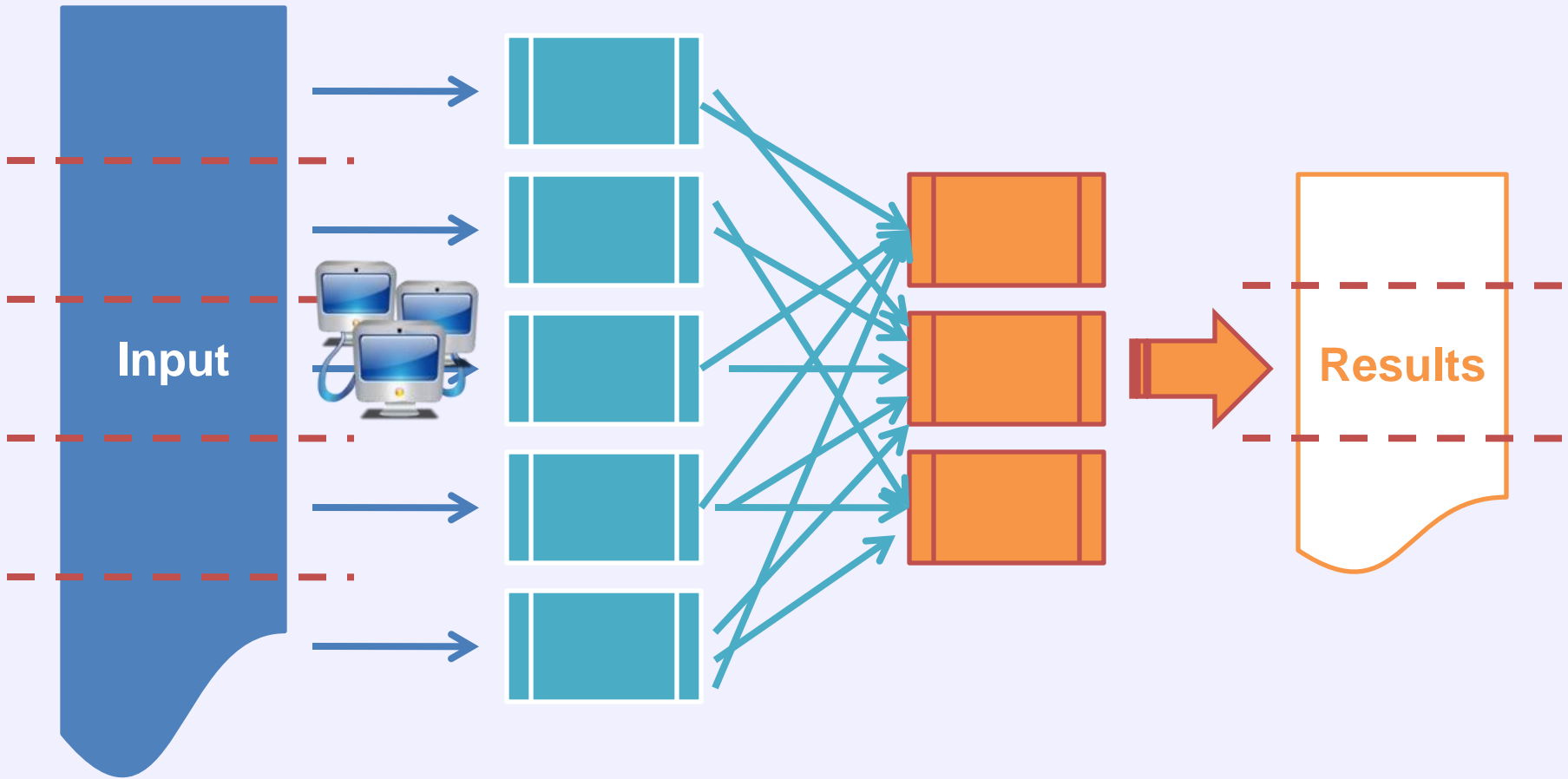
A Programmer's Nightmare



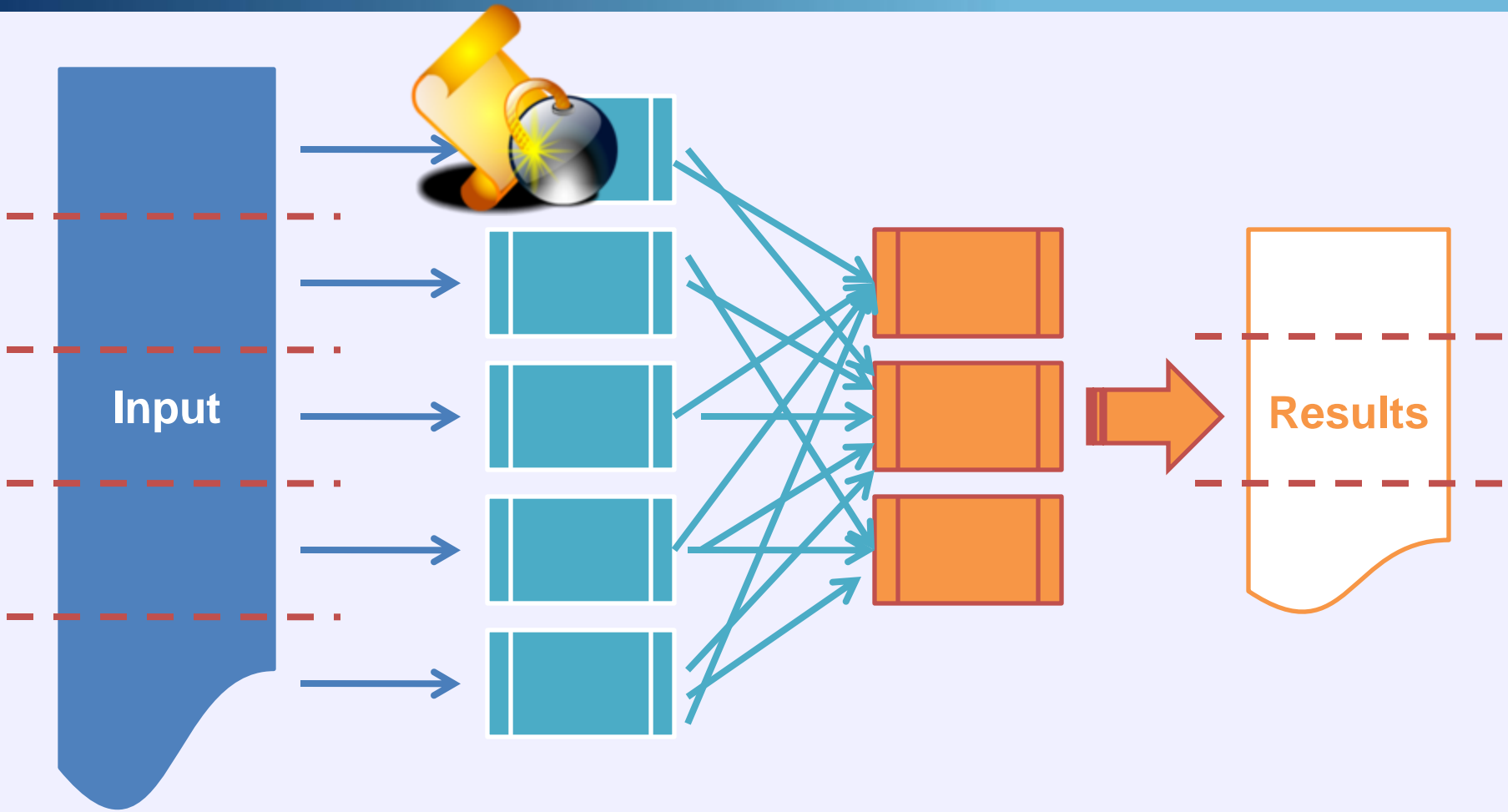
A Programmer's Nightmare



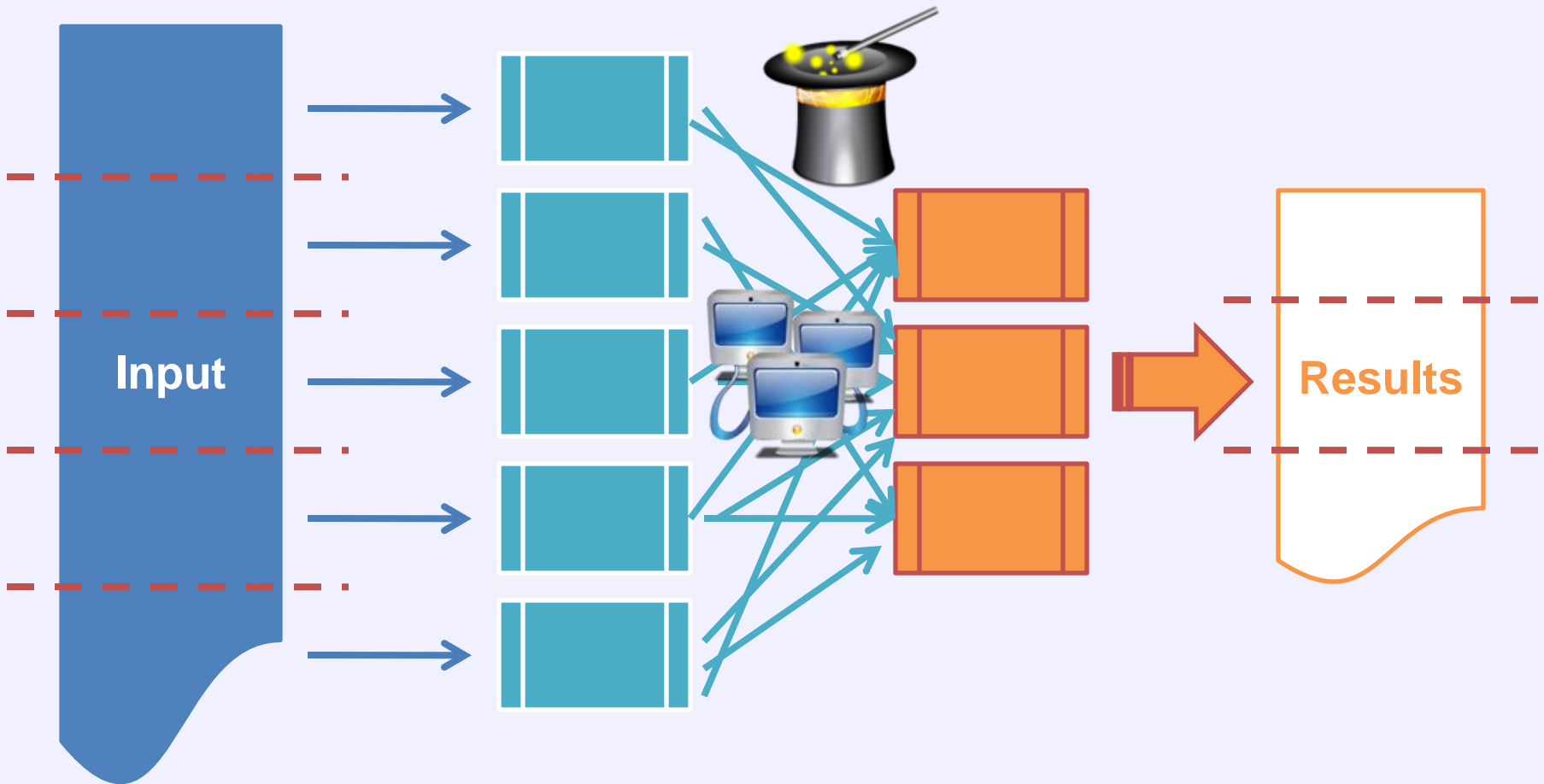
A Programmer's Nightmare



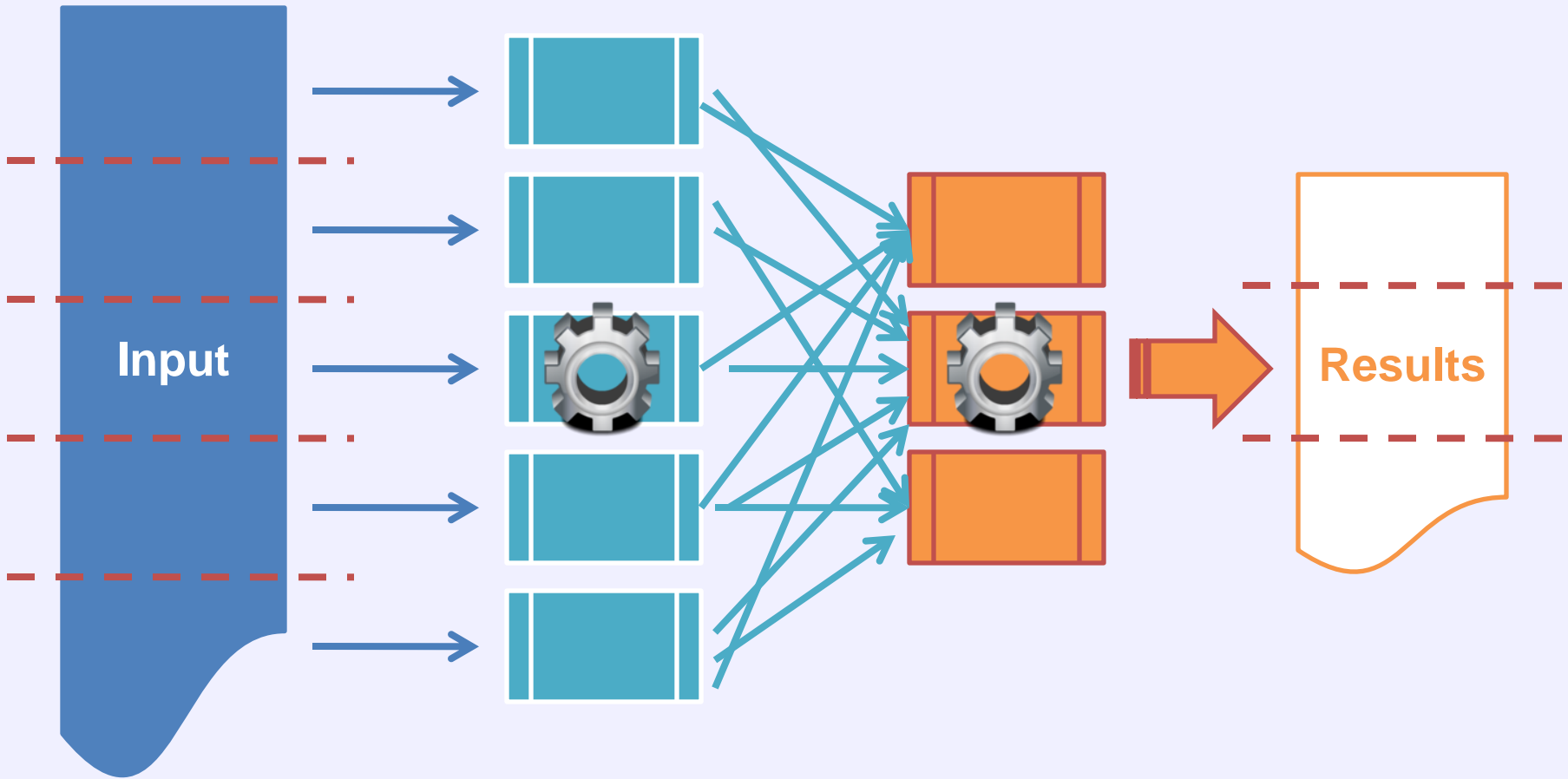
A Programmer's Nightmare



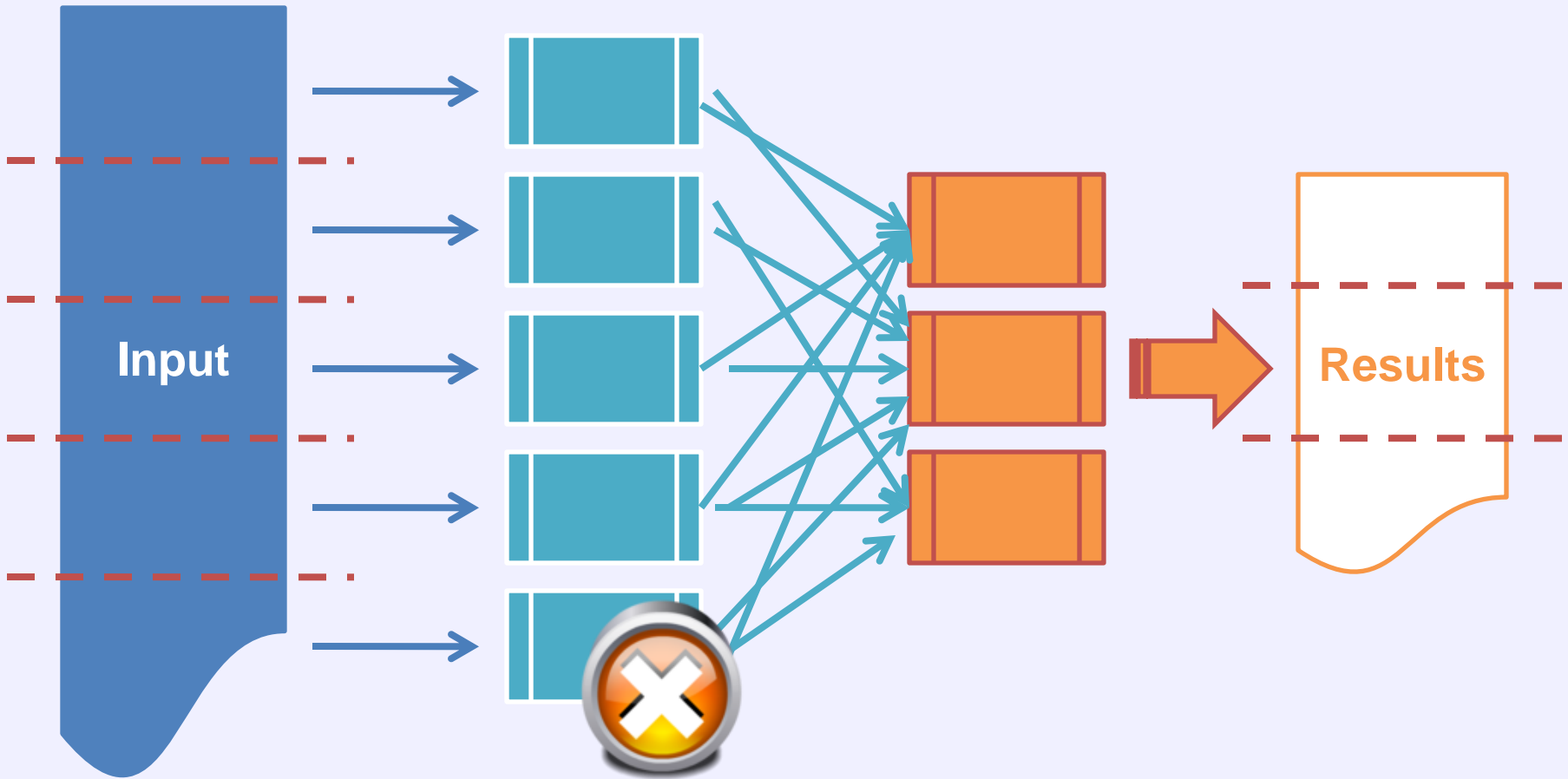
A Programmer's Nightmare



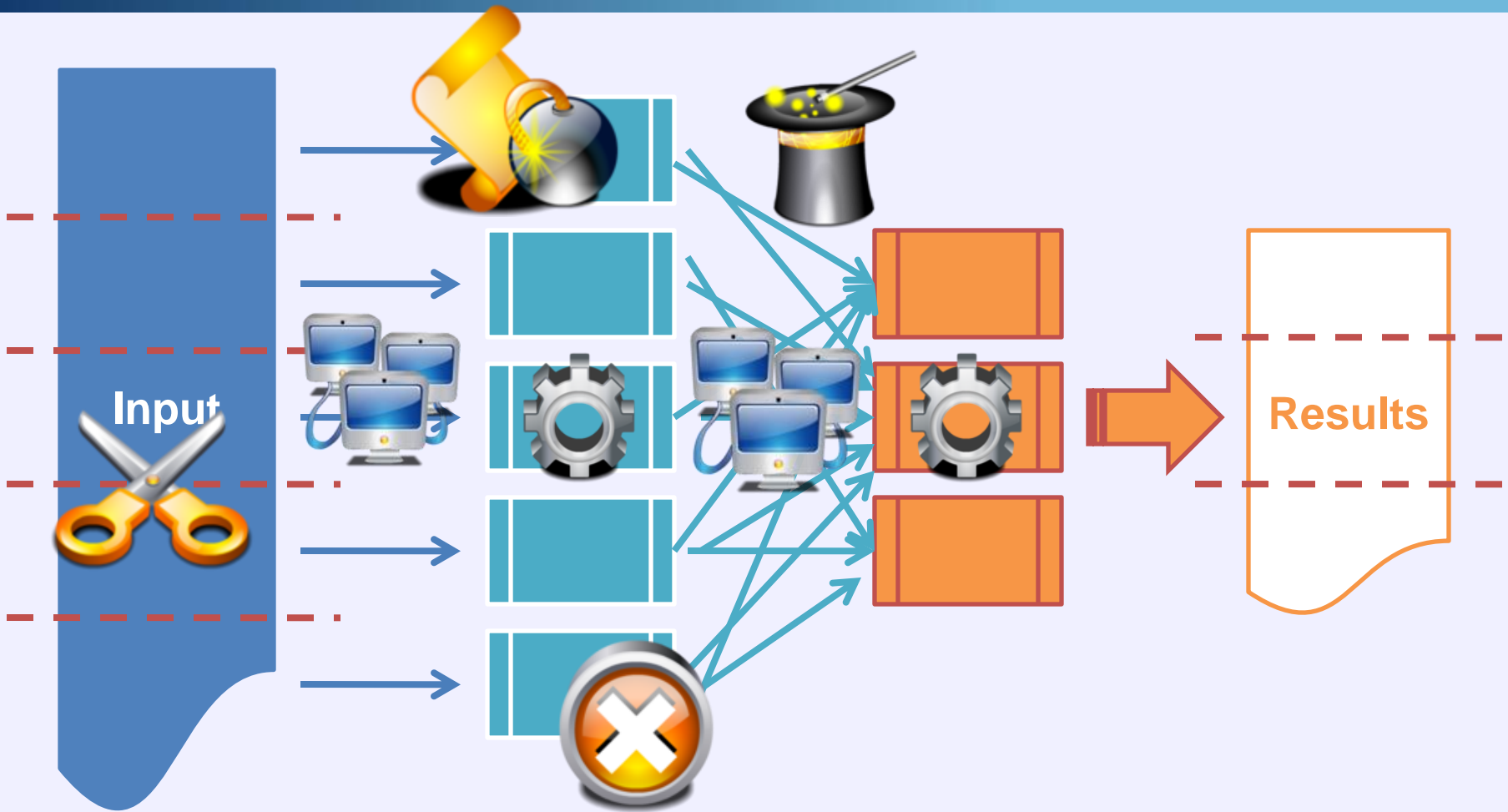
A Programmer's Nightmare



A Programmer's Nightmare



A Programmer's Nightmare



A Programmer's Nightmare

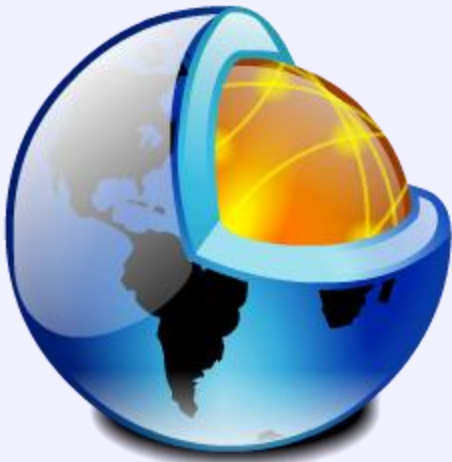
**Custom queries
take tons of custom
code**

Outline

- MapReduce – Back to its Cradle
- What MapReduce is and What it's Not
- The MapReduce Framework(s)
- Strengths and Weaknesses
- Summary

The Google World

Google works a lot on large Web bound data



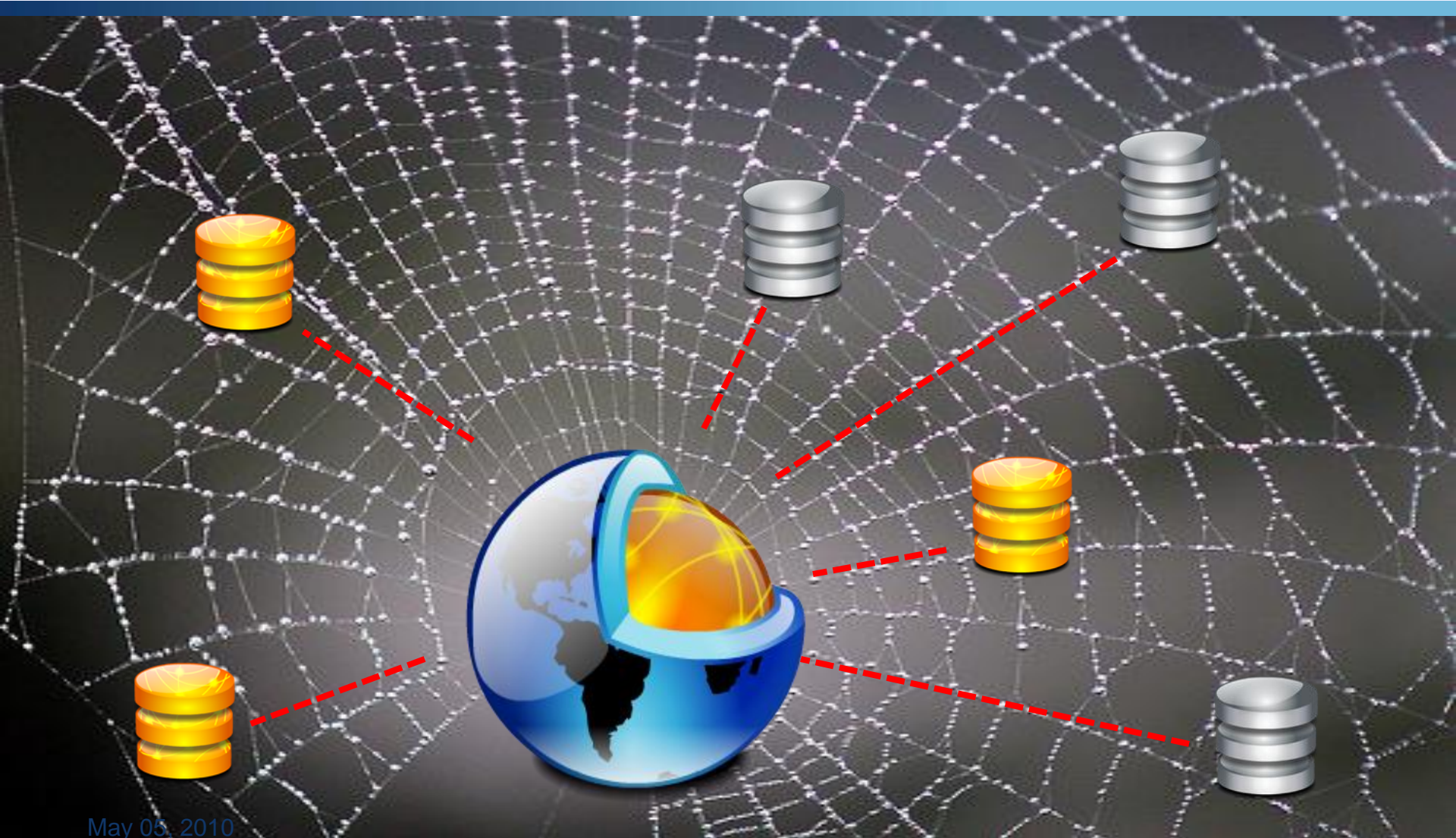
The Google World



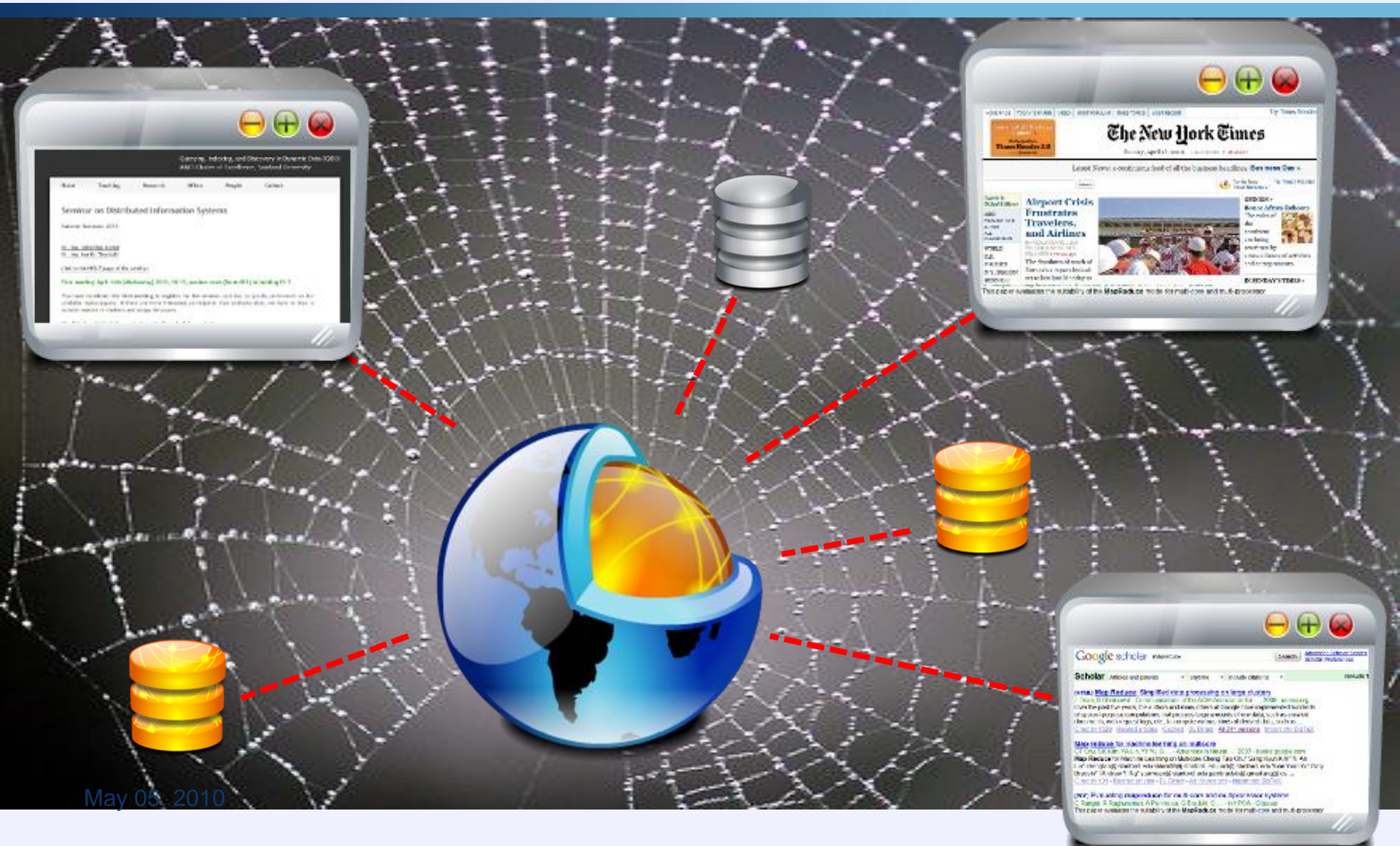
The Google World



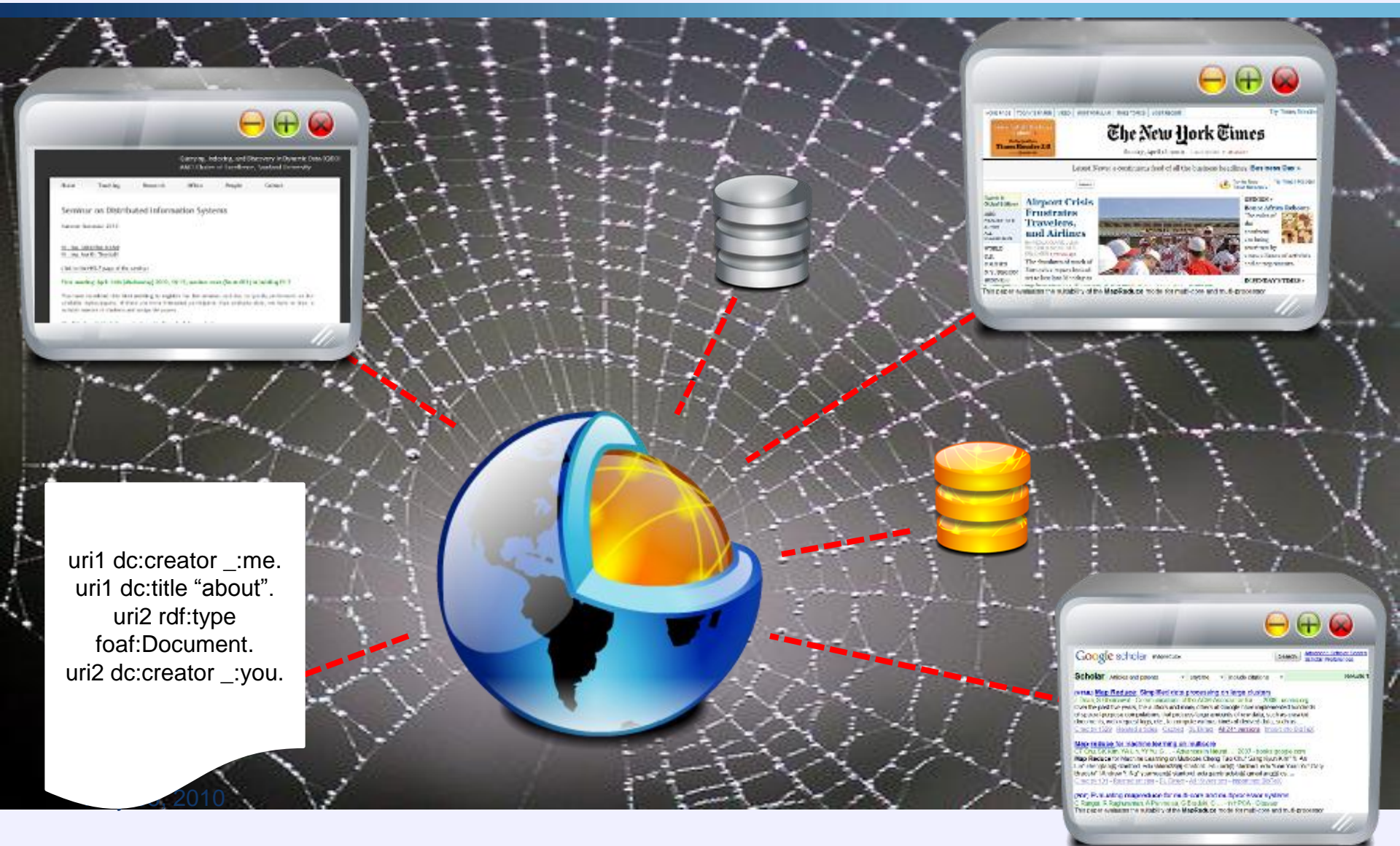
The Google World



The Google World



The Google World



uri1 dc:creator _:me.
uri1 dc:title "about".
uri2 rdf:type
foaf:Document.
uri2 dc:creator _:you.

The Google World



uri1 dc:creator _:me.
uri1 dc:title "about".
uri2 rdf:type
foaf:Document.
uri2 dc:creator _:you.



The Google World



uri1 dc:creator _:me.
uri1 dc:title "about".
uri2 rdf:type
foaf:Document.
uri2 dc:creator _:you.



The Google Way

- Cheap commodity hardware
 - Huge number of nodes
 - Inexpensive disks
 - Commodity networking HW
 - High failure rates
- Specific needs
 - Work with very large data from the Web
- Build custom systems



*

Some Google Systems

- GFS (Google File System)^[2]
 - Distributed file system
- Bigtable^[3]
 - The structured data “special case”
 - Based on GFS
- Custom query programs
 - Originally hand-written
 - Using some libraries
- Often unstructured or semi-structured data

^[2] Senjay Ghemawat, : “The Google File System”. *SIGOPS Operating Systems Review* 37(5), 2003

^[3] Fay Chang et al.: “Bigtable: A Distributed Storage System for Structured Data” in *OSDI* 2006

Some Google Systems

- GFS (Google File System)^[2]
 - Distributed file system
- Bigtable^[3]
 - The structured data “special case”
 - Based on GFS
- Custom query programs
 - Originally hand-written
 - Using some libraries
- Often unstructured or semi-structured data

^[2] Senjay Ghemawat, : “The Google File System”. *SIGOPS Operating Systems Review* 37(5), 2003

^[3] Fay Chang et al.: “Bigtable: A Distributed Storage System for Structured Data” in *OSDI* 2006

Some Google Systems

- GFS (Google File System)^[2]
 - Distributed file system
 - Bigtable^[3]
 - The structured data “special case”
 - Based on GFS
 - Custom query programs
 - Originally hand-written
 - Using some libraries
- Often unstructured or semi-structured data

^[2] Senjay Ghemawat, : “The Google File System”. *SIGOPS Operating Systems Review* 37(5), 2003

^[3] Fay Chang et al.: “Bigtable: A Distributed Storage System for Structured Data” in *OSDI* 2006

Some Google Systems


- GFS (Google File System)^[2]
 - Distributed file system
- Bigtable^[3]
 - The structured data “special case”
 - Based on GFS
- Custom query programs
 - Originally hand-written
 - Using some libraries

■ Often unstructured or semi-structured data

^[2] Senjay Ghemawat, : “The Google File System”. *SIGOPS Operating Systems Review* 37(5), 2003

^[3] Fay Chang et al.: “Bigtable: A Distributed Storage System for Structured Data” in *OSDI* 2006

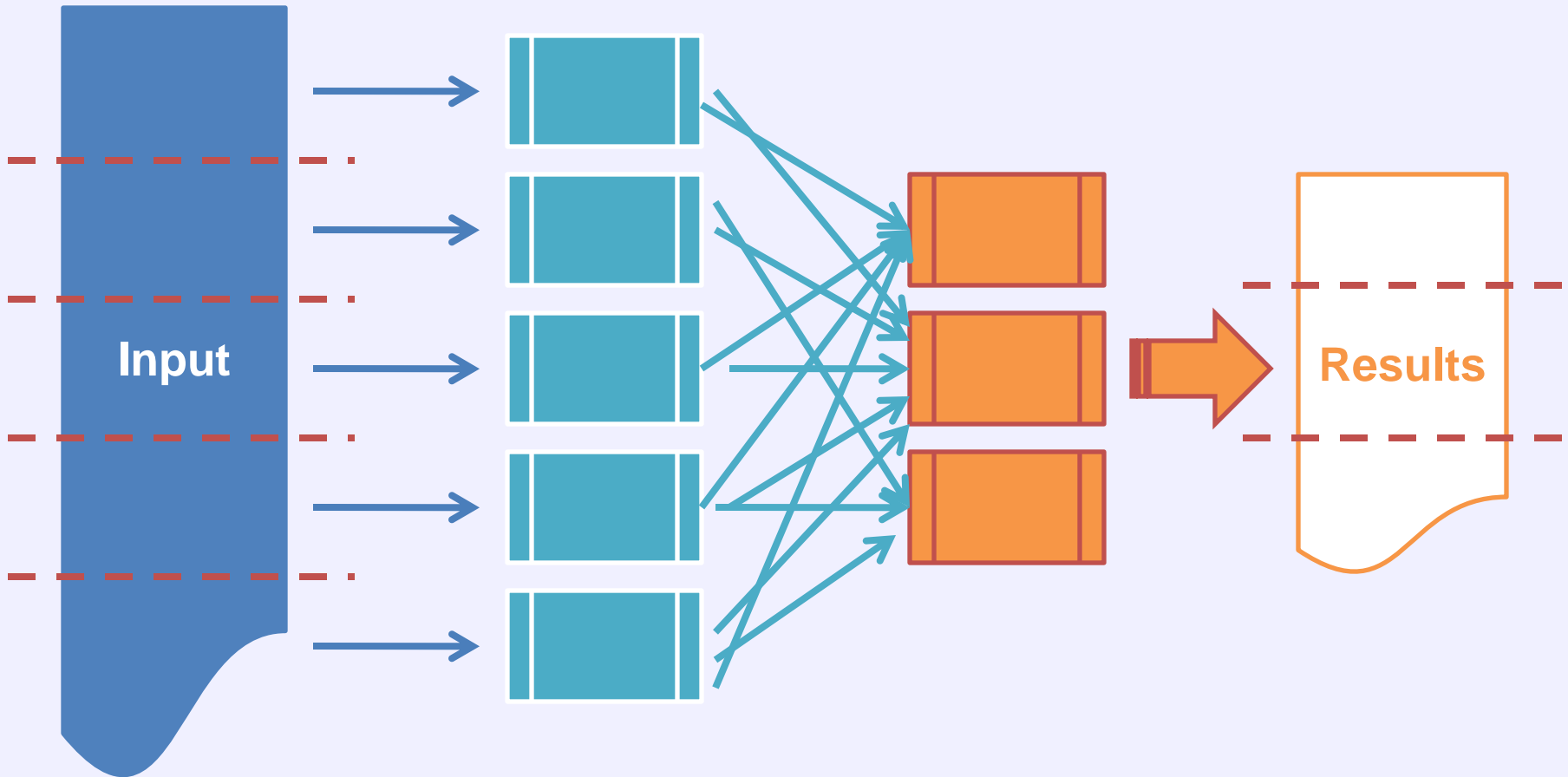
Architectural Redundancy



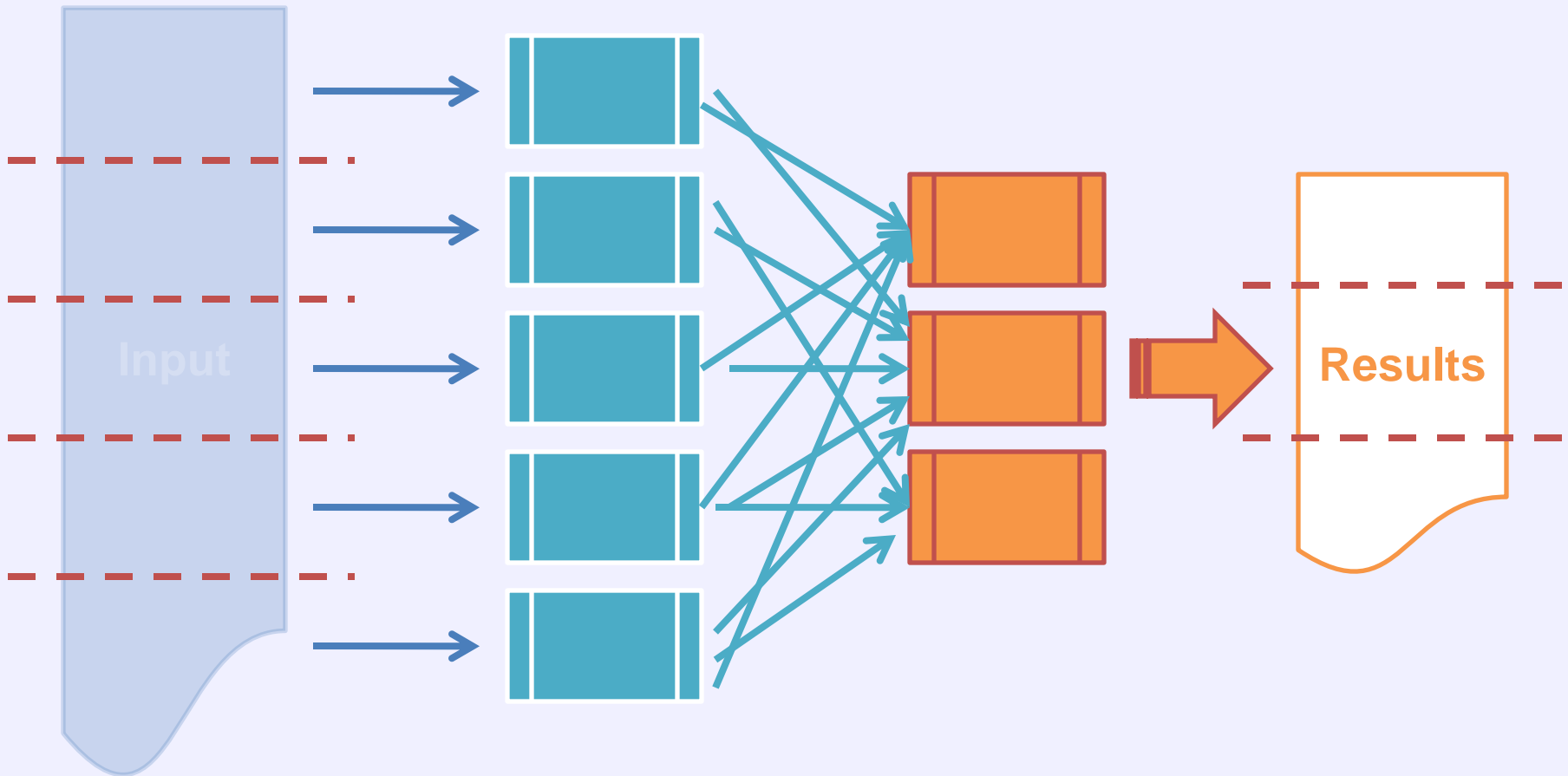
The diagram features a central blue box with white text. To the left is a blue vertical bar with a rounded bottom, labeled 'Input', with four horizontal dashed red lines. To the right is an orange-outlined box with a rounded bottom, labeled 'Results', with two horizontal dashed red lines. A light blue horizontal bar is positioned above and below the central box. The background is white with a dark blue horizontal line at the top.

**Custom queries
take tons of custom
code**

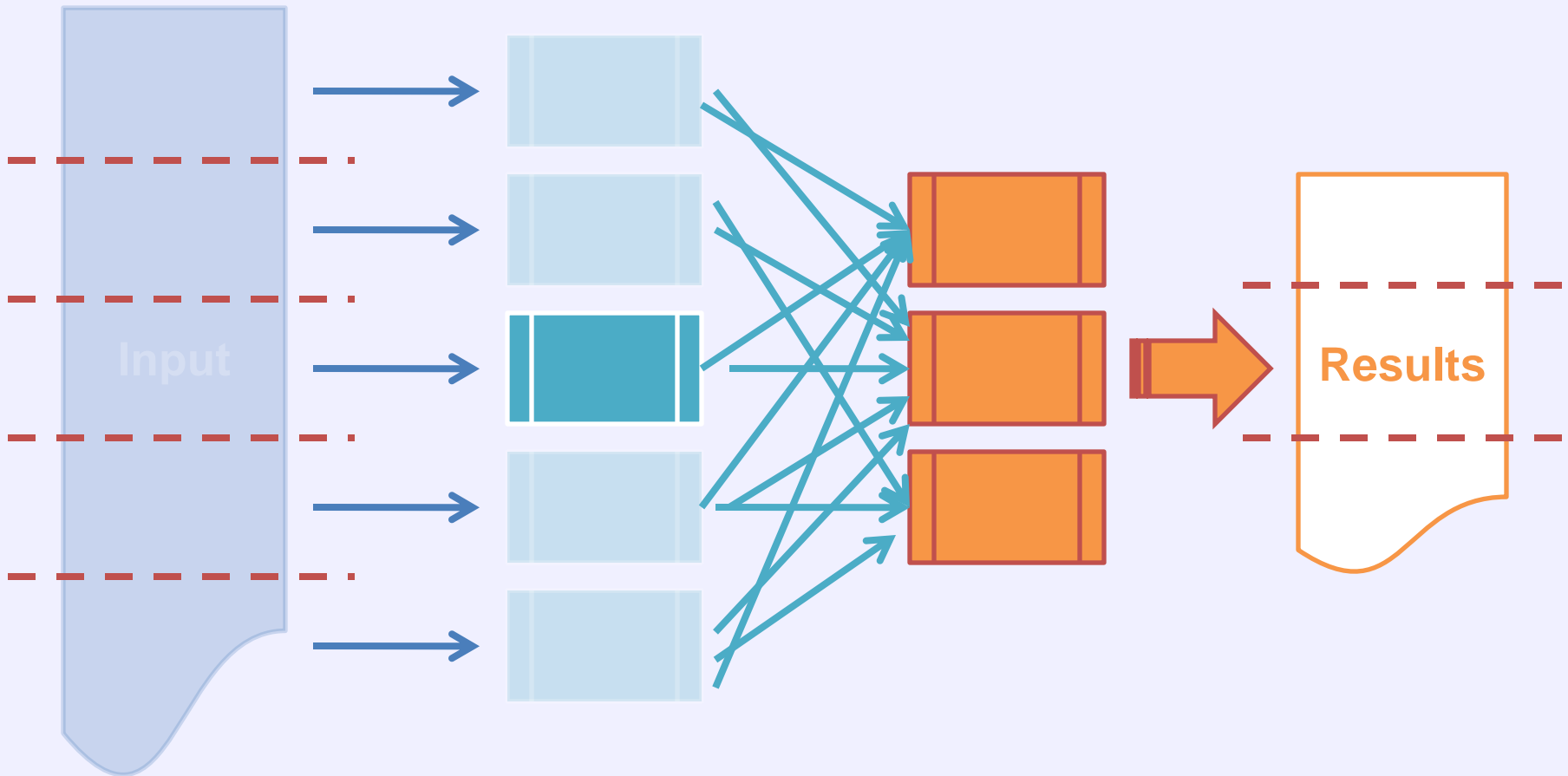
Architectural Redundancy



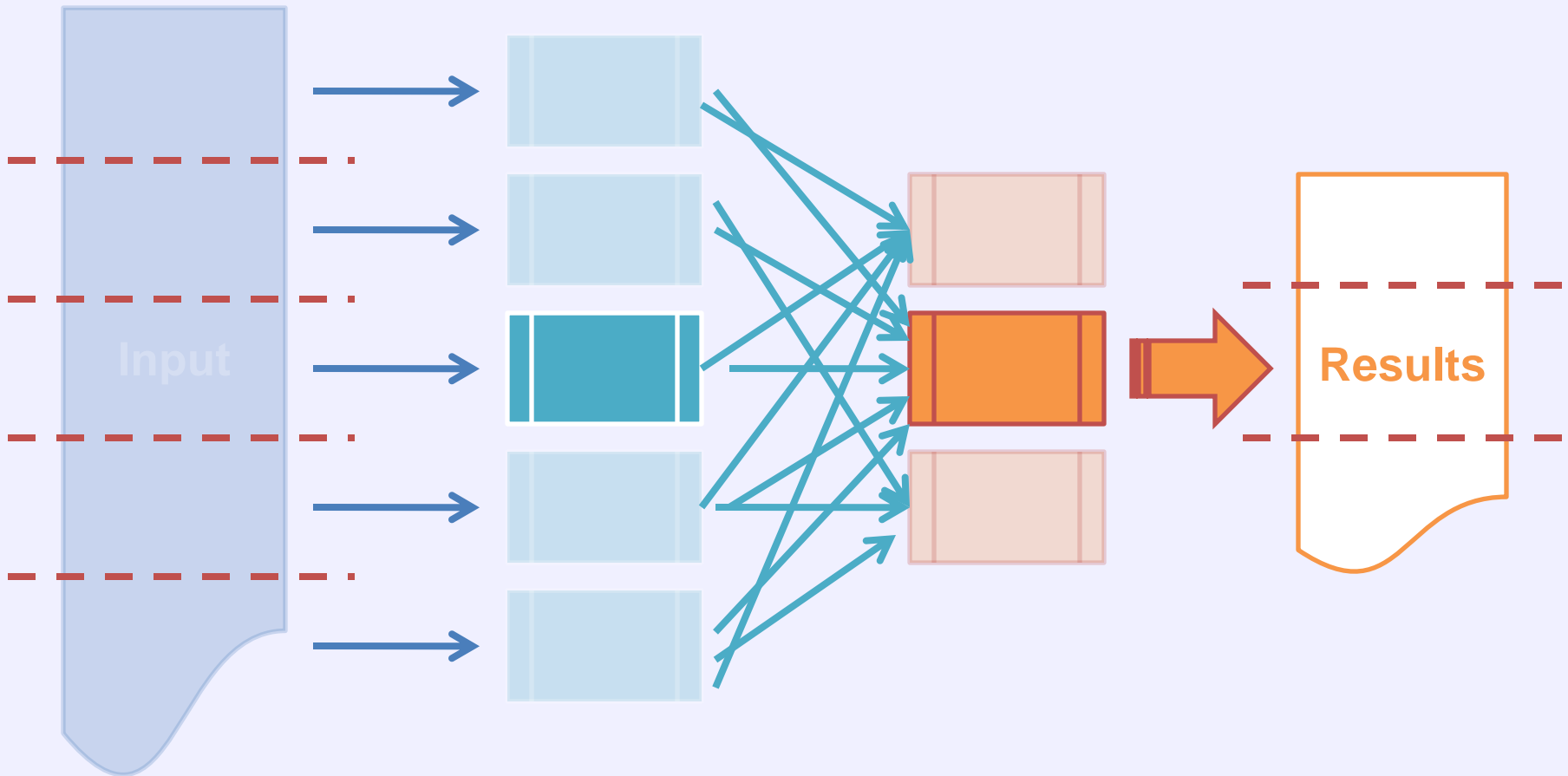
Architectural Redundancy



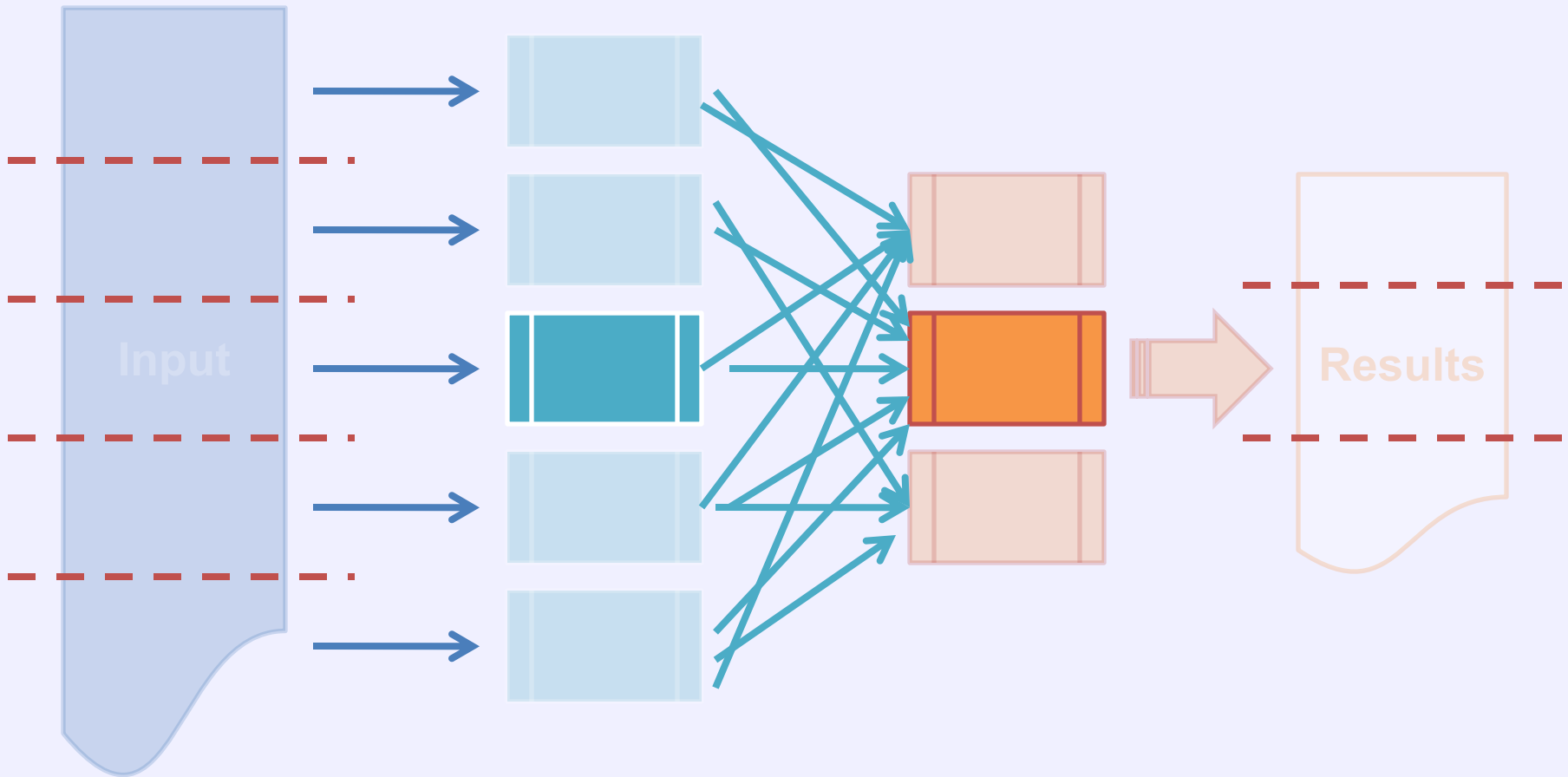
Architectural Redundancy



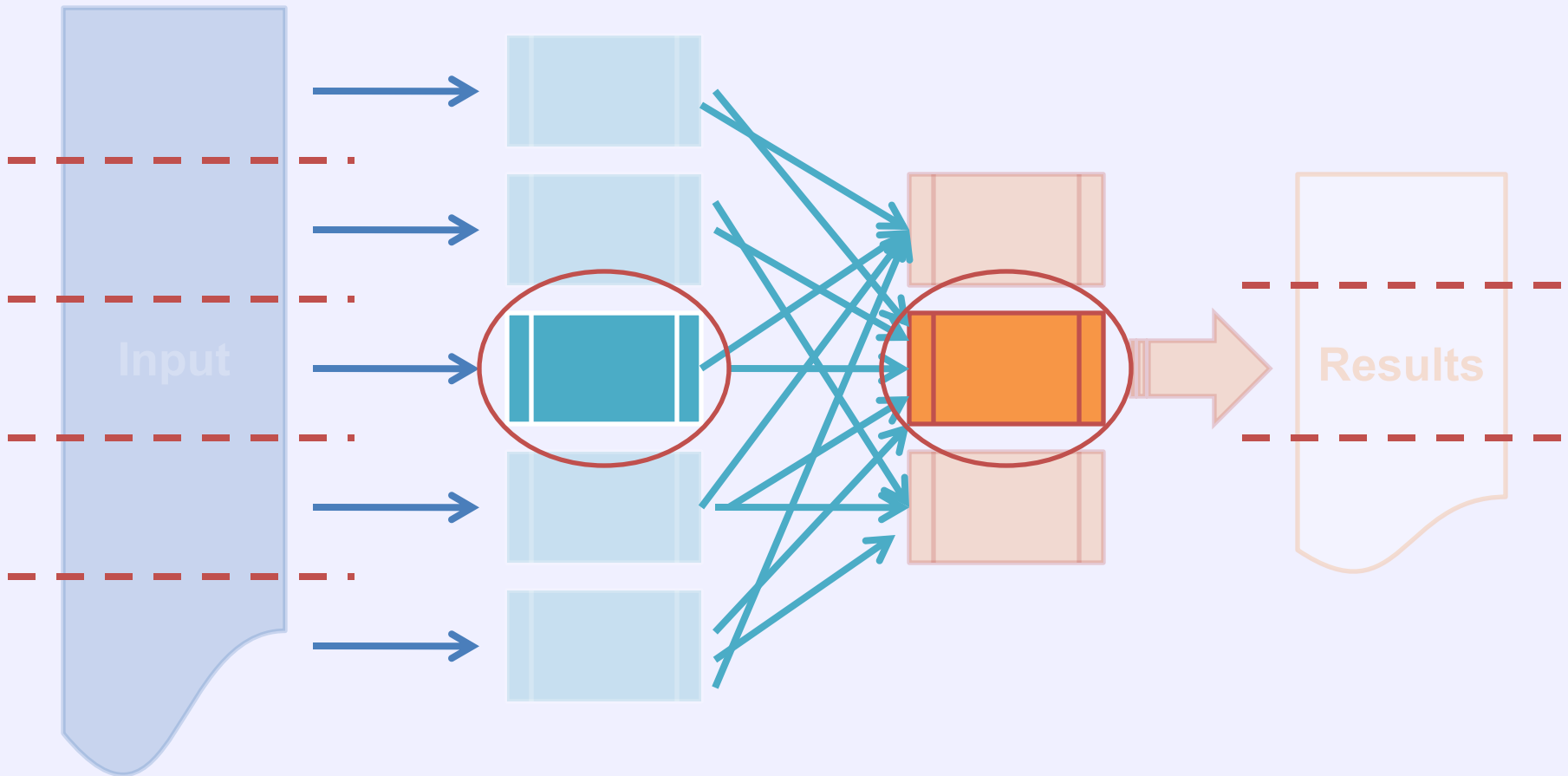
Architectural Redundancy



Architectural Redundancy



Architectural Redundancy



Data Level Redundancy



- Takes records
 - one by one
 - key, value
- Processes records
 - Independently
- Outputs intermediate
 - 1..n per input record
 - key', value'

Data Level Redundancy



- Takes records
 - one by one
 - key, value
- Processes records
 - Independently
- Outputs intermediate
 - 1..n per input record
 - key', value'

Data Level Redundancy



- Takes records
 - one by one
 - key, value
- Processes records
 - Independently
- Outputs intermediate
 - 1..n per input record
 - key', value'

Data Level Redundancy



- Takes records
 - one by one
 - key, value
- Processes records
 - Independently
- Outputs intermediate
 - 1..n per input record
 - key', value'

Data Level Redundancy

map

- Takes records
 - one by one
 - key, value
- Processes records
 - Independently
- Outputs intermediate
 - 1..n per input record
 - key', value'




Data Level Redundancy



map

- Takes records
 - one by one
 - key, value
- Processes records
 - Independently
- Outputs intermediate
 - 1..n per input record
 - key', value'


- 
- Takes intermediate
 - Groups with same key
 - key', value'[]
 - Processes records
 - Group-wise
 - Outputs result
 - Per group
 - Any format

Data Level Redundancy

The word "map" is written in orange lowercase letters inside a teal rectangular box with a thin white border.

map

- Takes records
 - one by one
 - key, value
- Processes records
 - Independently
- Outputs intermediate
 - 1..n per input record
 - key', value'


- 
- An orange rectangular box with a thin white border, representing a grouped map operation.
- Takes intermediate
 - Groups with same key
 - key', value'[]
 - Processes records
 - Group-wise
 - Outputs result
 - Per group
 - Any format

Data Level Redundancy

The word "map" is written in orange lowercase letters inside a light blue rounded rectangle with a thin white border.

map

- Takes records
 - one by one
 - key, value
- Processes records
 - Independently
- Outputs intermediate
 - 1..n per input record
 - key', value'

- 
- An orange rounded rectangle with a thin white border, representing a grouped map operation.
- Takes intermediate
 - Groups with same key
 - key', value'[]
 - Processes records
 - Group-wise
 - Outputs result
 - Per group
 - Any format

Data Level Redundancy

map

- Takes records
 - one by one
 - key, value
- Processes records
 - Independently
- Outputs intermediate
 - 1..n per input record
 - key', value'

reduce

- Takes intermediate
 - Groups with same key
 - key', value'[]
- Processes records
 - Group-wise
- Outputs result
 - Per group
 - Any format

Outline

- MapReduce – Back to its Cradle
- What MapReduce is and What it's Not
- The MapReduce Framework(s)
- Strengths and Weaknesses
- Summary

What is MapReduce?



Framework?



hadoop

Google

Some Google System?

map()
reduce()
Programming Paradigm?

What is MapReduce?

Confused?

What is MapReduce?

- It **is** a framework
 - Though some people argue that it is not
- It **is** a programming paradigm
 - Though it is not really novel and rather trivial
- It is **partially defined by** its systems
 - Though it is not Hadoop (nor Google MR)
- It **is loosely defined**
 - Even in the original paper, and ever since

What is MapReduce?

- It **is** a framework
 - Though some people argue that it is not
- It **is** a programming paradigm
 - Though it is not really novel and rather trivial
- It is **partially defined by** its systems
 - Though it is not Hadoop (nor Google MR)
- It **is loosely defined**
 - Even in the original paper, and ever since

What is MapReduce?

- It **is** a framework
 - Though some people argue that it is not
- It **is** a programming paradigm
 - Though it is not really novel and rather trivial
- It is **partially defined by** its systems
 - Though it is not Hadoop (nor Google MR)
- **It is loosely defined**
 - Even in the original paper, and ever since

What is MapReduce?

- It **is** a framework
 - Though some people argue that it is not
- It **is** a programming paradigm
 - Though it is not really novel and rather trivial
- It is **partially defined by** its systems
 - Though it is not Hadoop (nor Google MR)
- **It is loosely defined**
 - Even in the original paper, and ever since

“Inspired by...”

- `map()` & `reduce()` in functional programming
- `(map (lambda (x) (* x x)) '(1 2 3))`
→ `'(1 4 9)`
- `(reduce + 0 '(1 2 3))` → 6
- Very similar concepts

“Inspired by...”

- `map()` & `reduce()` in functional programming
- `(map (lambda (x) (* x x)) '(1 2 3))`
→ `'(1 4 9)`
- `(reduce + 0 '(1 2 3))` → 6
- Very similar concepts

“Inspired by...”

- `map()` & `reduce()` in functional programming
- `(map (lambda (x) (* x x)) '(1 2 3))`
→ `'(1 4 9)`
- `(reduce + 0 '(1 2 3))` → 6
- Very similar concepts

“Inspired by...”

- `map()` & `reduce()` in functional programming
- `(map (lambda (x) (* x x)) '(1 2 3))`
→ `'(1 4 9)`
- `(reduce + 0 '(1 2 3))` → 6
- Very similar concepts

“Inspired by...”

- `map()` & `reduce()` in functional programming
- `(map (lambda (x) (* x x)) '(1 2 3))`
→ `'(1 4 9)`
- `(reduce + 0 '(1 2 3))` → 6
- Very similar concepts

“Inspired by...”

- `map()` & `reduce()` in functional programming
- `(map (lambda (x) (* x x)) '(1 2 3))` → `'(1 4 9)`
- `(reduce + 0 '(1 2 3))` → `6`
- Very similar concepts

“Inspired by...”

- `map()` & `reduce()` in functional programming
- `(map (lambda (x) (* x x)) '(1 2 3))`
→ `'(1 4 9)`
- `(reduce + 0 '(1 2 3))` → 6
- Very similar concepts

“Inspired by...”

- `map()` & `reduce()` in functional programming
- `(map (lambda (x) (* x x)) '(1 2 3))`
→ `'(1 4 9)`
- `(reduce + 0 '(1 2 3))` → 6
- Very similar concepts

“Inspired by...”

- `map()` & `reduce()` in functional programming
- `(map (lambda (x) (* x x)) '(1 2 3))`
→ `'(1 4 9)`
- `(reduce + 0 '(1 2 3))` → 6
- Very similar concepts

“Inspired by...”

- `map()` & `reduce()` in functional programming
- `(map (lambda (x) (* x x)) '(1 2 3))`
→ `'(1 4 9)`
- `(reduce + 0 '(1 2 3))` → 6
- Very similar concepts

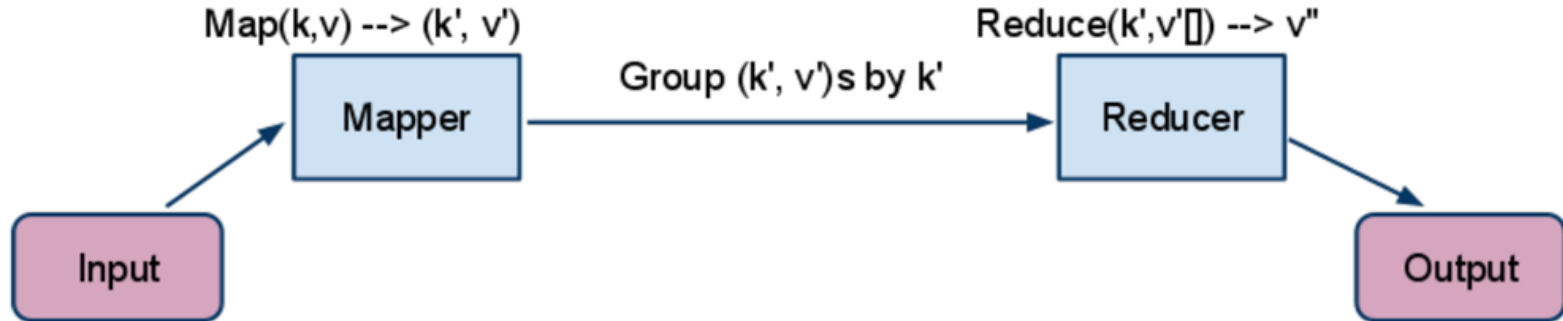
“Inspired by...”

- `map()` & `reduce()` in functional programming
 - `(map (lambda (x) (* x x)) '(1 2 3))`
→ `'(1 4 9)`
 - `(reduce + 0 '(1 2 3))` → 6
 - Very similar concepts
- “[MapReduce] is inspired by the *map* and *reduce* primitives present in Lisp”
(Dean/Ghemawat)^[1]

Programming Model



Inspired by Map/Reduce in functional programming languages, such as LISP from 1960's, but not equivalent



Map & Reduce Elsewhere

- `(map (map-udf) '((k1,v1) (k2,v2)))`
→ `'((ik1,iv1) (ik2,iv2))`
- `(reduce (reduce-udf) '((ik1,iv1) ...))`
→ `result`
- Concept present in basically all functional programming languages
- Implemented in other languages (Python)

Map & Reduce Elsewhere

- `(map (map-udf) '((k1,v1) (k2,v2)))`
→ `'((ik1,iv1) (ik2,iv2))`
- `(reduce (reduce-udf) '((ik1,iv1) ...))`
→ `result`
- Concept present in basically all functional programming languages
- Implemented in other languages (Python)

Map & Reduce Elsewhere

- `(map (map-udf) '((k1, v1) (k2, v2)))`
→ `'((ik1, iv1) (ik2, iv2))`
- `(reduce (reduce-udf) '((ik1, iv1) ...))`
→ `result`
- Concept present in basically all functional programming languages
- Implemented in other languages (Python)

Map & Reduce Elsewhere

- `(map (map-udf) '((k1,v1) (k2,v2)))`
→ `'((ik1,iv1) (ik2,iv2))`
- `(reduce (reduce-udf) '((ik1,iv1) ...))`
→ `result`
- Concept present in basically all functional programming languages
- Implemented in other languages (Python)

Map & Reduce Elsewhere

- `(map (map-udf) '((k1,v1) (k2,v2)))`
→ `'((ik1,iv1) (ik2,iv2))`
- `(reduce (reduce-udf) '((ik1,iv1) ...))`
→ `result`

- Concept present in basically all functional programming languages
- Implemented in other languages (Python)

MapReduce Semantics

- Semantics have been analyzed^[4]
- Using Haskell to model
- Comparing with map and reduce in FP

MapReduce Semantics

- Semantics have been analyzed^[4]
- Using Haskell to model
- Comparing with map and reduce in FP

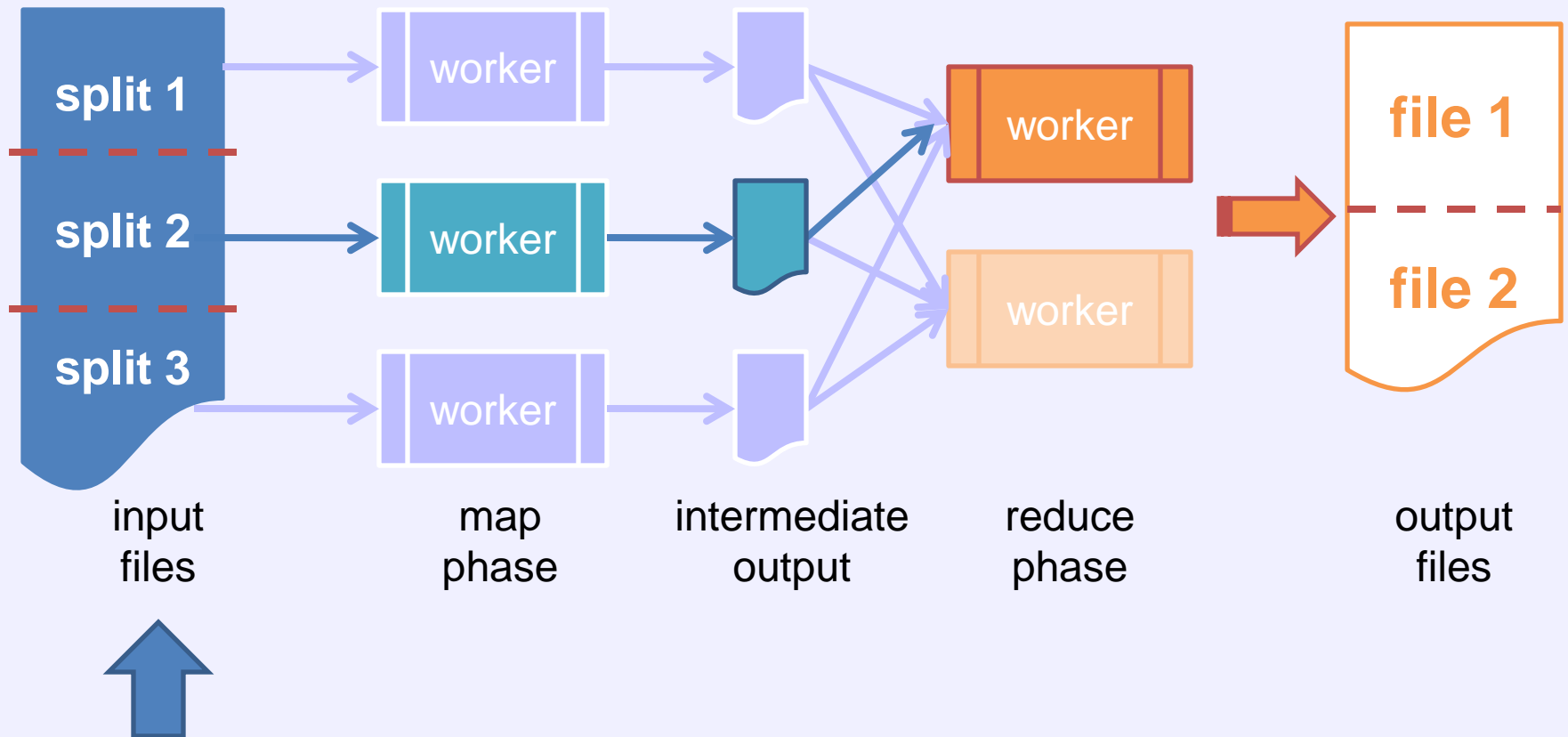
MapReduce Semantics

- Semantics have been analyzed^[4]
 - Using Haskell to model
 - Comparing with map and reduce in FP
- Google's MapReduce is essentially a special case of map/reduce in FP

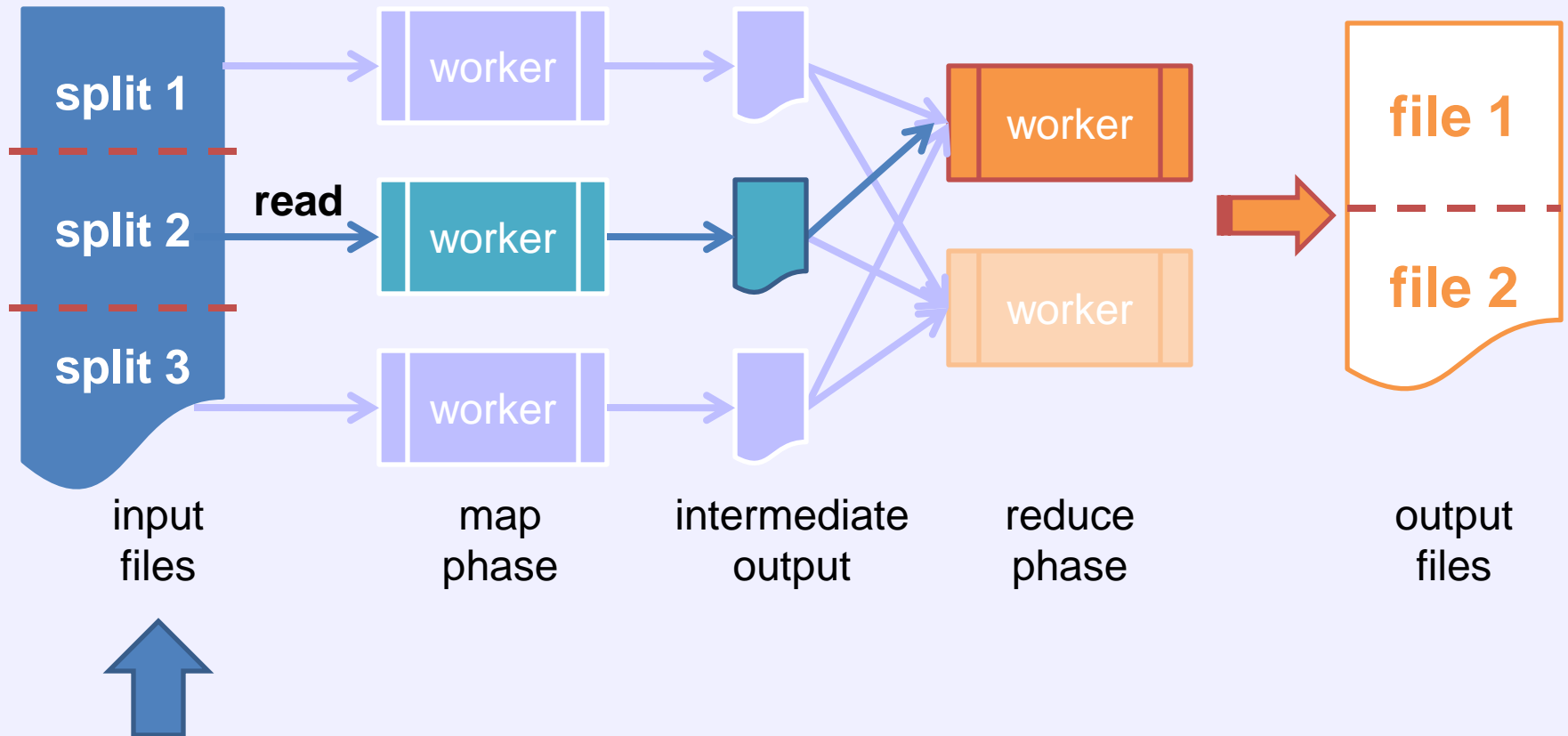
Outline

- MapReduce – Back to its Cradle
- What MapReduce is and What it's Not
- The MapReduce Framework(s)
- Strengths and Weaknesses
- Summary

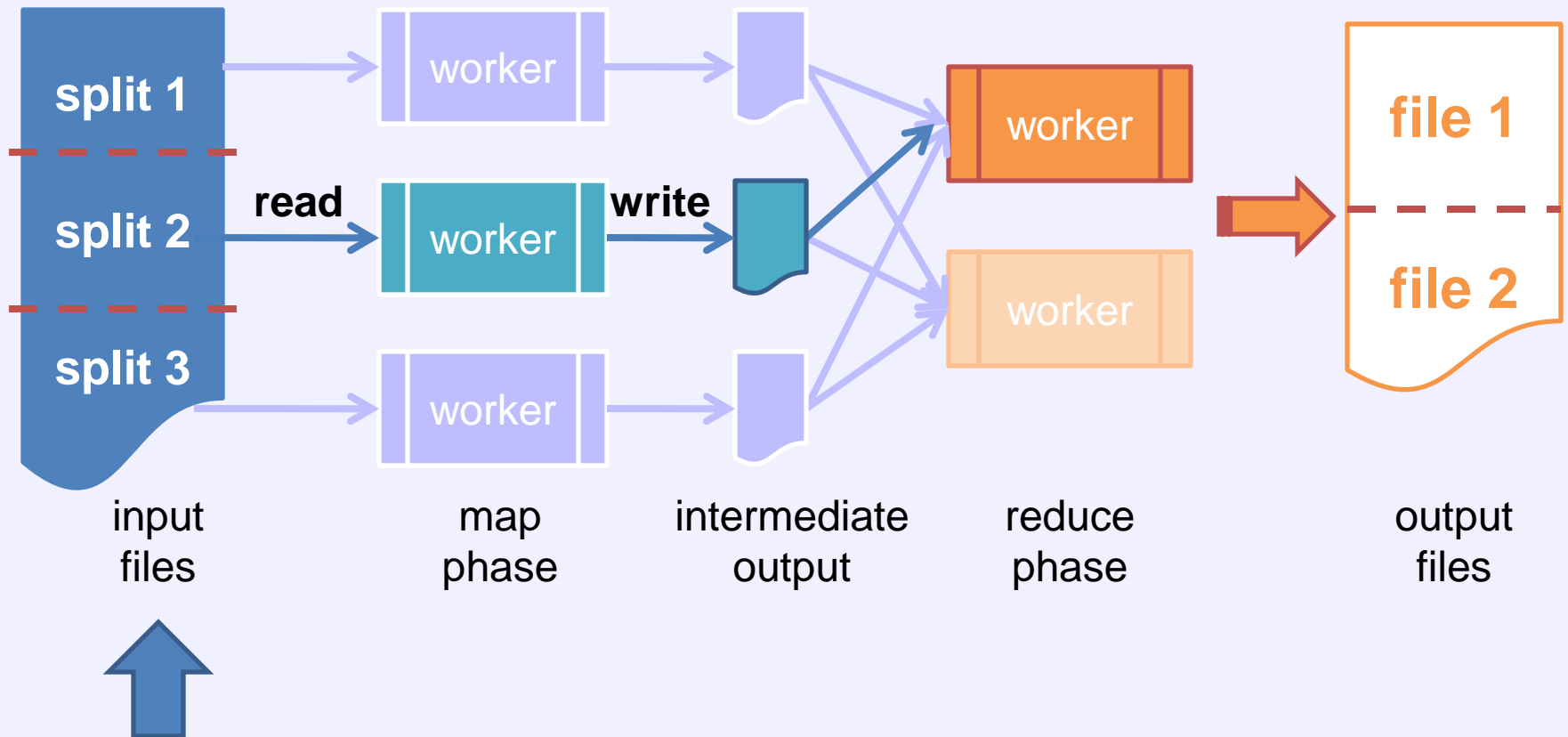
Architectural Details



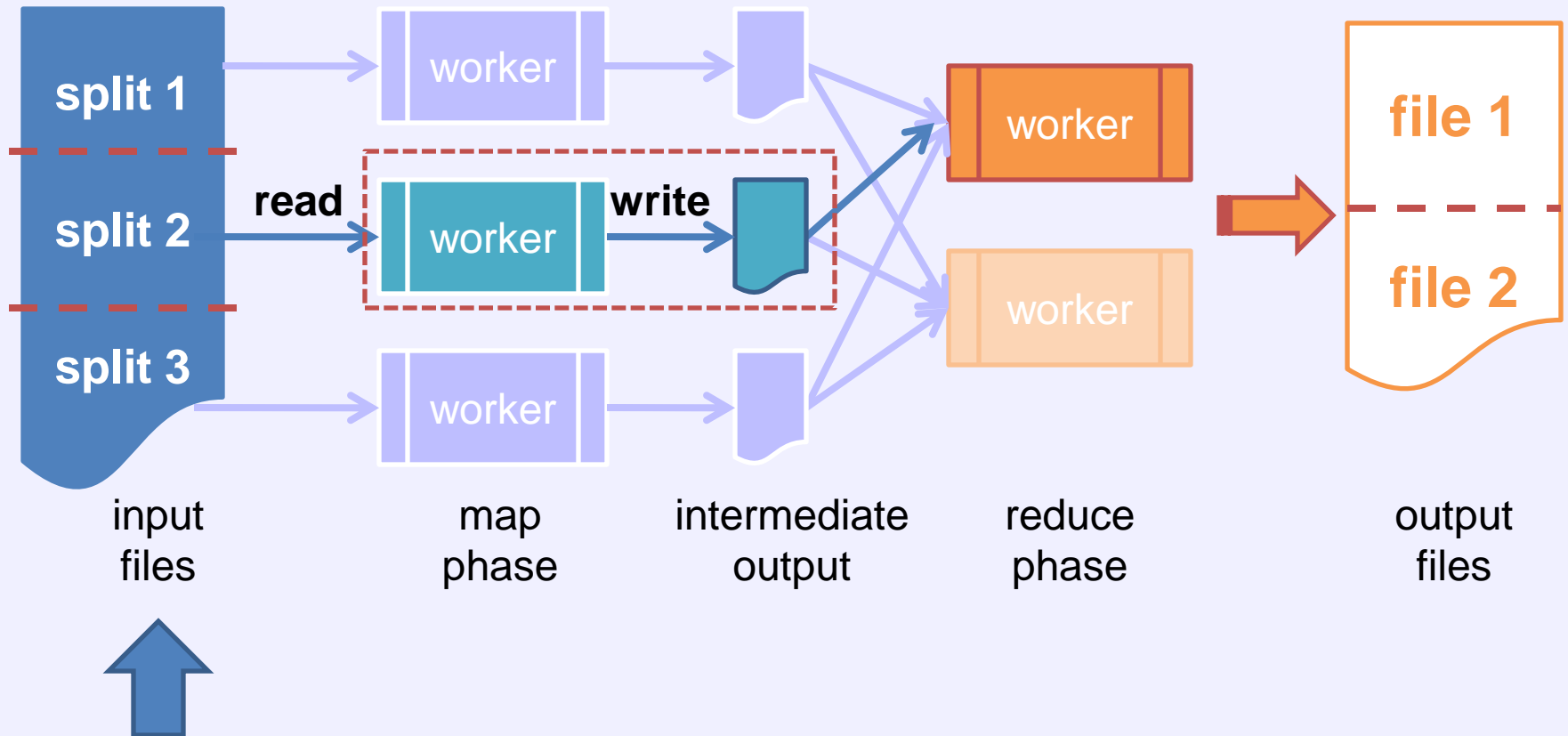
Architectural Details



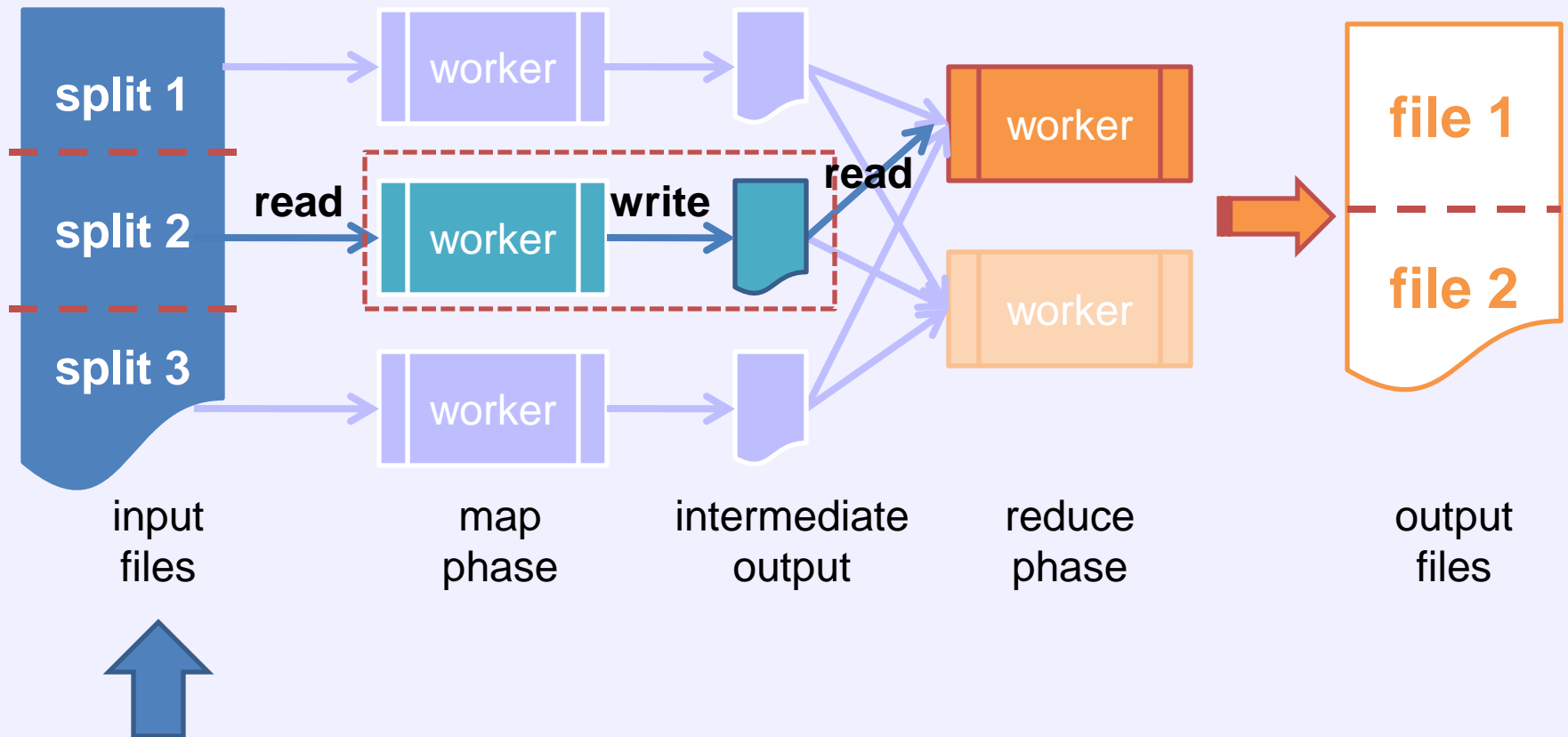
Architectural Details



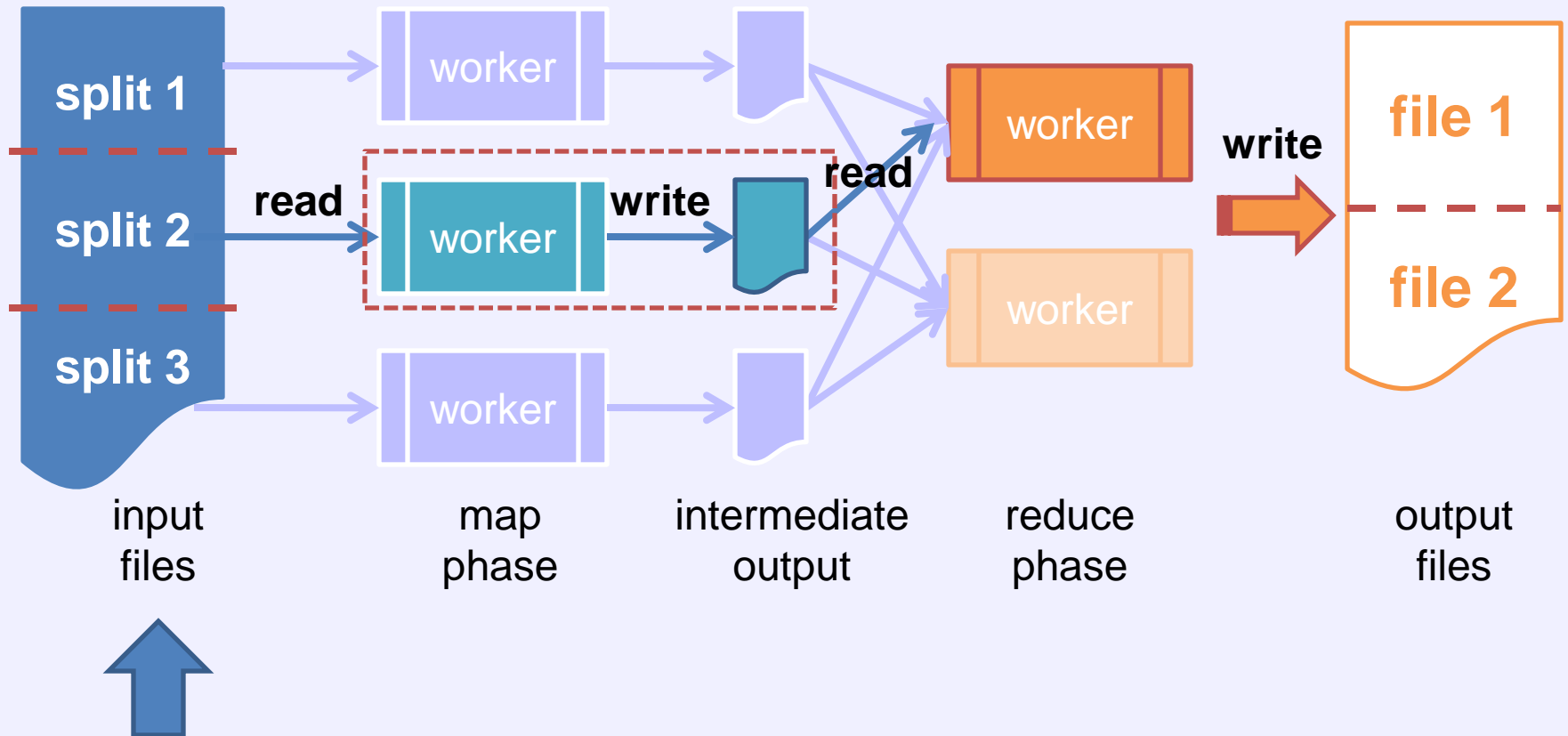
Architectural Details



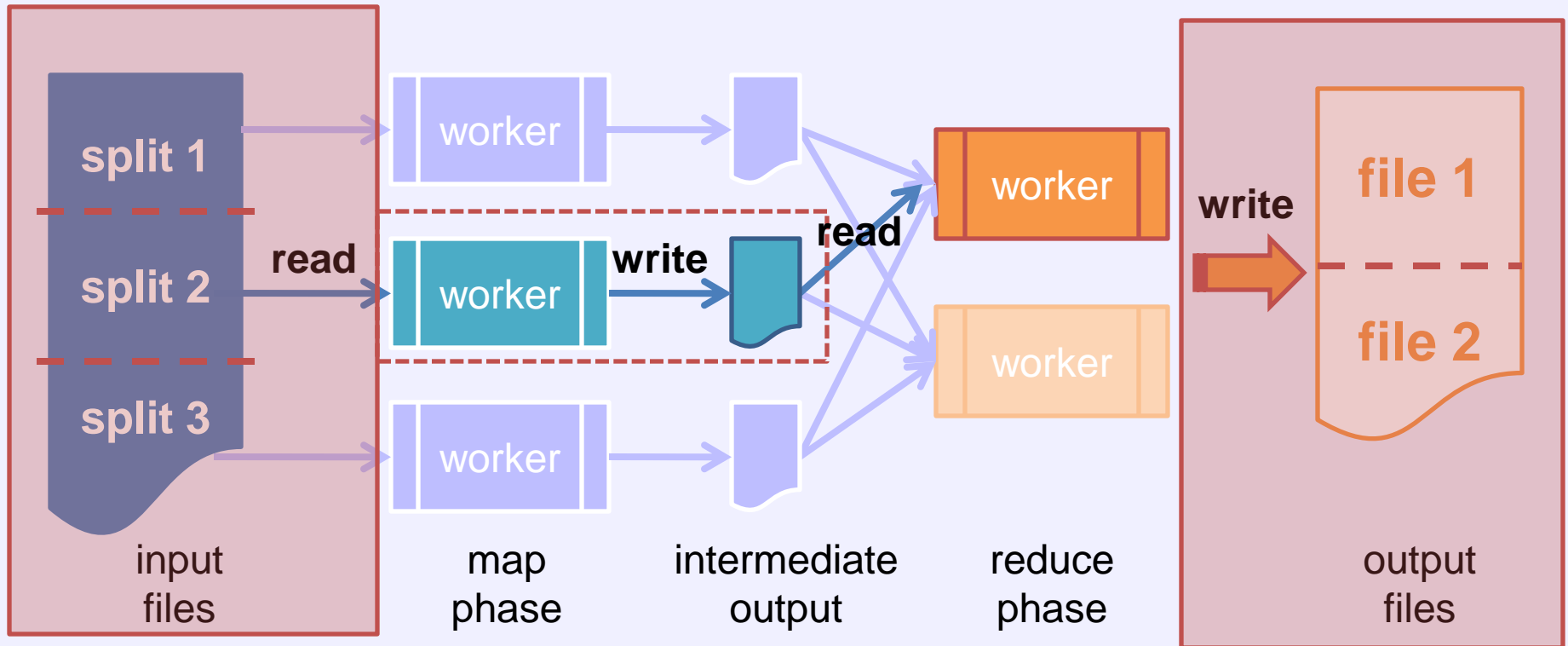
Architectural Details



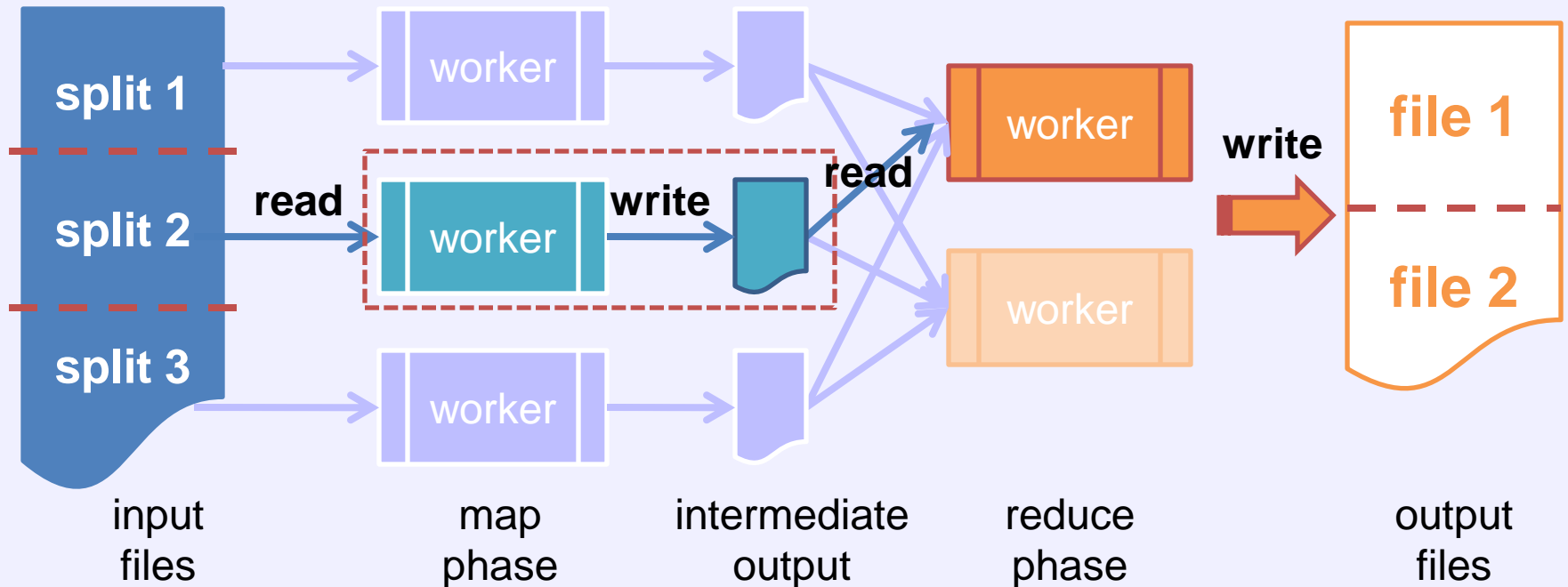
Architectural Details



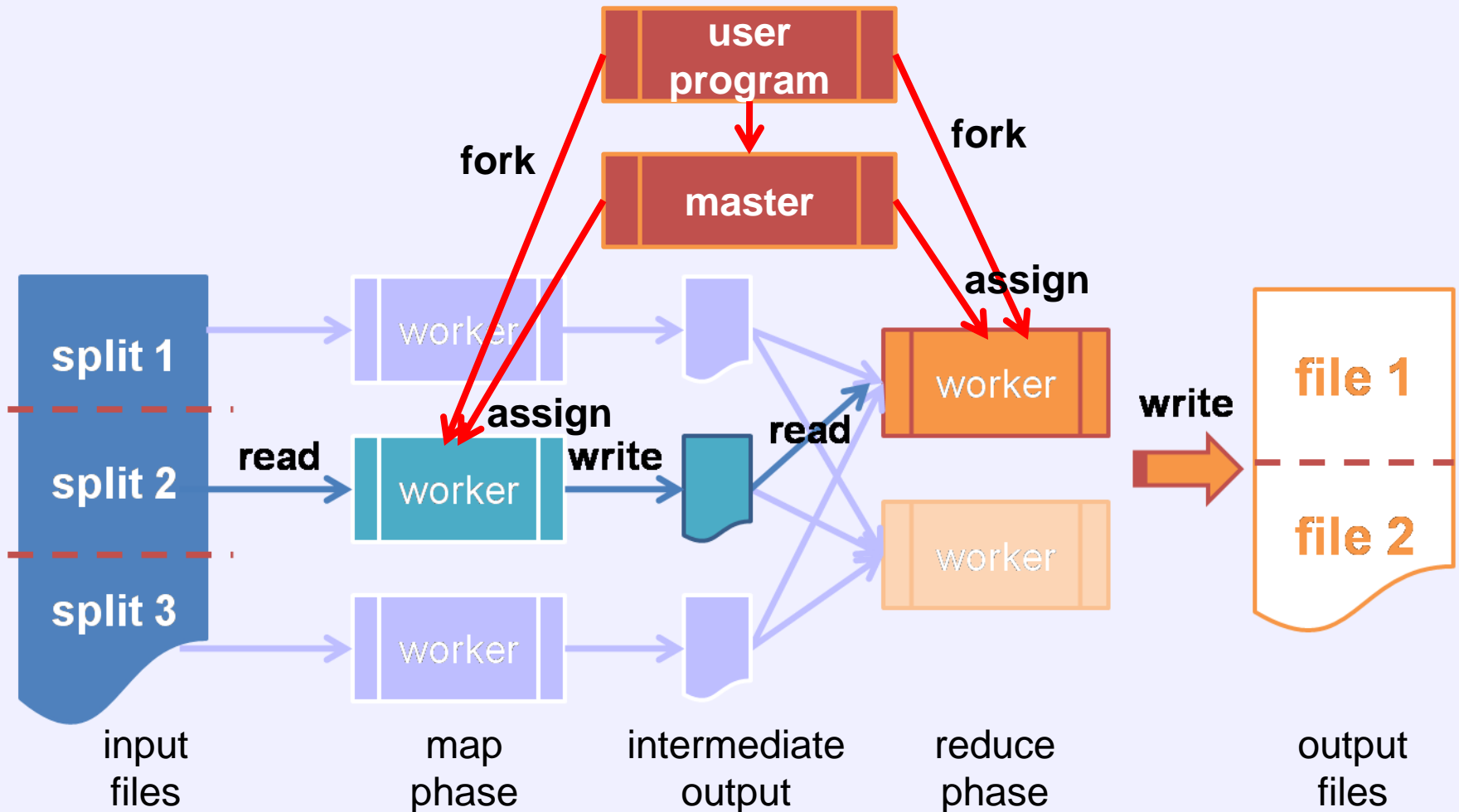
Architectural Details



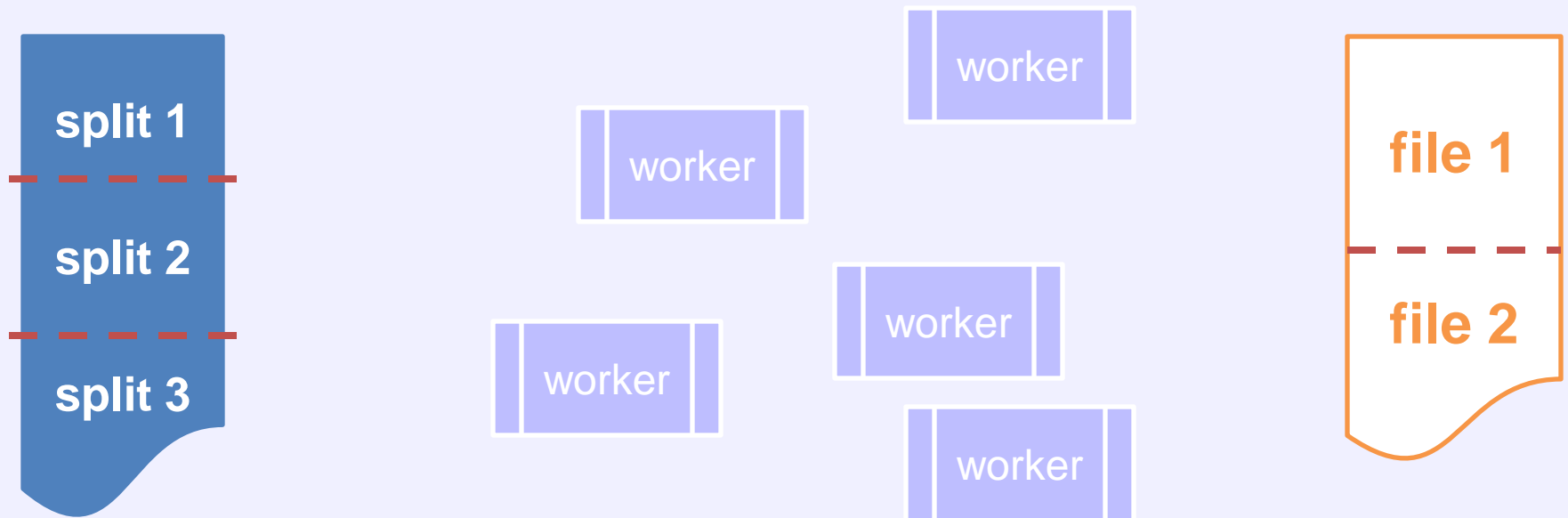
Architectural Details



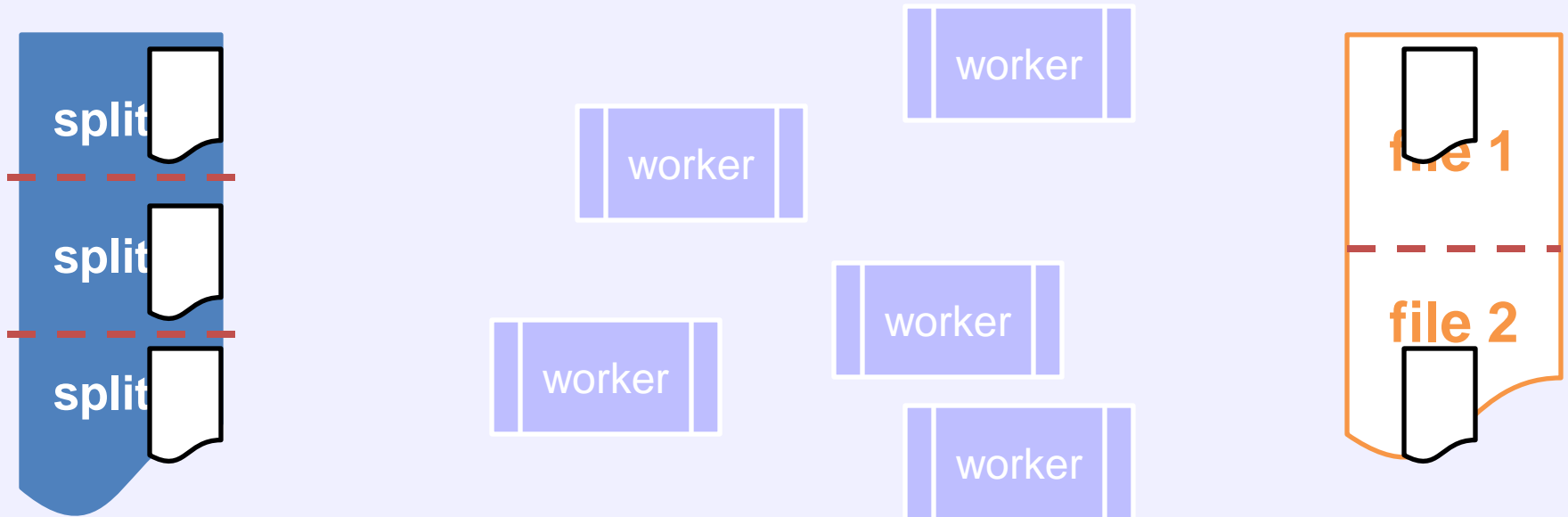
Architectural Details



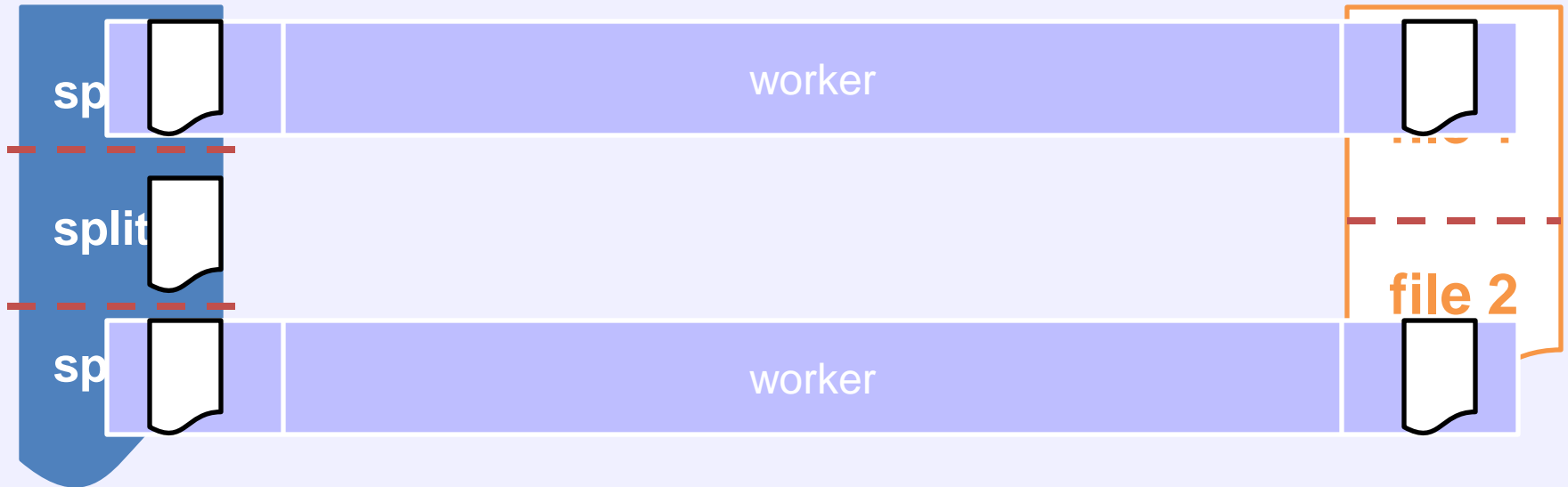
Data Flow & Locality



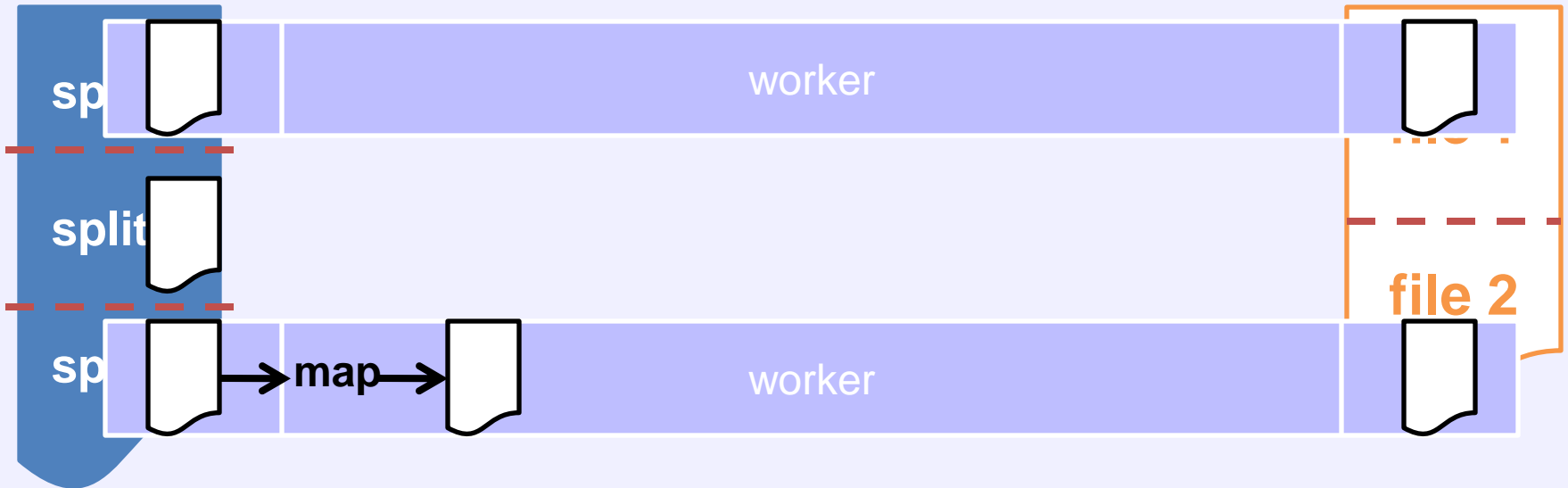
Data Flow & Locality



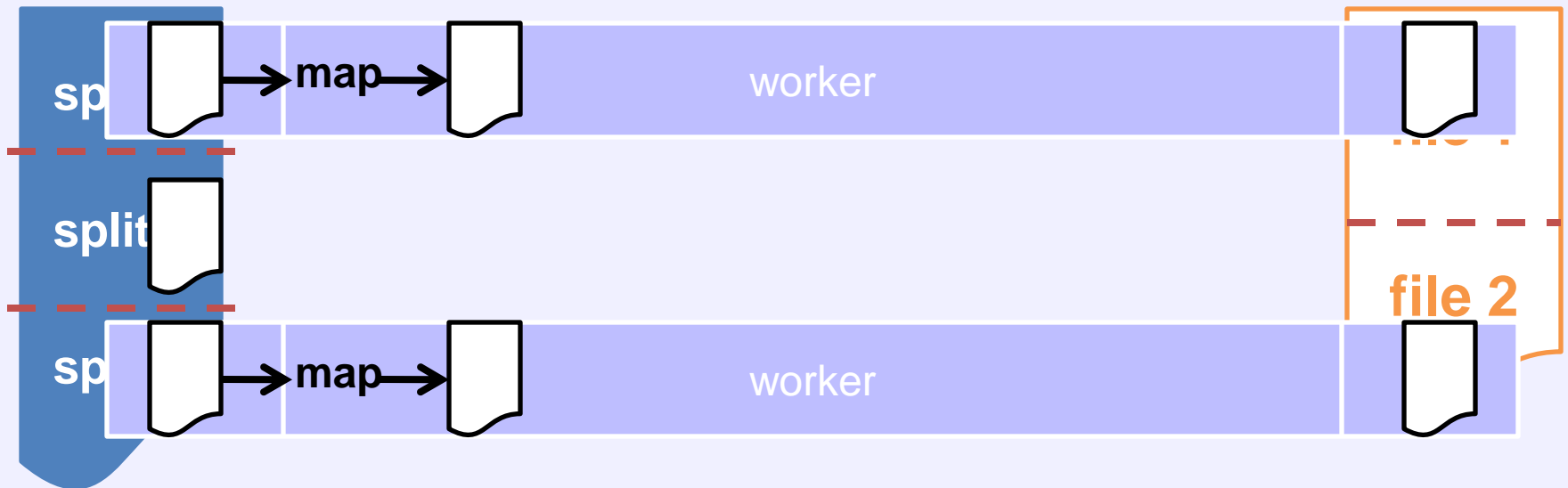
Data Flow & Locality



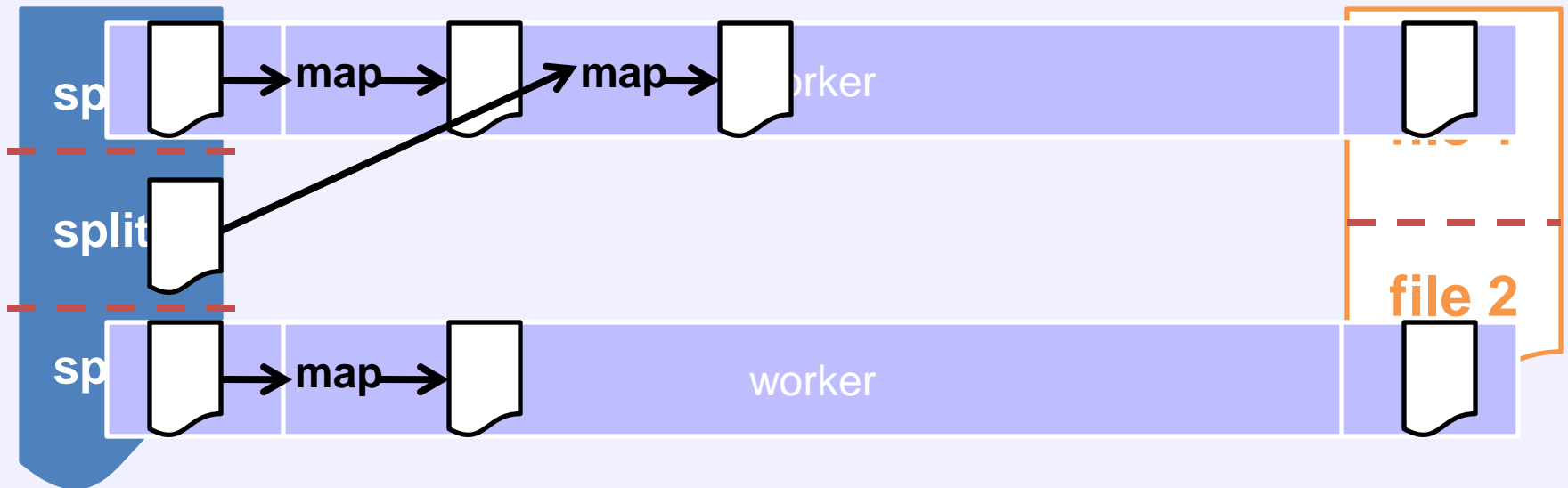
Data Flow & Locality



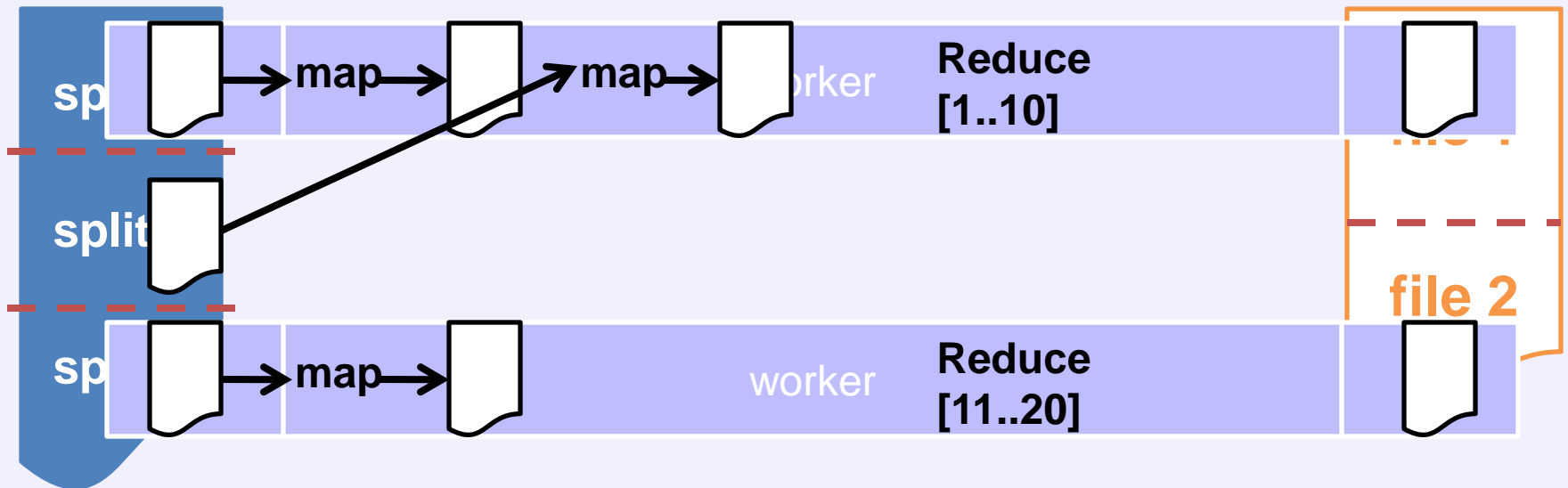
Data Flow & Locality



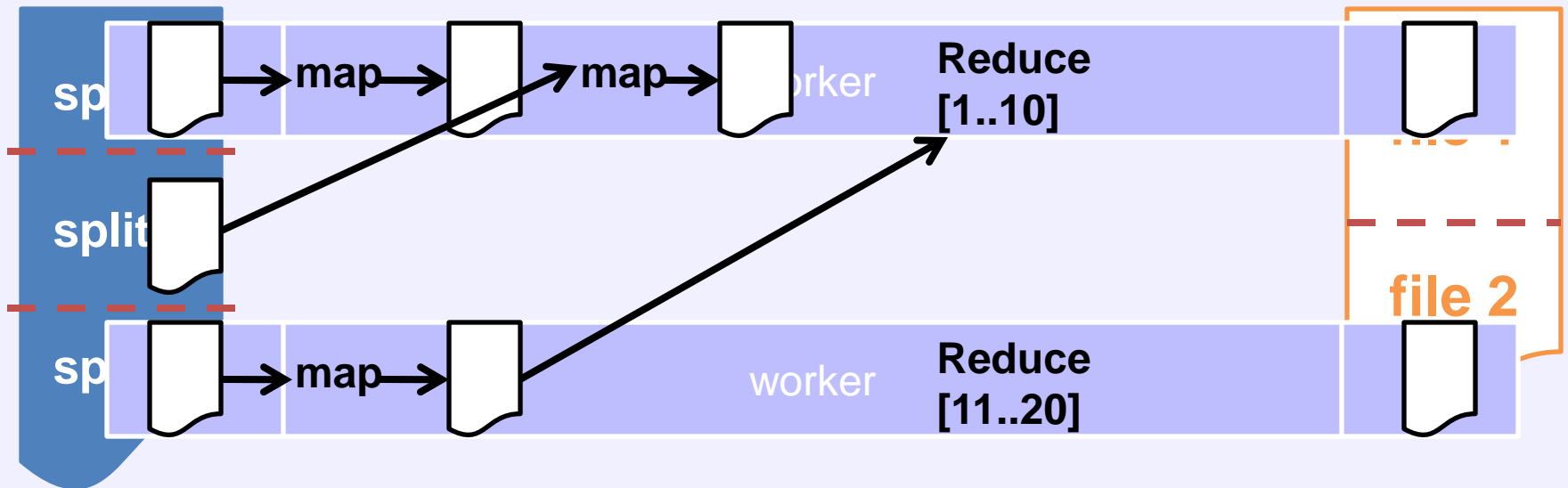
Data Flow & Locality



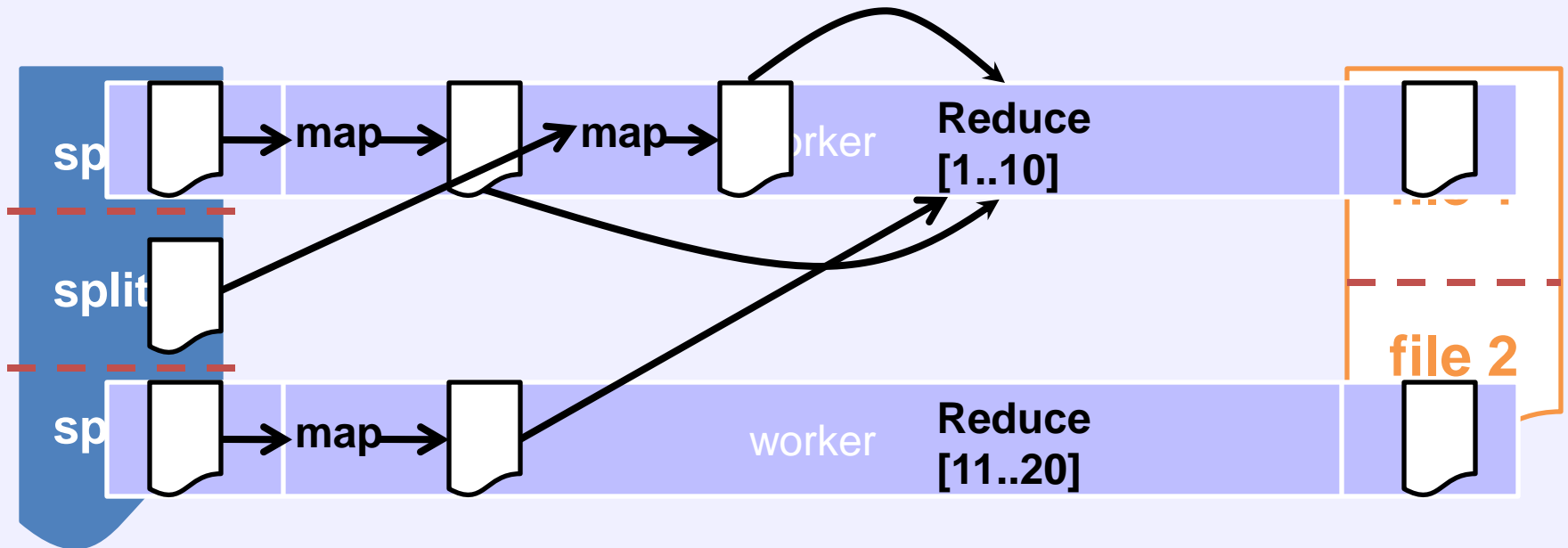
Data Flow & Locality



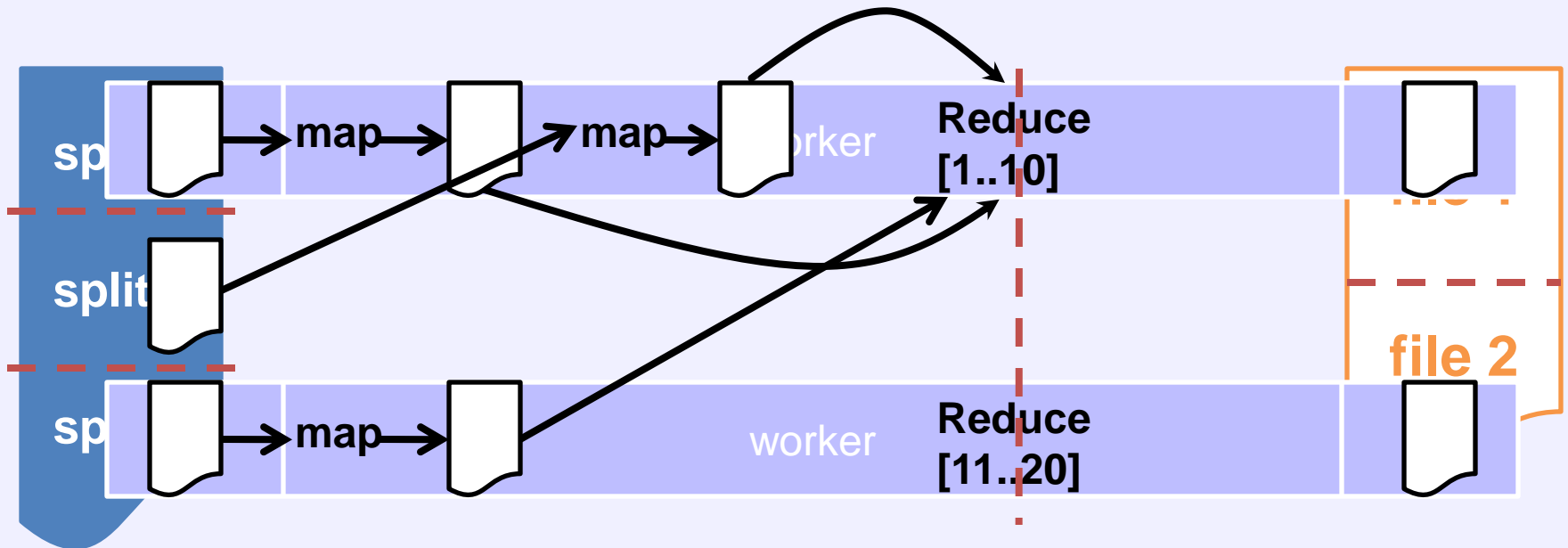
Data Flow & Locality



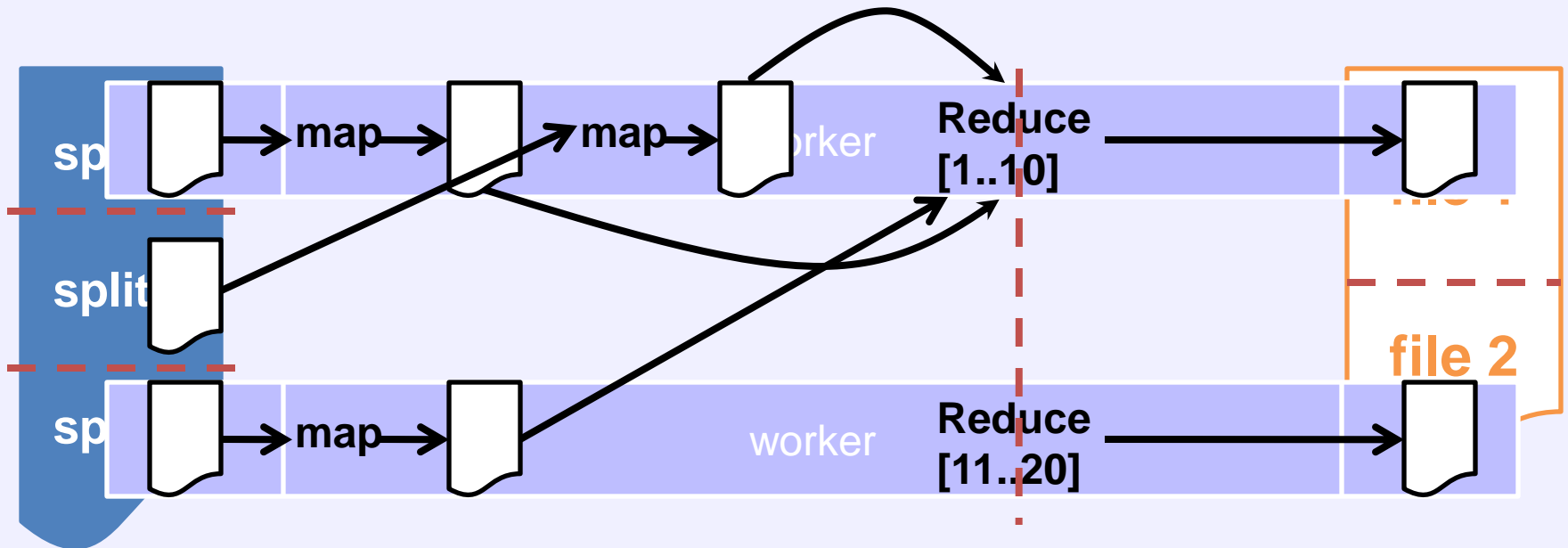
Data Flow & Locality



Data Flow & Locality



Data Flow & Locality



Combining

- Combiner instead of starting reducer early
- “Mini-reducer” in each map task
- Requires associative, cumulative reducer
- Might also reduce network traffic

Combining

- Combiner instead of starting reducer early
- “Mini-reducer” in each map task
- Requires associative, cumulative reducer
- Might also reduce network traffic

Combining

- Combiner instead of starting reducer early
- “Mini-reducer” in each map task
- Requires associative, cumulative reducer
- Might also reduce network traffic

Combining

- Combiner instead of starting reducer early
- “Mini-reducer” in each map task
- Requires associative, cumulative reducer
- Might also reduce network traffic

Combining

- Combiner instead of starting reducer early
- “Mini-reducer” in each map task
- Requires associative, cumulative reducer
- Might also reduce network traffic

➤ Early aggregation

Cluster Farming

- Balancing
 - Break job in small tasks
 - Schedule tasks as workers report idle
- Backup tasks
 - Scope with “stragglers” (slow workers)
 - “Speculative execution”

Cluster Farming

- Balancing
 - Break job in small tasks
 - Schedule tasks as workers report idle
- Backup tasks
 - Scope with “stragglers” (slow workers)
 - “Speculative execution”

Fault Tolerance

- Task failures
 - Just redo task (tasks are small)
 - Potentially on different machine
- Worker failures
 - Reallocate running tasks
 - Don't schedule on worker anymore
 - What happens with intermediate output on that worker? (potentially re-schedule all)

Fault Tolerance

- Task failures
 - Just redo task (tasks are small)
 - Potentially on different machine
- Worker failures
 - Reallocate running tasks
 - Don't schedule on worker anymore
 - What happens with intermediate output on that worker? (potentially re-schedule all)

Fault Tolerance Semantics

- Tasks are individual maps or reduces
 - Atomicity of operations
- Data level parallelism
 - Operations don't interact
- Operations supposed to be deterministic
 - Repeated executions cause same output
- Side effect freeness
 - Generally no side effects (some exceptions)

Fault Tolerance Semantics

- Tasks are individual maps or reduces
 - Atomicity of operations
- Data level parallelism
 - Operations don't interact
- Operations supposed to be deterministic
 - Repeated executions cause same output
- Side effect freeness
 - Generally no side effects (some exceptions)

Fault Tolerance Semantics

- Tasks are individual maps or reduces
 - Atomicity of operations
- Data level parallelism
 - Operations don't interact
- Operations supposed to be deterministic
 - Repeated executions cause same output
- Side effect freeness
 - Generally no side effects (some exceptions)

Fault Tolerance Semantics

- Tasks are individual maps or reduces
 - Atomicity of operations
- Data level parallelism
 - Operations don't interact
- Operations supposed to be deterministic
 - Repeated executions cause same output
- Side effect freeness
 - Generally no side effects (some exceptions)

Fault Tolerance Semantics

- Tasks are individual maps or reduces
 - Atomicity of operations
 - Data level parallelism
 - Operations don't interact
 - Operations supposed to be deterministic
 - Repeated executions cause same output
 - Side effect freeness
 - Generally no side effects (some exceptions)
- FT measures lead to same overall output

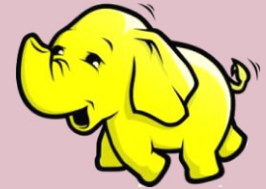
Implementations

- Google MapReduce
 - The original proposal, Google only
- Apache Hadoop
 - Open Source, used in academia
- Microsoft Dryad
 - Microsoft only, not exactly MapReduce
- Sector/Sphere^[5]
 - Research prototype, not exactly MapReduce



Implementations

- Google MapReduce
 - The original proposal, Google only
- Apache Hadoop
 - Open Source, used in academia
- Microsoft Dryad
 - Microsoft only, not exactly MapReduce
- Sector/Sphere^[5]
 - Research prototype, not exactly MapReduce



Implementations

- Google MapReduce
 - The original proposal, Google only
- Apache Hadoop
 - Open Source, used in academia
- Microsoft Dryad
 - Microsoft only, not exactly MapReduce
- Sector/Sphere^[5]
 - Research prototype, not exactly MapReduce



[5] Robert Grossman and Yunhong Gu: “Data Mining Using High Performance Data Clouds: Experimental Studies Using Sector and Sphere” in *KDD 2008*

Implementations

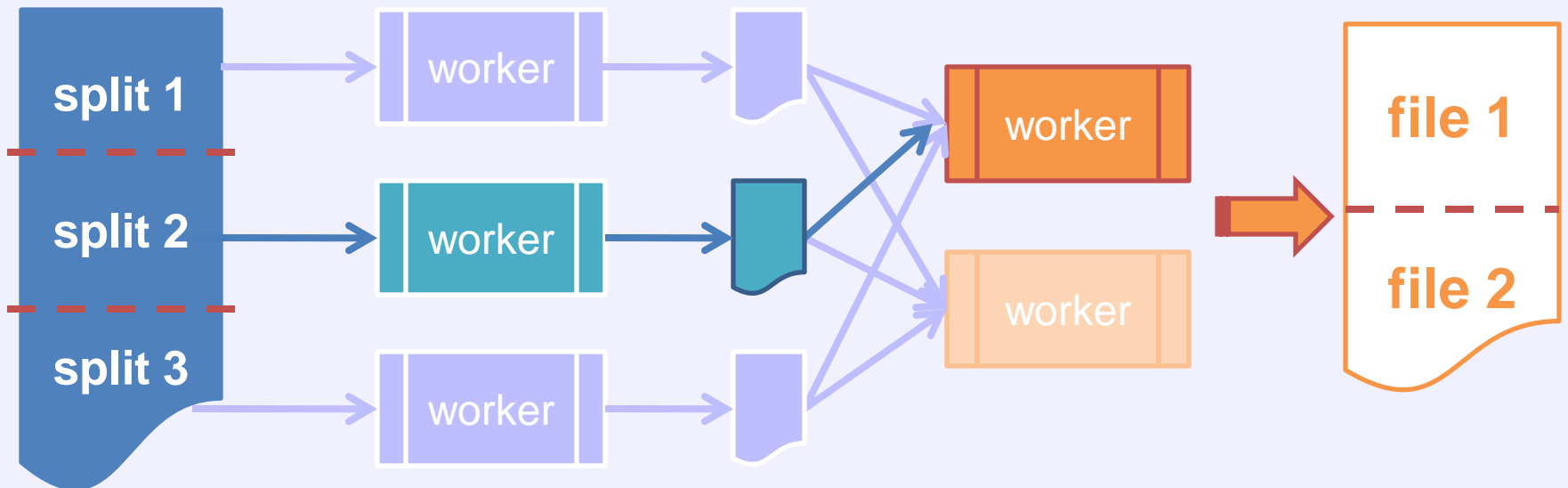
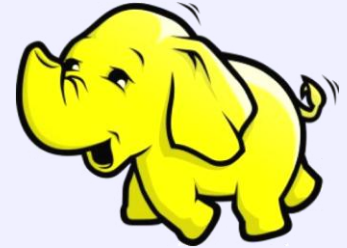
- Google MapReduce
 - The original proposal, Google only
- Apache Hadoop
 - Open Source, used in academia
- Microsoft Dryad
 - Microsoft only, not exactly MapReduce
- Sector/Sphere^[5]
 - Research prototype, not exactly MapReduce

[5] Robert Grossman and Yunhong Gu: “Data Mining Using High Performance Data Clouds: Experimental Studies Using Sector and Sphere” in *KDD 2008*

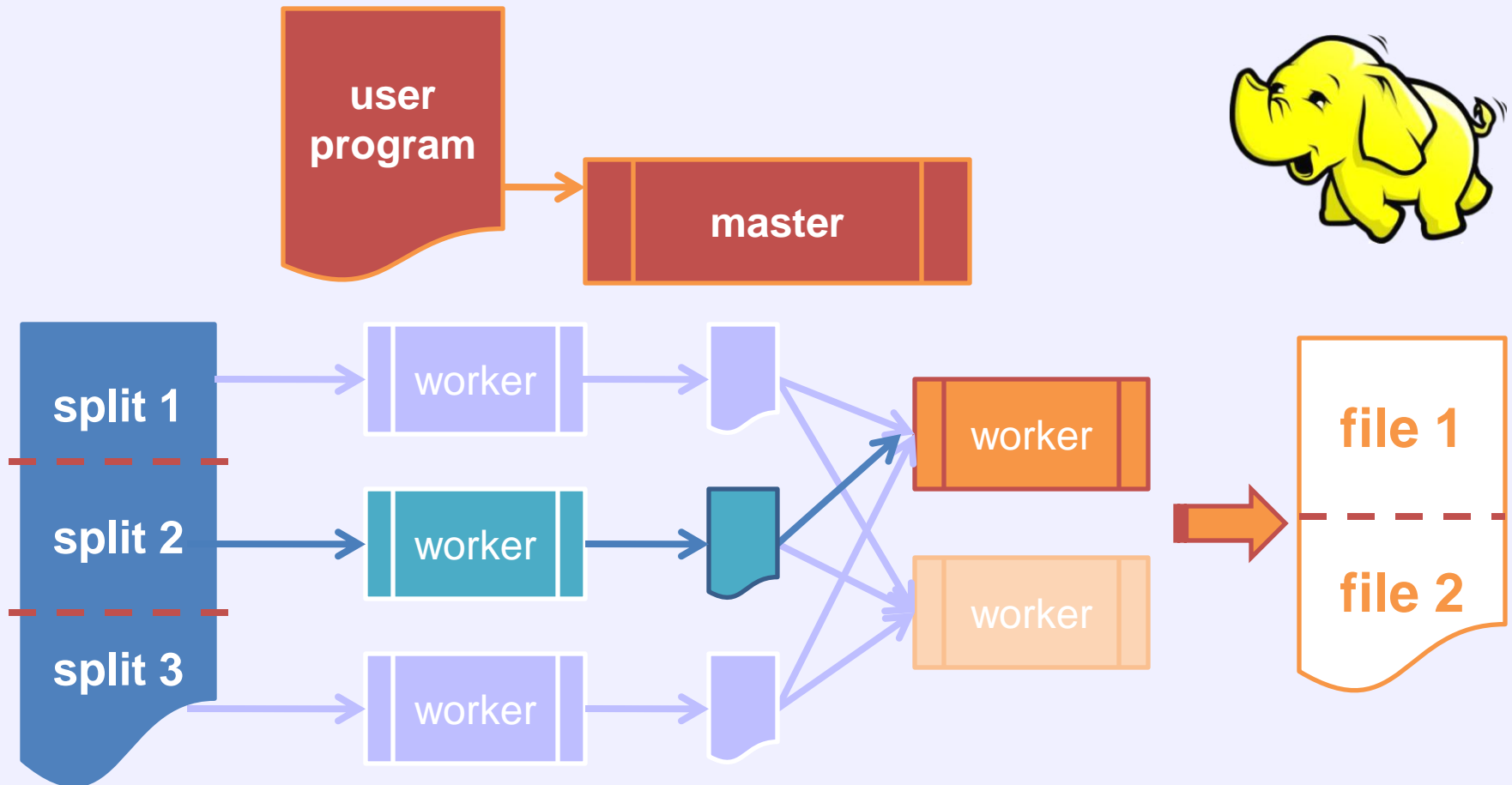
Implementations

	Google MR	Hadoop	Dryad	Sector
Published	2004	(2004-2008)	2007	(2008)
Availability	Proprietary	Open Source	Proprietary	Open Source
Used by	Google	Research, Yahoo!, Facebook, Amazon (EC2!)	Microsoft	Research
Implemented	C++	Java	C++	C++
Designed for	Data center	Data center	Data center	Several data centers

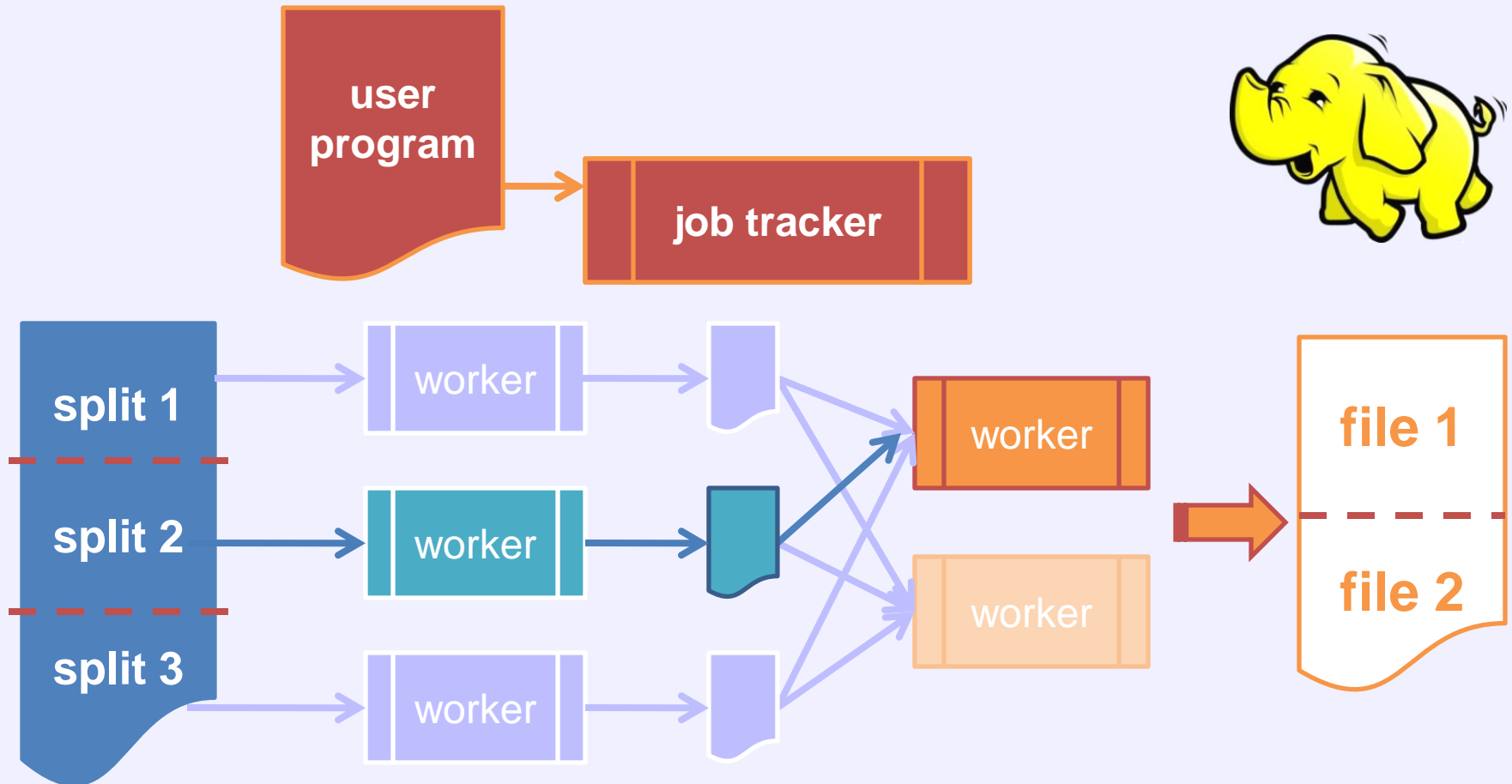
Hadoop Terminology



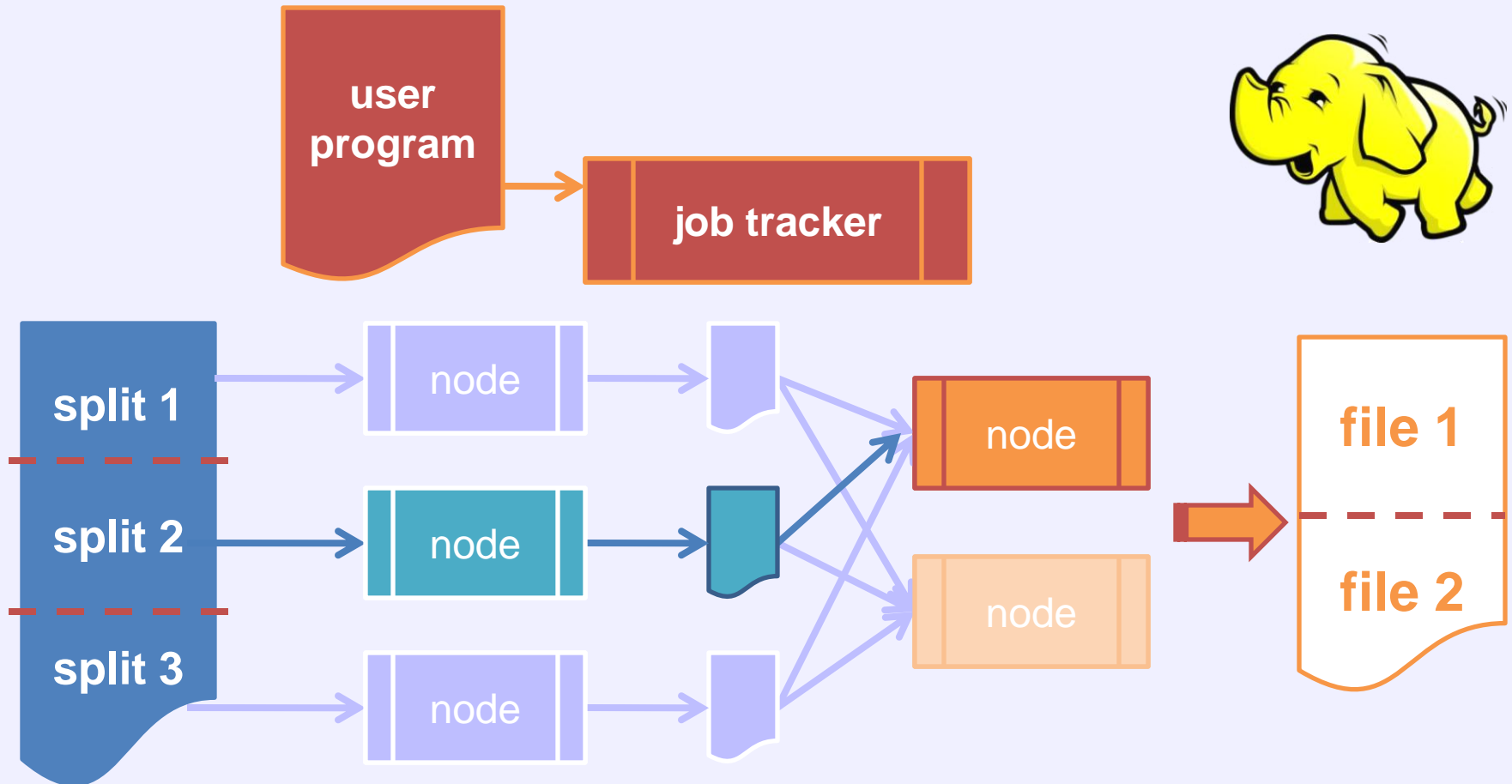
Hadoop Terminology



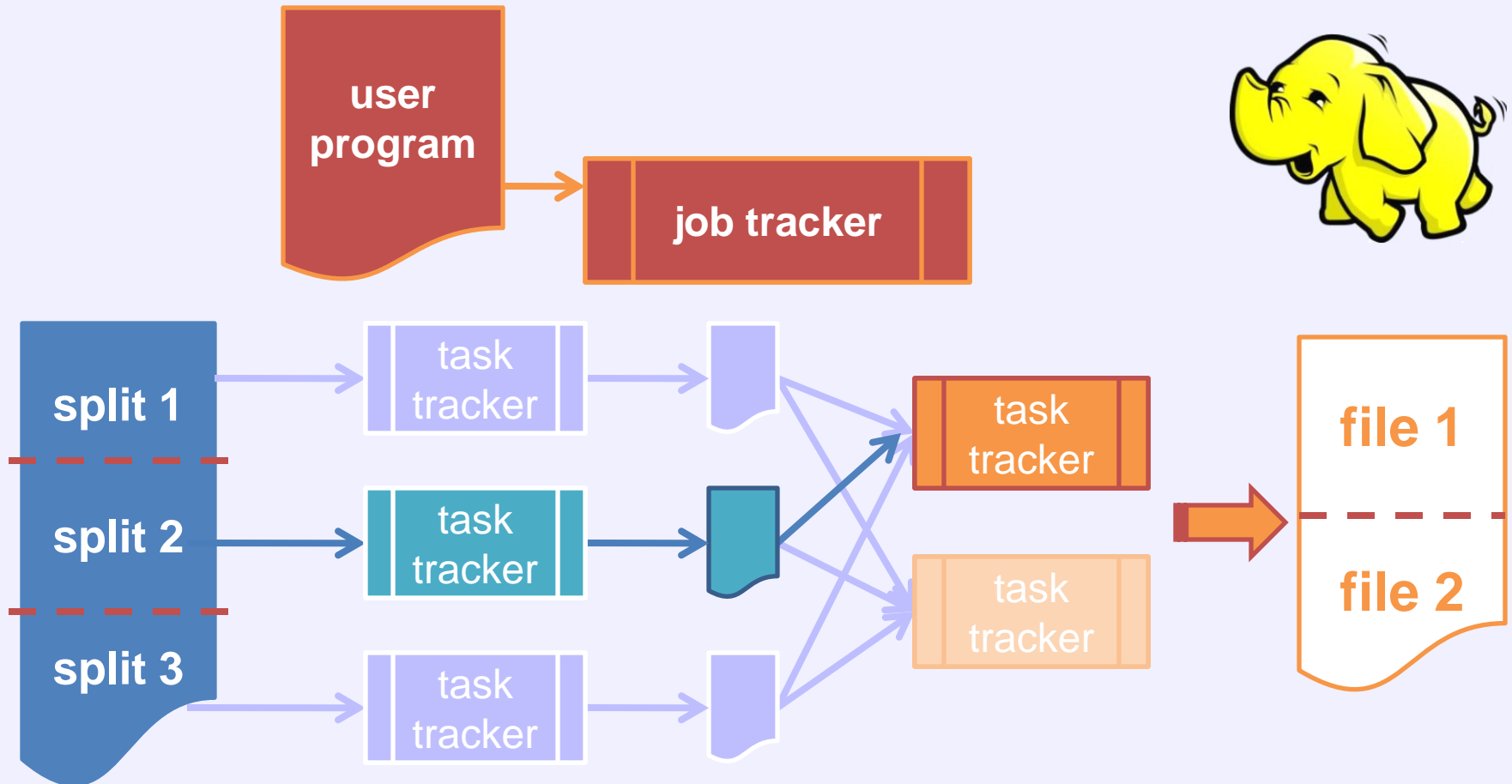
Hadoop Terminology



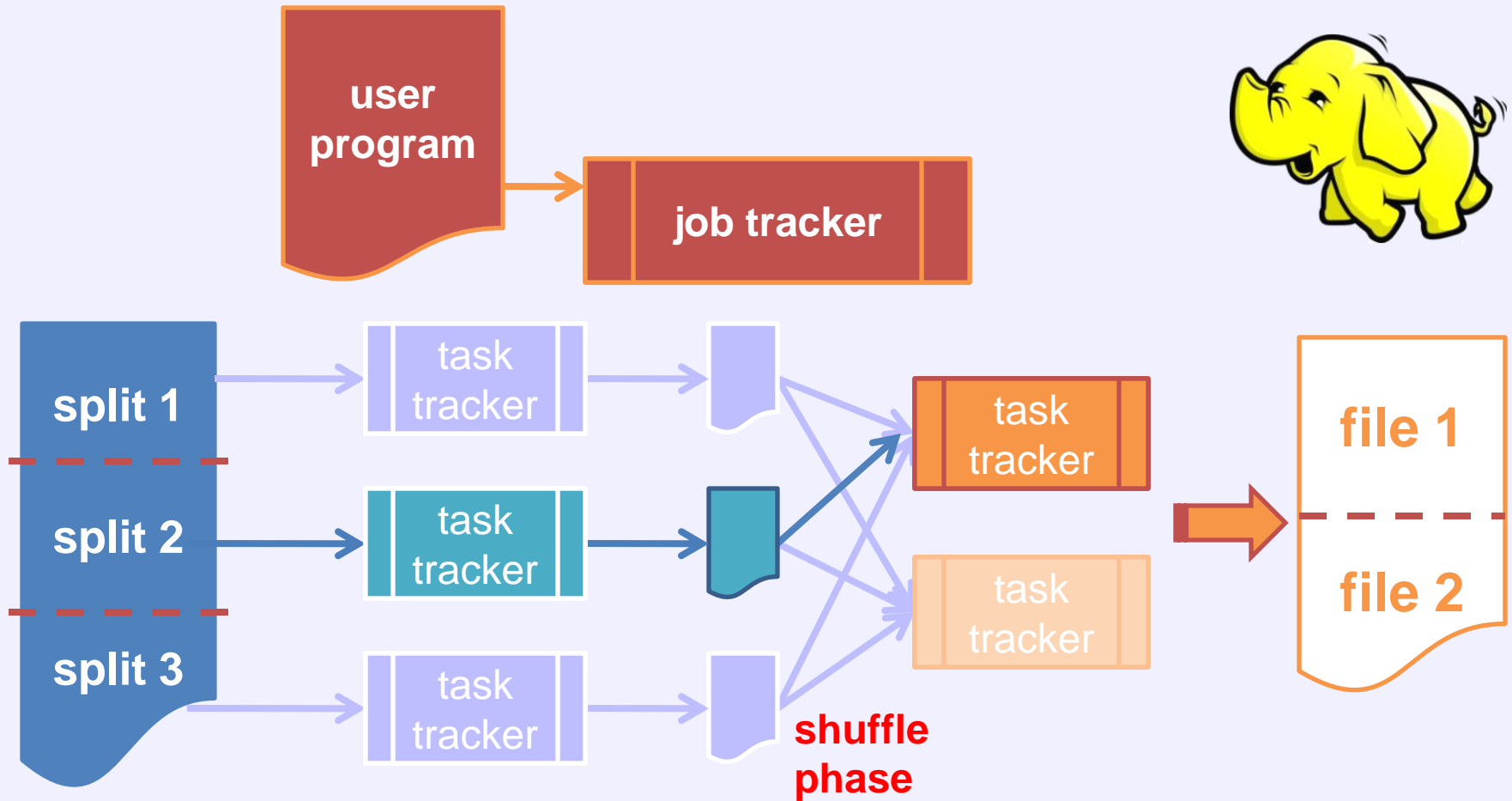
Hadoop Terminology



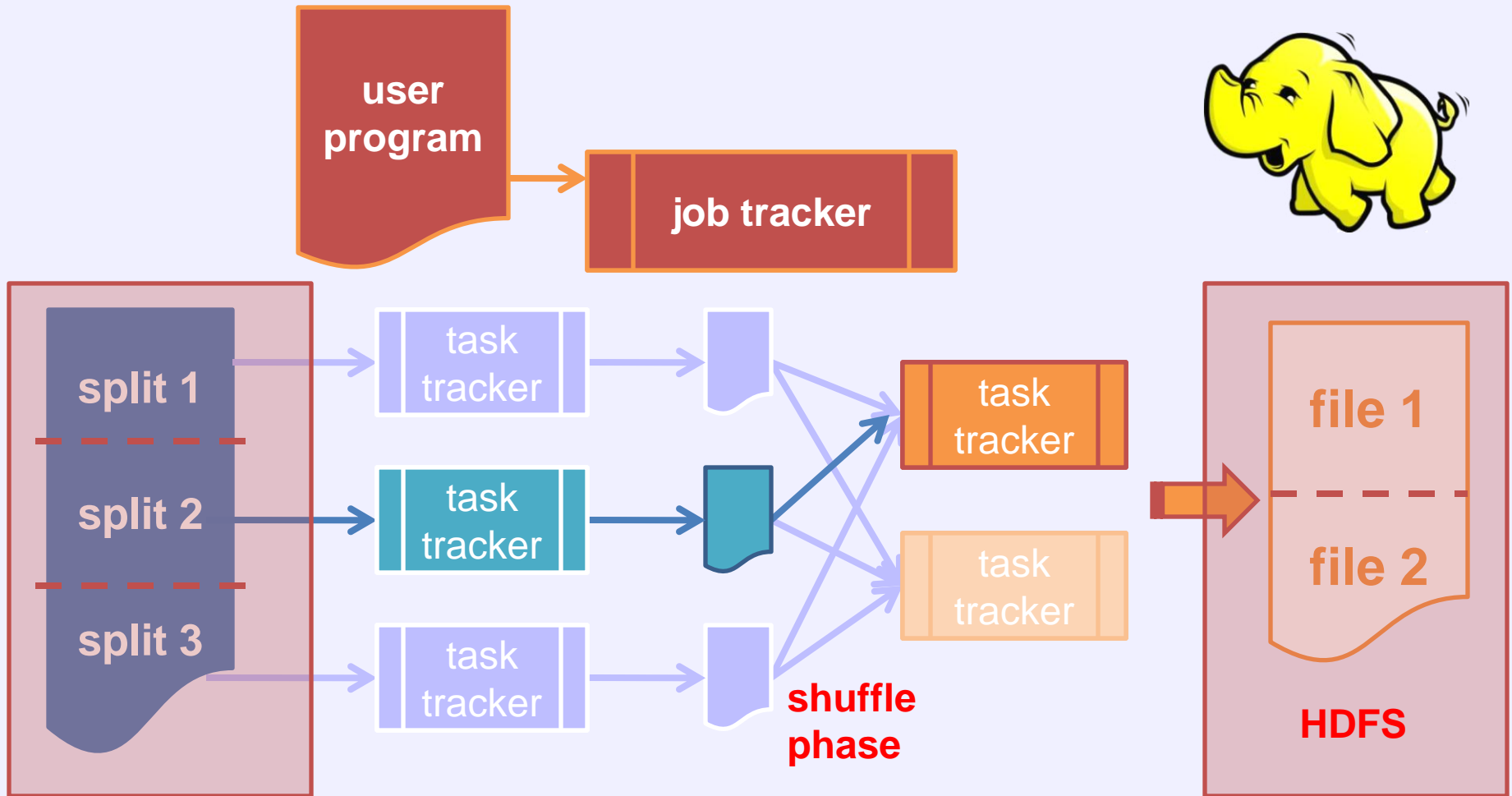
Hadoop Terminology



Hadoop Terminology



Hadoop Terminology



MapReduce in Use

- Facebook
 - 600 nodes cluster for warehouse
 - 2 PB, growing by 15 TB per day
 - Daily analyses, concurrent ad-hoc queries
- Google
 - Aug 2004: ~ 30,000 jobs, 217 machine years
 - Sep 2007: ~ 2 million jobs, 11,081 machine years
- Yahoo!
 - Using for web services
 - Won TeraSort contest in 2008 with Hadoop cluster
- Amazon
 - MapReduce on EC2

MapReduce in Use

- Facebook
 - 600 nodes cluster for warehouse
 - 2 PB, growing by 15 TB per day
 - Daily analyses, concurrent ad-hoc queries
- Google
 - Aug 2004: ~ 30,000 jobs, 217 machine years
 - Sep 2007: ~ 2 million jobs, 11,081 machine years
- Yahoo!
 - Using for web services
 - Won TeraSort contest in 2008 with Hadoop cluster
- Amazon
 - MapReduce on EC2

MapReduce in Use

- Facebook
 - 600 nodes cluster for warehouse
 - 2 PB, growing by 15 TB per day
 - Daily analyses, concurrent ad-hoc queries
- Google
 - Aug 2004: ~ 30,000 jobs, 217 machine years
 - Sep 2007: ~ 2 million jobs, 11,081 machine years
- Yahoo!
 - Using for web services
 - Won TeraSort contest in 2008 with Hadoop cluster
- Amazon
 - MapReduce on EC2

MapReduce in Use

- Facebook
 - 600 nodes cluster for warehouse
 - 2 PB, growing by 15 TB per day
 - Daily analyses, concurrent ad-hoc queries
- Google
 - Aug 2004: ~ 30,000 jobs, 217 machine years
 - Sep 2007: ~ 2 million jobs, 11,081 machine years
- Yahoo!
 - Using for web services
 - Won TeraSort contest in 2008 with Hadoop cluster
- Amazon
 - MapReduce on EC2

Outline

- MapReduce – Back to its Cradle
- What MapReduce is and What it's Not
- The MapReduce Framework(s)
- Strengths and Weaknesses
- Summary

Clever Recombination

- map & reduce from functional programming
- Applied for distributed systems
- Simple, intuitive interface

Clever Recombination

- map & reduce from functional programming
 - Applied for distributed systems
 - Simple, intuitive interface
- Highly useful system for large-scale data processing needs

Impact

- Inspired a lot of scientific publications
 - Extending the model or framework
 - Trying to combine with other techniques
- Impact on Industry
 - Solves actual problems
 - Used by many companies

Impact

- Inspired a lot of scientific publications
 - Extending the model or framework
 - Trying to combine with other techniques
- Impact on Industry
 - Solves actual problems
 - Used by many companies

Not Really New

- Programming model is not new
 - Functional programming
- Distributed systems are not novel
 - Well...
- Fault tolerance, balancing, etc.
 - Studied in various fields (especially DS)

Not Really New

- Programming model is not new
 - Functional programming
- Distributed systems are not novel
 - Well...
- Fault tolerance, balancing, etc.
 - Studied in various fields (especially DS)

Not Really New

- Programming model is not new
 - Functional programming
- Distributed systems are not novel
 - Well...
- Fault tolerance, balancing, etc.
 - Studied in various fields (especially DS)

Not Really New

- Programming model is not new
 - Functional programming
 - Distributed systems are not novel
 - Well...
 - Fault tolerance, balancing, etc.
 - Studied in various fields (especially DS)
- Could be considered pure engineering

Summary

- It is hard to process very large datasets
 - Even harder with non-homogeneous data
 - Need massive parallelism
 - Hard to implement case-by-case
 - MapReduce: parallelization framework
 - Uses FP concepts
 - Simple and elegant solution
 - Huge impact

Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact

Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact

Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact

Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact

Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact

Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact

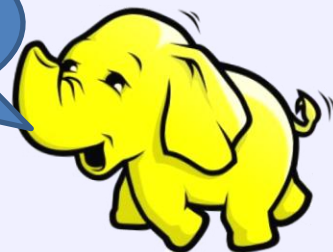
Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact

Summary

- It is hard to process very large datasets
- Even harder with non-homogeneous data
- Need massive parallelism
- Hard to implement case-by-case
- MapReduce: parallelization framework
- Uses FP concepts
- Simple and elegant solution
- Huge impact

Thank you!
Questions?



Google MR Usage Numbers

	Aug 2004	Mar 2006	Sep 2007
Number of jobs	29,000	171,000	2,217,000
Avg. runtime [sec]	634	874	395
Total machine years	217	2,002	11,081
Map input [TB]	3,288	52,254	403,152
Intermediate (map) output [TB]	758	6,743	34,774
Final (reduce) output [TB]	193	2,970	14,018
Machines per job [avg]	157	268	394
Unique mappers	395	1958	4083
Unique reducers	269	1208	2418