CompSci 401: Cloud Computing

Software for the Cloud

Prof. Ítalo Cunha

昆山杜克大学
DUKE KUNSHAN UNIVERSITY

How to build applications for the cloud?

- Software design and architecture
- Distributed and parallel algorithms
- New programming languages and libraries

How to build applications for the cloud?

- Software design and architecture
- Distributed and parallel algorithms
- New programming languages and libraries

- Application development frameworks

Cloud-native applications

- Specifically designed for the cloud
- What can we do better?
  - Improve security
  - Reduce software flaws
  - Enhance performance
  - Increase scalability
  - Improve reliability

CompSci 401: Cloud Computing

Parallel Computation

Prof. Ítalo Cunha

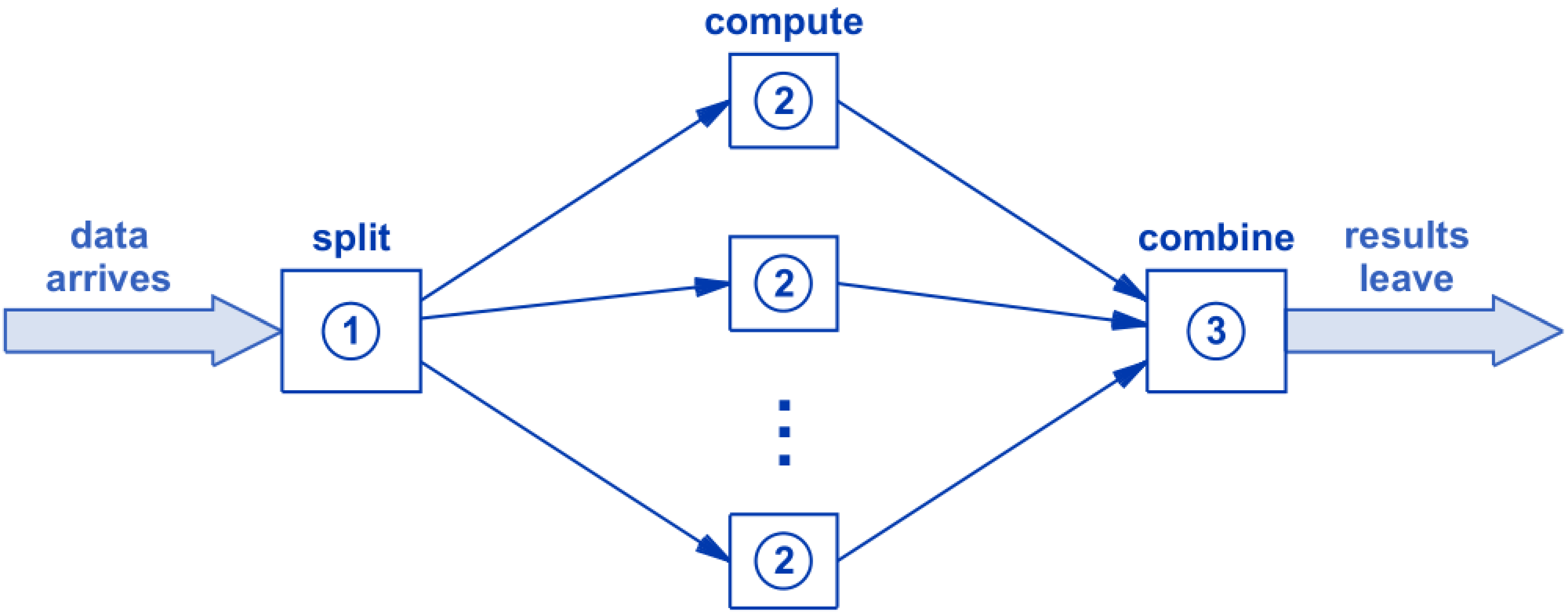
昆山杜克大学
DUKE KUNSHAN UNIVERSITY

Datacenters have large amounts of resources

- Thousands of cores
- Tebibytes of RAM
- Petabytes of storage
- Need mechanisms to effectively use these resources
    - Efficient use of resources (low overhead)
    - Programmer efficiency (low time to develop applications)

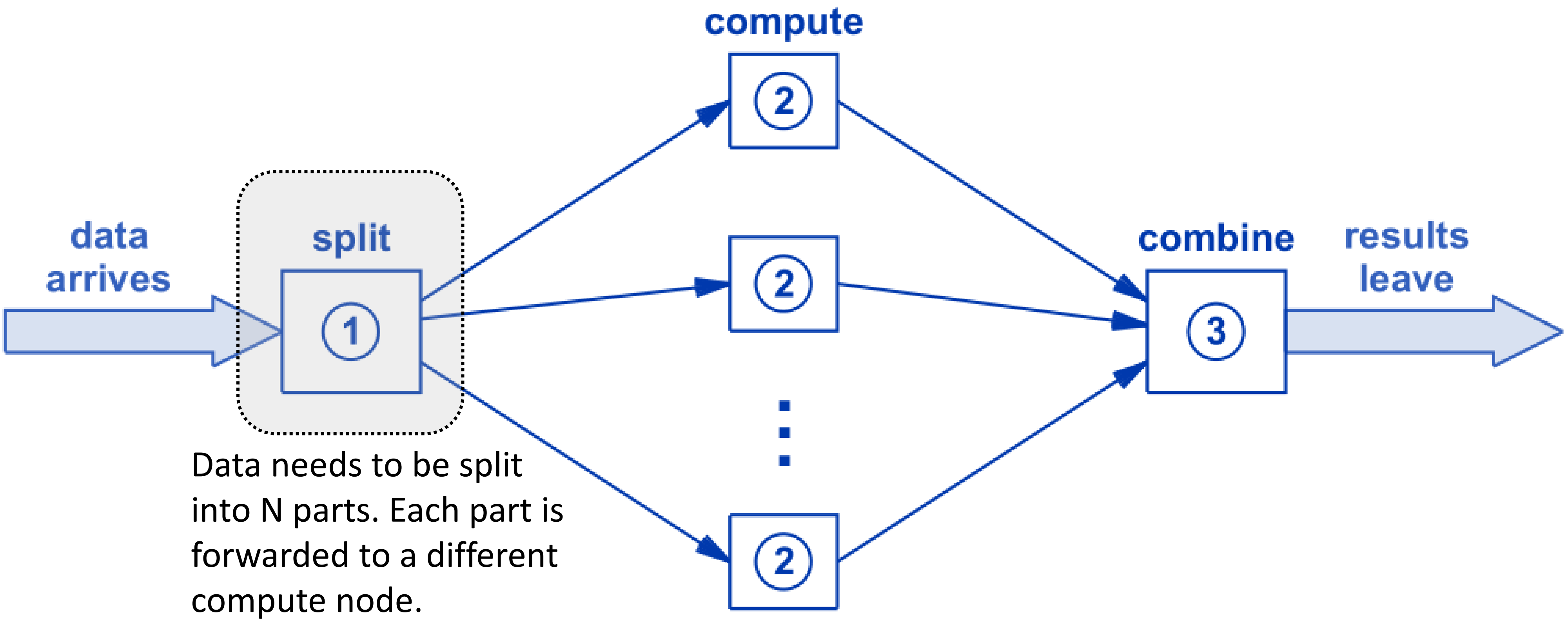
Data parallelism

- Common, widely-applicable approach



Data parallelism

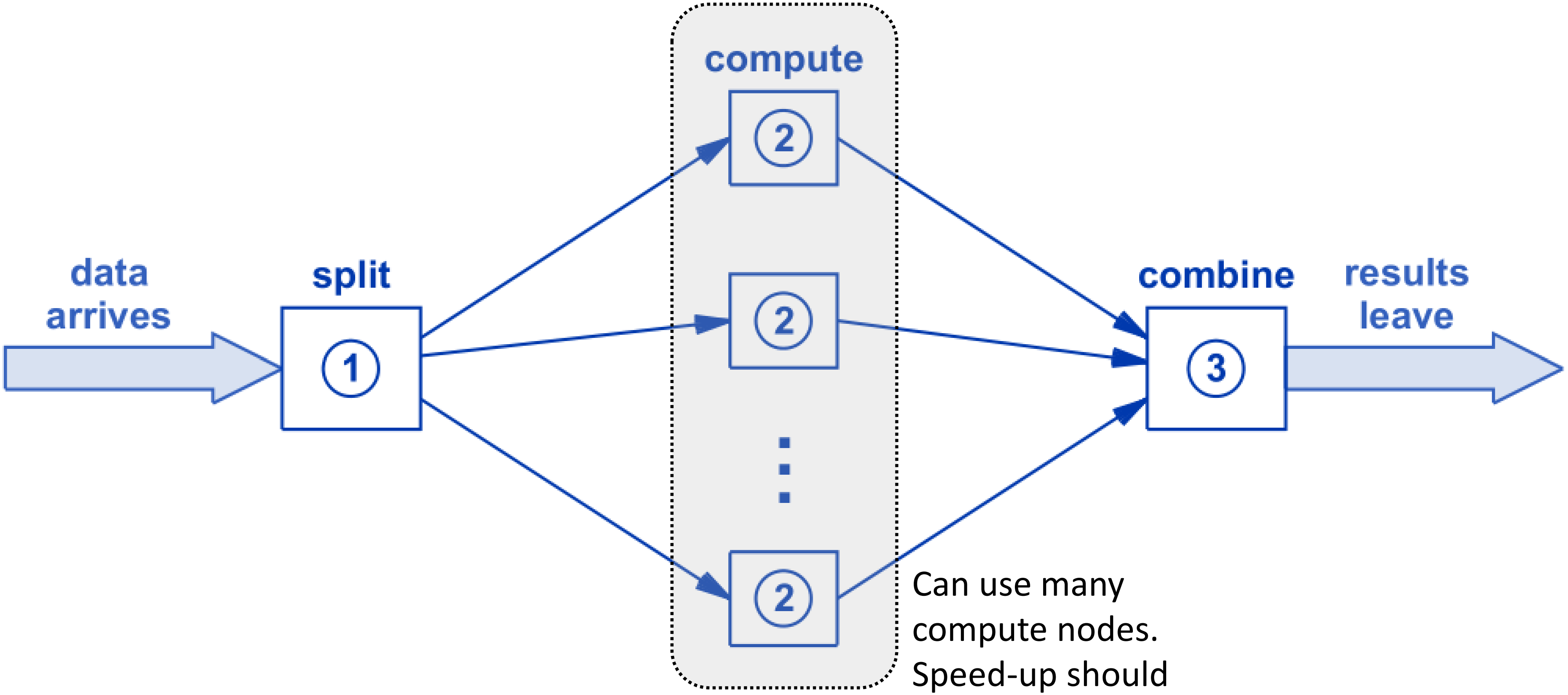
- Common, widely-applicable approach



Data needs to be split into N parts. Each part is forwarded to a different compute node.

Data parallelism

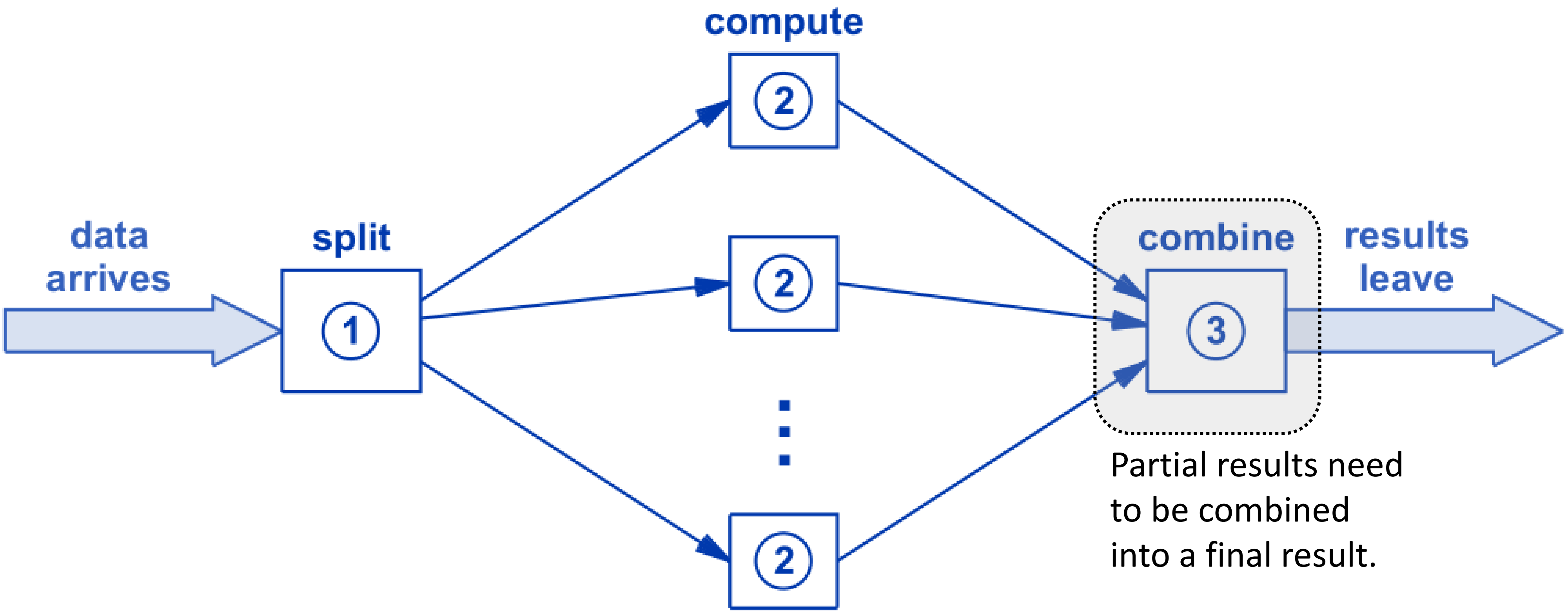
- Common, widely-applicable approach



Can use many
compute nodes.
Speed-up should
be proportional.

Data parallelism

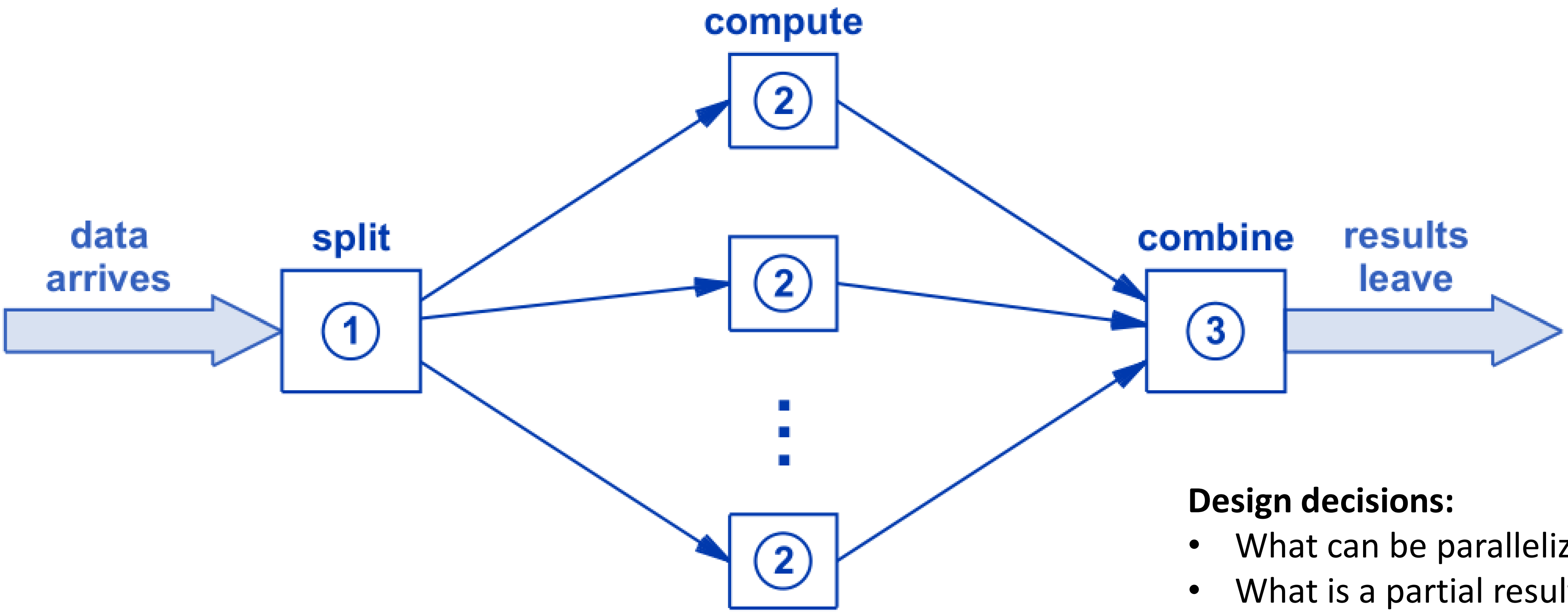
- Common, widely-applicable approach



Partial results need
to be combined
into a final result.

Data parallelism

- Common, widely-applicable approach



Design decisions:

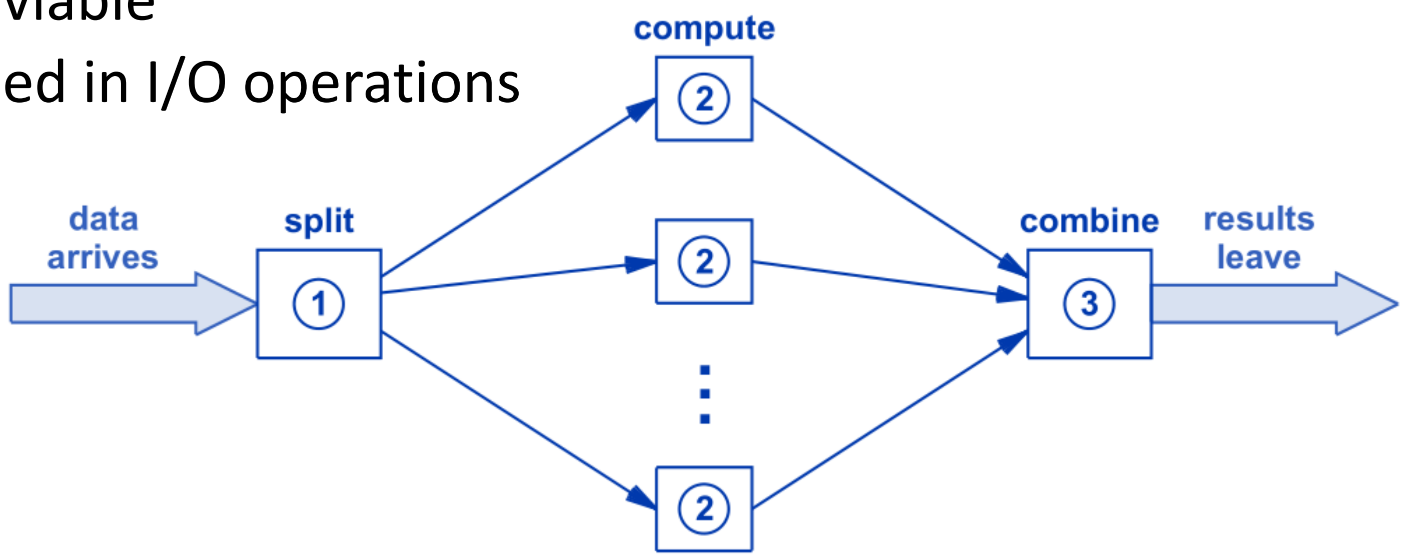
- What can be parallelized?
- What is a partial result?
- How to combine?

Limitations of data parallelism

- Some problems do not allow for data parallelism
  - For example, data may not be splitable

Limitations of data parallelism

- Some problems do not allow for data parallelism
  - For example, data may not be splitable
- Limited parallelism
  - Want performance to be proportional to number of compute nodes
  - Too small splits may not be viable
  - Most of the time may be used in I/O operations



Limitations of data parallelism

- Some problems do not allow for data parallelism
  - For example, data may not be splitable
- Limited parallelism
  - Want performance to be proportional to number of compute nodes
  - Too small splits may not be viable
  - Most of the time may be used in I/O operations
- Overheads
  - Splitting data and combining partial results requires CPU time
  - Transmitting data between servers takes up network bandwidth

CompSci 401: Cloud Computing

The MapReduce Paradigm

Prof. Ítalo Cunha

昆山杜克大学
DUKE KUNSHAN UNIVERSITY

Counting words in a book

Input text

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. A iaculis at erat pellentesque adipiscing commodo elit at imperdiet. Gravida cum sociis natoque penatibus et. Nec ullamcorper sit amet risus nullam eget felis eget. Et netus et malesuada fames ac. Iaculis eu non diam phasellus vestibulum lorem sed risus. Ullamcorper eget nulla facilisi etiam dignissim diam quis. Sociis natoque penatibus et magnis dis. Aliquam id diam maecenas ultricies mi. Vitae turpis massa sed elementum tempus egestas. Sed id semper risus in hendrerit gravida. Odio morbi quis commodo odio aenean sed adipiscing diam. Arcu felis bibendum ut tristique et egestas. Enim sed faucibus turpis in eu mi bibendum. Molestie ac feugiat sed lectus vestibulum mattis. Vitae sapien pellentesque habitant morbi tristique senectus et netus et. Aliquam ultrices sagittis orci a scelerisque purus semper.

Auctor elit sed vulputate mi sit. Gravida neque convallis a cras. Vitae sapien pellentesque habitant morbi. Dignissim convallis aenean et tortor at risus viverra adipiscing. Diam sollicitudin tempor id eu nisl nunc mi ipsum faucibus. Lorem sed risus ultricies tristique nulla aliquet enim tortor at. Sem integer vitae justo eget magna fermentum iaculis eu. Iaculis at erat pellentesque adipiscing commodo elit. Dictum sit amet justo donec enim diam vulputate. Aenean pharetra magna ac placerat vestibulum. Scelerisque varius morbi enim nunc. Tristique senectus et netus et malesuada fames ac turpis. Dictum at tempor commodo ullamcorper a lacus. Malesuada pellentesque elit eget gravida cum sociis natoque. Eget nullam non nisi est sit amet. Suscipit adipiscing bibendum est ultricies. Viverra aliquet eget sit amet tellus cras adipiscing enim eu. Mattis pellentesque id nibh tortor id aliquet lectus proin. Non sodales neque sodales ut.

In pellentesque massa placerat duis ultricies lacus sed. A arcu cursus vitae congue. Purus sit amet volutpat consequat mauris nunc congue. Vitae sapien pellentesque habitant morbi. Fringilla phasellus faucibus scelerisque eleifend. Fringilla urna porttitor rhoncus dolor. Sed pulvinar proin gravida hendrerit lectus. Commodo odio aenean sed adipiscing diam donec. Elit scelerisque mauris pellentesque pulvinar pellentesque habitant morbi. Mauris in aliquam sem fringilla. In hac habitasse platea dictumst quisque sagittis purus sit. Quam adipiscing vitae proin sagittis nisl rhoncus mattis rhoncus. Non nisi est sit amet. Auctor eu augue ut lectus arcu bibendum at. Orci ac auctor augue mauris augue.

Vestibulum lectus mauris ultrices eros. Nisl suscipit adipiscing bibendum est ultricies. Eros in cursus turpis massa tincidunt dui ut. Mattis molestie a iaculis at erat pellentesque. Augue ut lectus arcu bibendum. Ut aliquam purus sit amet luctus venenatis lectus magna fringilla. Elit scelerisque mauris pellentesque pulvinar pellentesque habitant morbi. Nunc scelerisque viverra mauris in aliquam sem fringilla. Sodales ut eu sem integer vitae justo eget magna. A arcu cursus vitae congue mauris rhoncus aenean. Nunc eget lorem dolor sed viverra ipsum nunc aliquet. Risus pretium quam vulputate dignissim suspendisse. Fermentum et sollicitudin ac orci phasellus. Ut porttitor leo a diam sollicitudin tempor id. Lacus sed turpis tincidunt id aliquet risus feugiat in.

Nec dui nunc mattis enim. Aliquet risus feugiat in ante. Mi proin sed libero enim sed faucibus turpis in. Lacus luctus accumsan tortor posuere ac. Sollicitudin tempor id eu nisl nunc mi. Nam aliquam sem et tortor consequat id porta nibh venenatis. Arcu cursus euismod quis viverra nibh cras pulvinar mattis nunc. Nunc congue nisi vitae suscipit tellus mauris a diam maecenas. Id semper risus in hendrerit gravida rutrum quisque. Mollis aliquam ut porttitor leo a diam. Porttitor eget dolor morbi non arcu risus. Scelerisque fermentum dui faucibus in ornare quam viverra orci. Magnis dis parturient montes nascetur ridiculus mus mauris vitae ultricies. Adipiscing elit ut aliquam purus sit. Feugiat nibh sed pulvinar proin. Sit amet aliquam id diam maecenas. Facilisis volutpat est velit egestas dui id ornare arcu odio. A erat nam at lectus urna. Semper eget duis at tellus at urna condimentum mattis pellentesque.

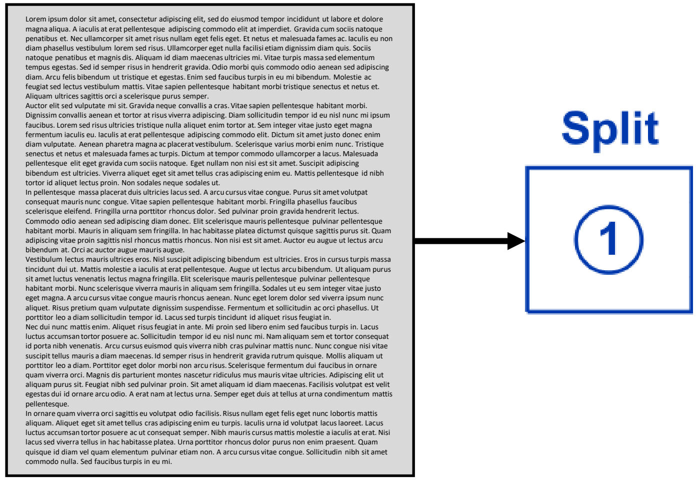
In ornare quam viverra orci sagittis eu volutpat odio facilisis. Risus nullam eget felis eget nunc lobortis mattis aliquam. Aliquet eget sit amet tellus cras adipiscing enim eu turpis. Iaculis urna id volutpat lacus laoreet. Lacus luctus accumsan tortor posuere ac ut consequat semper. Nibh mauris cursus mattis molestie a iaculis at erat. Nisi lacus sed viverra tellus in hac habitasse platea. Urna porttitor rhoncus dolor purus non enim praesent. Quam quisque id diam vel quam elementum pulvinar etiam non. A arcu cursus vitae congue. Sollicitudin nibh sit amet commodo nulla. Sed faucibus turpis in eu mi.

Result

Count	Word
10,500	the
9,968	to
8,553	a
5,972	at
4,851	of
4,269	from
4,055	and
3,448	with

Counting words in a book

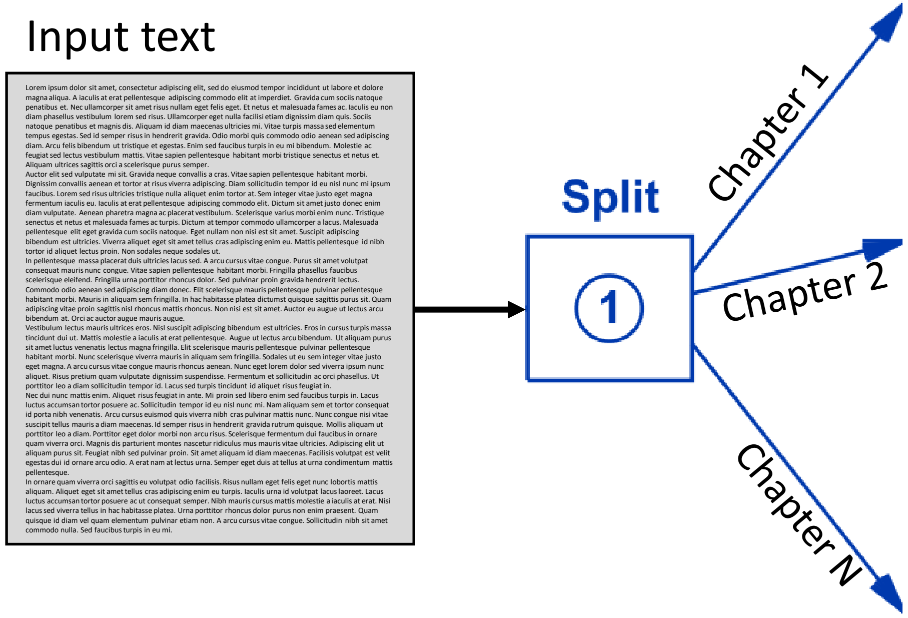
Input text



Result

Count	Word
10,500	the
9,968	to
8,553	a
5,972	at
4,851	of
4,269	from
4,055	and
3,448	with

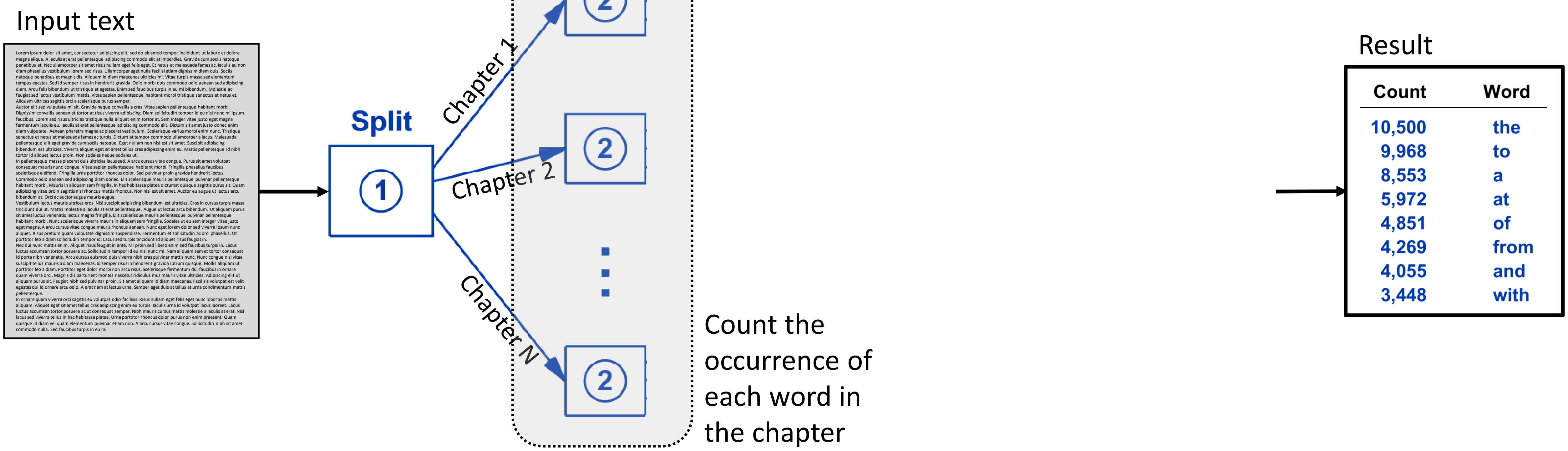
Counting words in a book



Result

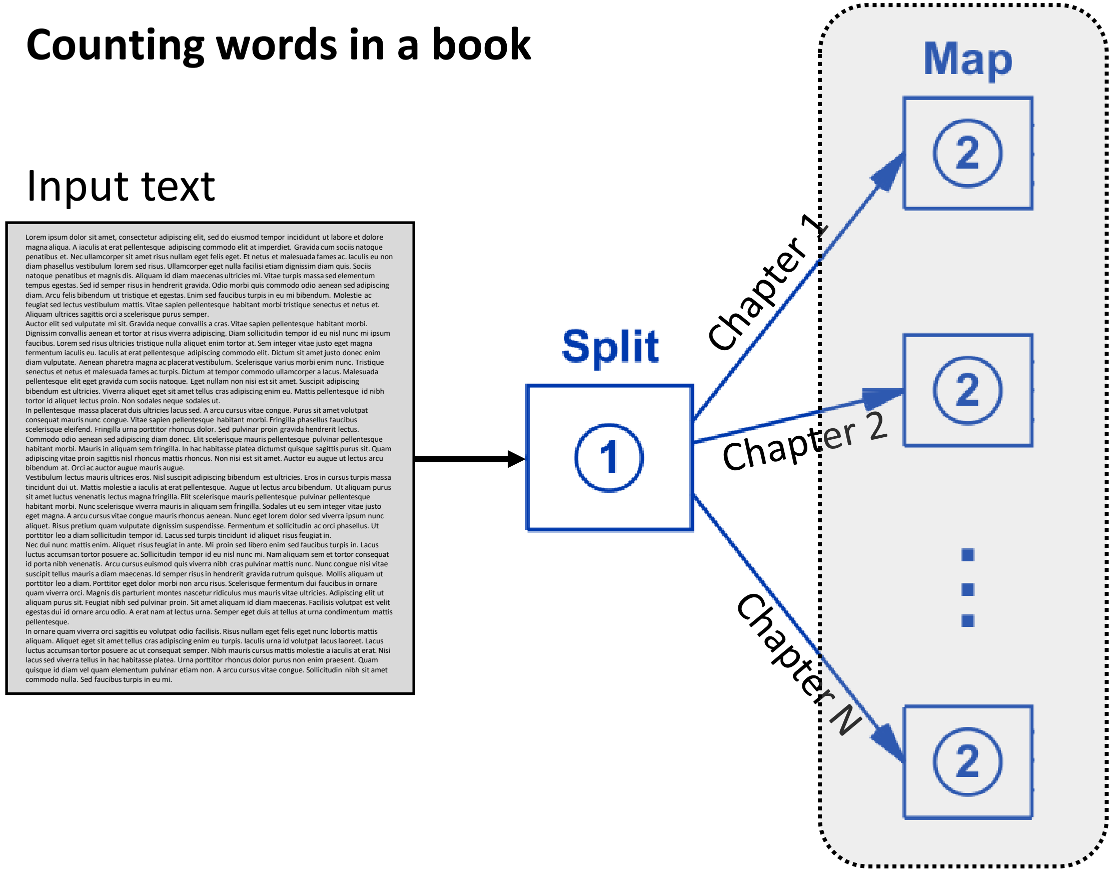
Count	Word
10,500	the
9,968	to
8,553	a
5,972	at
4,851	of
4,269	from
4,055	and
3,448	with

Counting words in a book



Counting words in a book

Input text



Count the occurrence of each word in the chapter

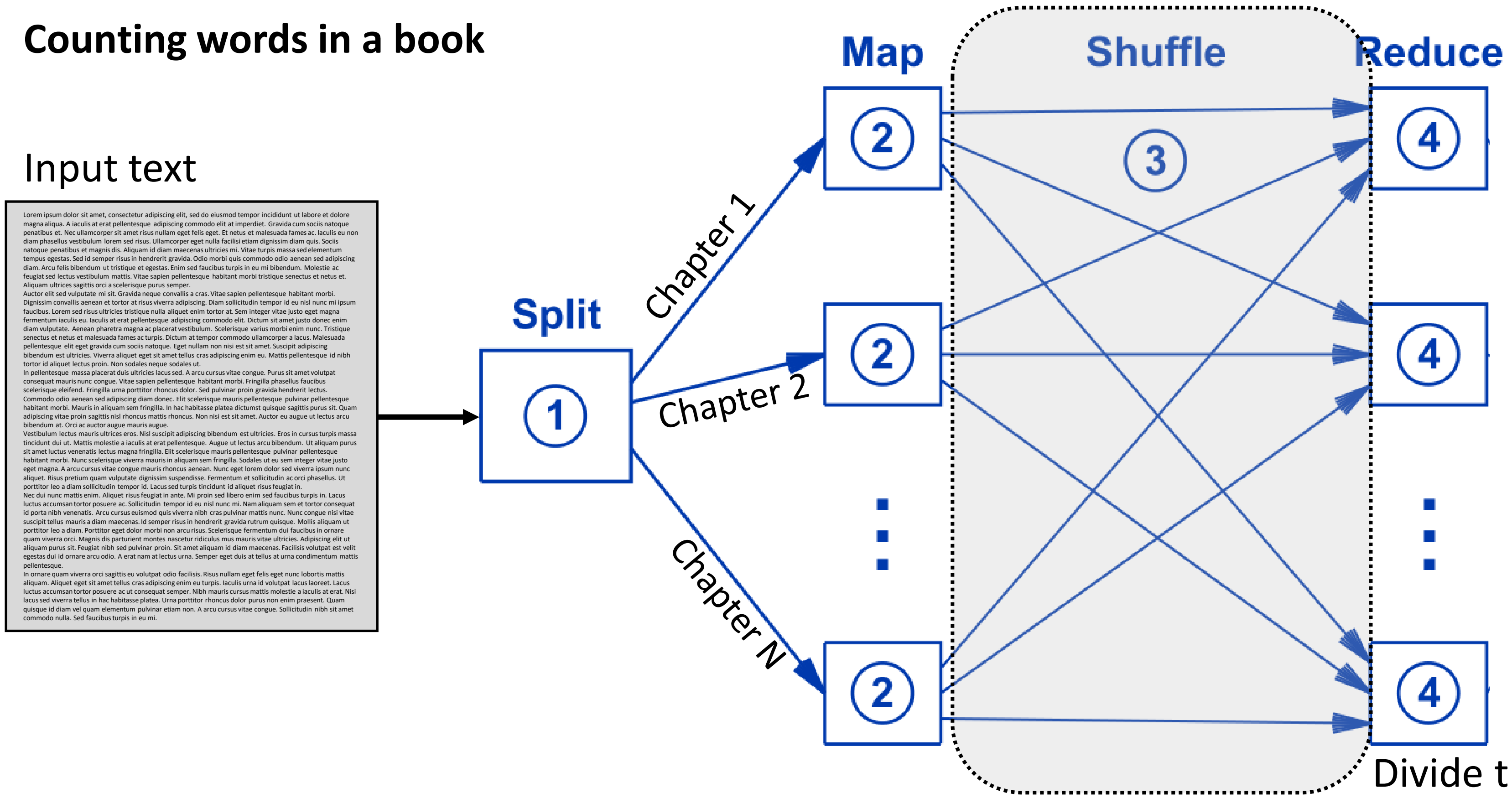
Result

Count	Word
10,500	the
9,968	to
8,553	a
5,972	at
4,851	of
4,269	from
4,055	and
3,448	with

Partial result after Map

Chapter 1		Chapter 2		Chapter 3	
288	the	166	the	489	the
191	a	62	to	376	a
170	and	61	a	192	to
149	of	59	of	136	of
144	to	57	and	128	and
89	that	36	that	101	is
71	is	34	in	75	that
66	in	18	is	59	an

Counting words in a book



Divide the work among instances of Reduce

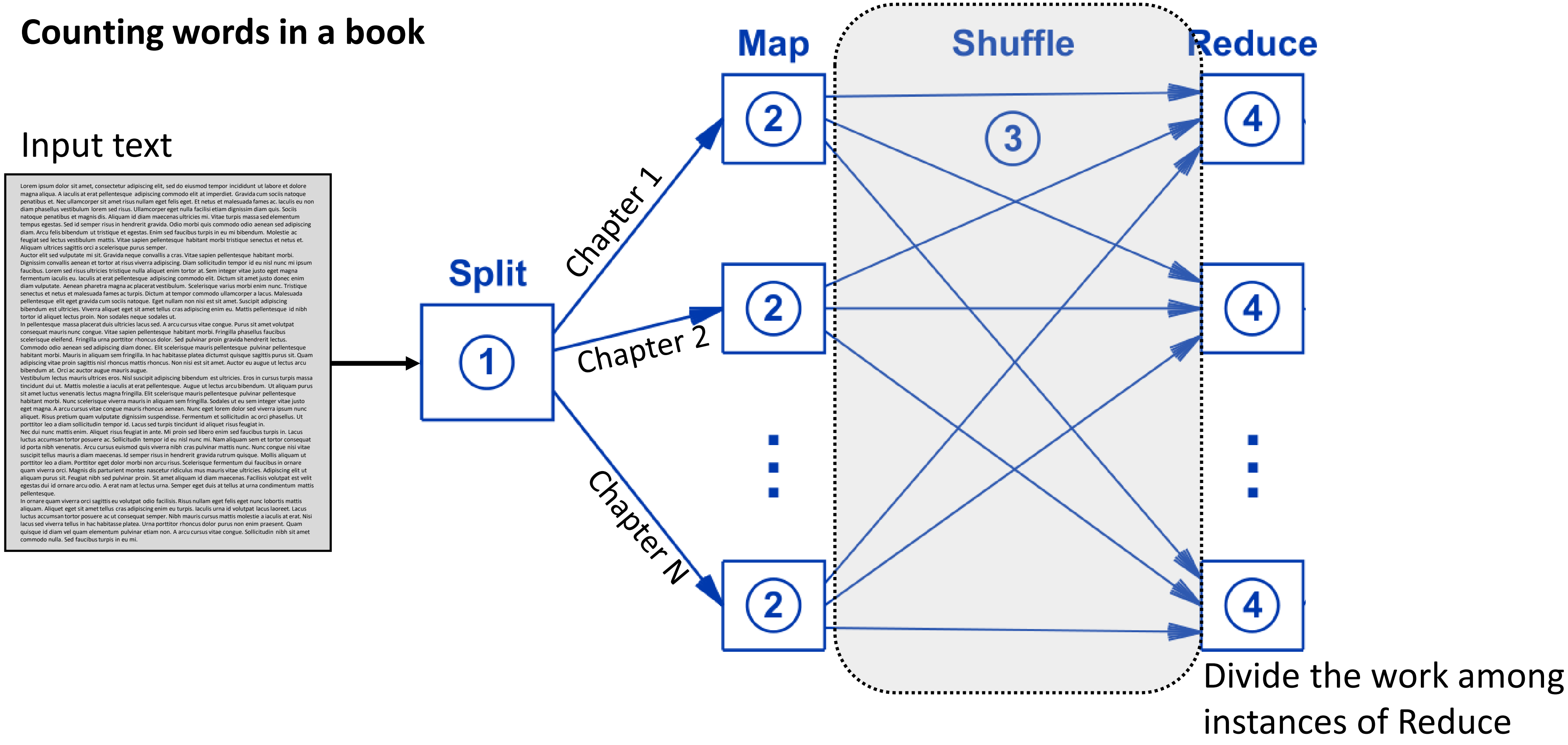
Result

Count	Word
10,500	the
9,968	to
8,553	a
5,972	at
4,851	of
4,269	from
4,055	and
3,448	with

Partial result after Map

Chapter 1		Chapter 2		Chapter 3	
288	the	166	the	489	the
191	a	62	to	376	a
170	and	61	a	192	to
149	of	59	of	136	of
144	to	57	and	128	and
89	that	36	that	101	is
71	is	34	in	75	that
66	in	18	is	59	an

Counting words in a book



Result

Count	Word
10,500	the
9,968	to
8,553	a
5,972	at
4,851	of
4,269	from
4,055	and
3,448	with

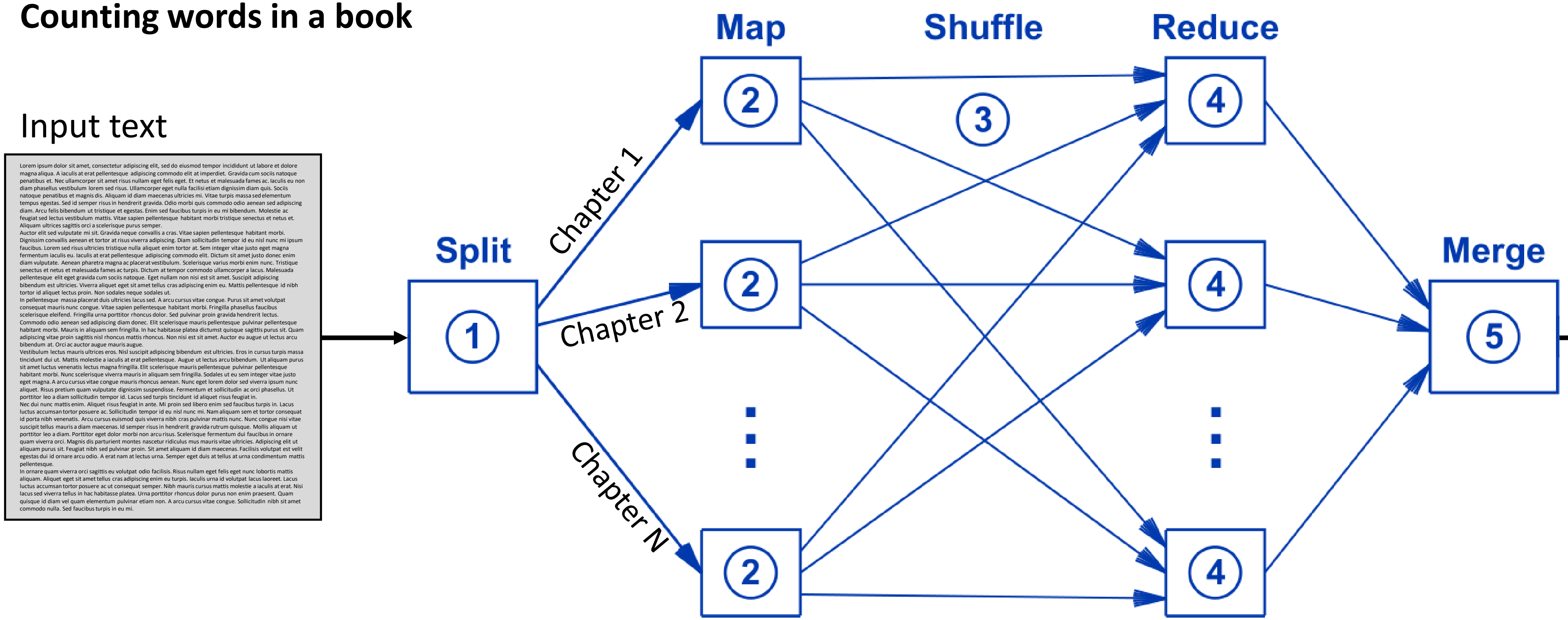
Partial result after Map

Chapter 1		Chapter 2		Chapter 3	
288	the	166	the	489	the
191	a	62	to	376	a
170	and	61	a	192	to
149	of	59	of	136	of
144	to	57	and	128	and
89	that	36	that	101	is
71	is	34	in	75	that
66	in	18	is	59	an

Shuffling of partial results

Reduce Instance	Words Starting With
1	A through E
2	F through J
3	K through O
4	P through T
5	U through Z

Counting words in a book



Result

Count	Word
10,500	the
9,968	to
8,553	a
5,972	at
4,851	of
4,269	from
4,055	and
3,448	with

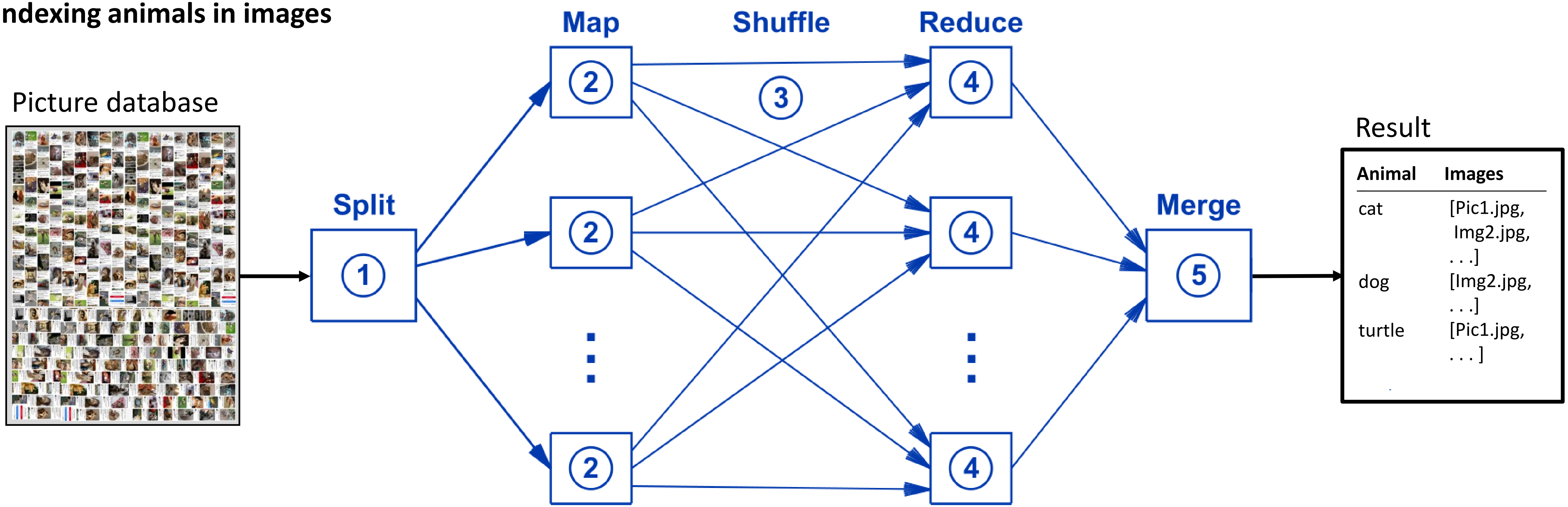
Partial result after Map

Chapter 1		Chapter 2		Chapter 3	
288	the	166	the	489	the
191	a	62	to	376	a
170	and	61	a	192	to
149	of	59	of	136	of
144	to	57	and	128	and
89	that	36	that	101	is
71	is	34	in	75	that
66	in	18	is	59	an

Shuffling of partial results

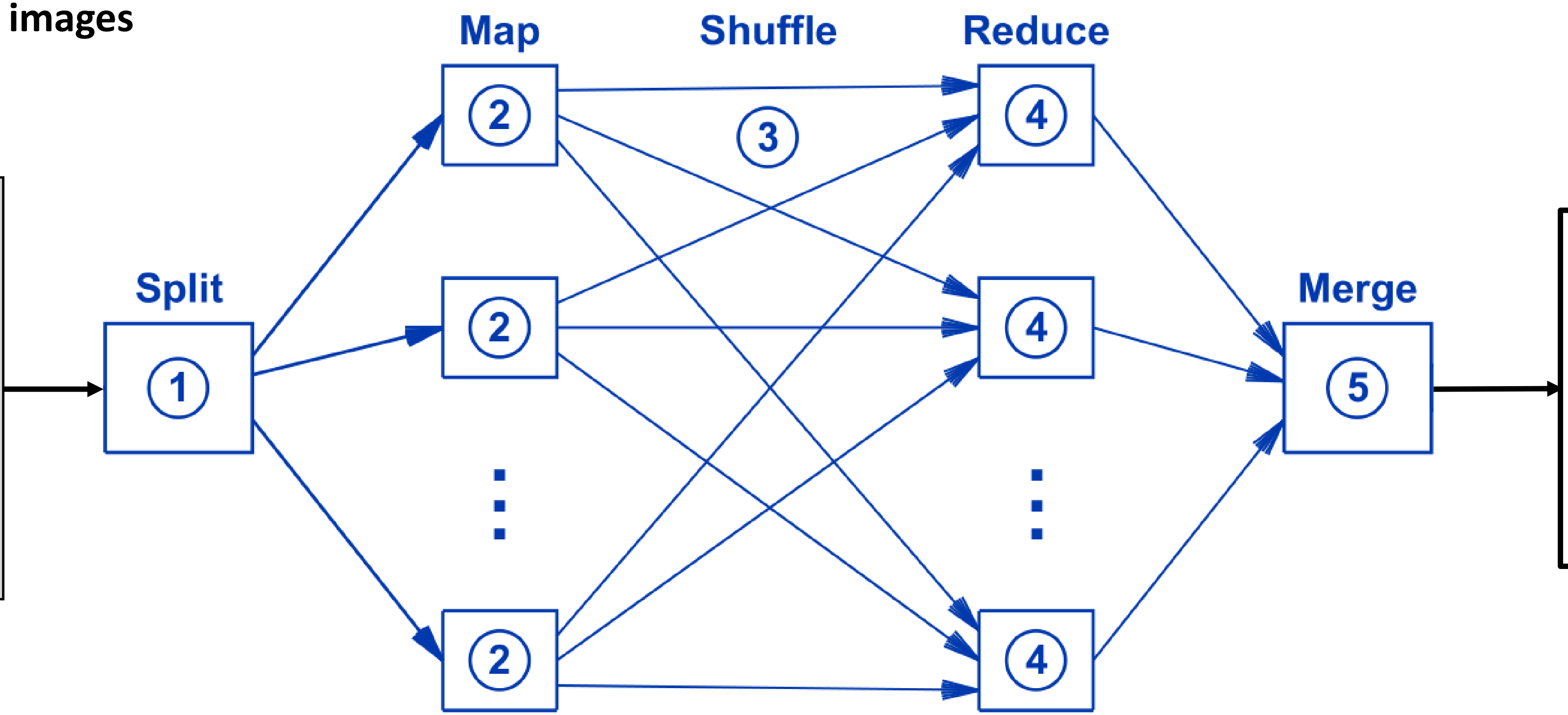
Reduce Instance	Words Starting With
1	A through E
2	F through J
3	K through O
4	P through T
5	U through Z

Indexing animals in images



Indexing animals in images

Picture database



Result

Animal	Images
cat	[Pic1.jpg, Img2.jpg, . . .]
dog	[Img2.jpg, . . .]
turtle	[Pic1.jpg, . . .]

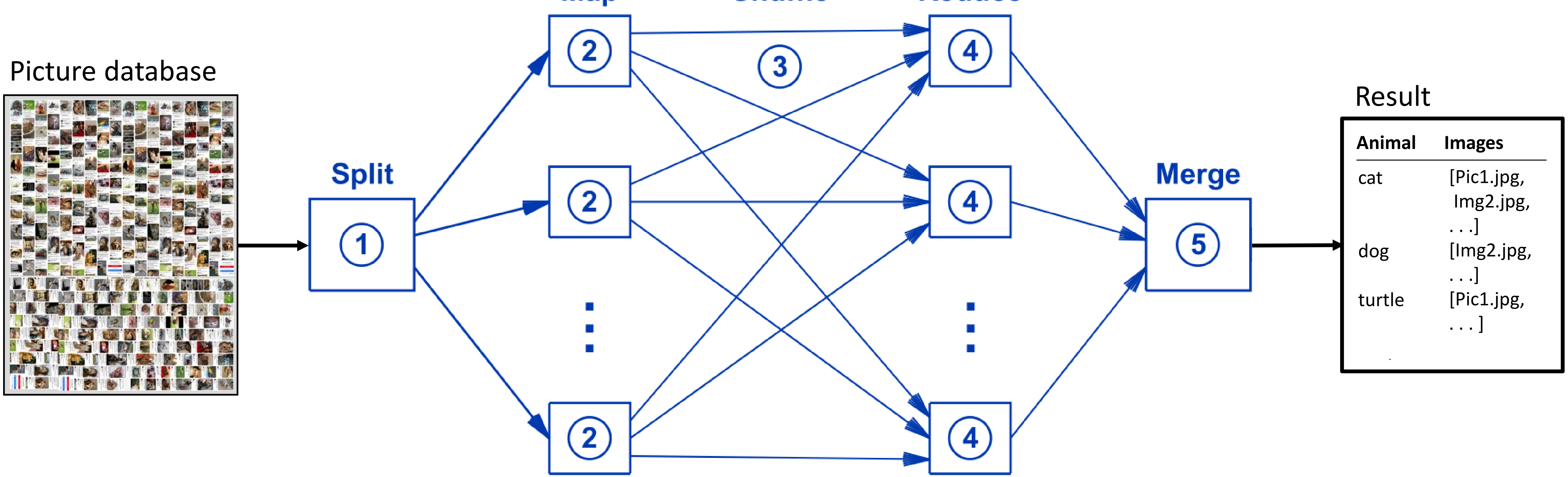
Partial result after Map

Split 1	Split 2	Split 3
cat Pic1.jpg	cat Img2.jpg	. . .
turtle Pic1.jpg	dog Img2.jpg	
. . .	. . .	

Pic1.jpg in split 1

Img2.jpg in split 2

Indexing animals in images



Partial result after Map

Split 1	Split 2	Split 3
cat Pic1.jpg	cat Img2.jpg	. . .
turtle Pic1.jpg	dog Img2.jpg	
. . .	. . .	

Shuffling of partial results

Reduce Instance	Animals Starting With
1	A through E
2	F through J
3	K through O
4	P through T
5	U through Z

Mathematical formulation of MapReduce

- Split step maps each data item *d* to a Map instance M_i
  - d → M_i
- Map step transforms data into a list *L*
  - Each item *j* is a pair (K_j , V_j)
  - K_j is a key, V_j is a value
- The shuffle step maps each key to a Reduce instance R_k
  - $K_j \rightarrow R_k$

Mathematical formulation of MapReduce

- Split step maps each data item *d* to a Map instance M_i
  - d → M_i
- Map step transforms data into a list *L*
  - Each item *j* is a pair (K_j , V_j)
  - K_j is a key, V_j is a value
- The shuffle step maps each key to a Reduce instance R_k
  - K_j → R_k

In our examples, keys were:

- Word W
- Animal A

And values were:

- Count of occurrences of word W in chapter
- Picture name where animal A appeared

Mathematical formulation of MapReduce

- Split step maps each data item *d* to a Map instance M_i
  - d → M_i
- Map step transforms data into a list *L*
  - Each item *j* is a pair (K_j , V_j)
  - K_j is a key, V_j is a value
- The shuffle step maps each key to a Reduce instance R_k
  - K_j → R_k
  - Shuffle stage merges and sorts values for each key K_j before calling Reduce
    - Reduce gets an iterator

Summary of examples

	Split → Map	Map → Shuffle	Shuffle → Reduce
Book word count	(offset, line of text)	multiple (word, count in line)	(word, list of counts)
Animal identification	(byte offset, figure)	multiple (animal, figure file)	(animal, list of files)

CompSci 401: Cloud Computing

Distributing Load

Prof. Ítalo Cunha

昆山杜克大学
DUKE KUNSHAN UNIVERSITY

Considerations when splitting input

- Load distribution
  - We want compute instances to perform comparable amounts of work
    - The total execution time is determined by the longest-running instance
  - Should send equal amounts of input to each compute instance
    - Complexity of some tasks do not correlate with data size (use a specific mechanism)
- Small but meaningful data pieces
  - Smaller data pieces make it easier to distribute load, but
  - Logical piece of data may be needed in Map and Reduce stages
    - For example, we need a whole picture to detect an animal (Map)
    - Need the complete name of the file where the animal appeared (Reduce)

Distributing load with hash functions

- A hash function maps an input to a random output
  - Hash function has *one* (random) output for each input key
  - One-to-one mapping of inputs to outputs
- Can be used to distribute load when there are many items
  - Many English words start with the letter ‘s’
  - Use whole word as key instead of just the first letter
  - Hash the word, use output to send data to a specific reduce instance

Distributing load with hash functions

- A hash function maps an input to a random output
  - Hash function has *one* (random) output for each input key
  - One-to-one mapping of inputs to outputs
- Can be used to distribute load when there are many items
  - Many English words start with the letter ‘s’
  - Use whole word as key instead of just the first letter
  - Hash the word, use output to send data to a specific reduce instance

Why not build a large table with the mapping?

- Do not need for the table and maintaining the mapping
- Do not need to know the words (table rows) in advance

Distributing load incurs overhead

- Splitting and merging data
- Transmitting data over the network
- Simple tasks on small datasets in MapReduce may take longer
- MapReduce is better suited for large datasets and complex tasks

CompSci 401: Cloud Computing

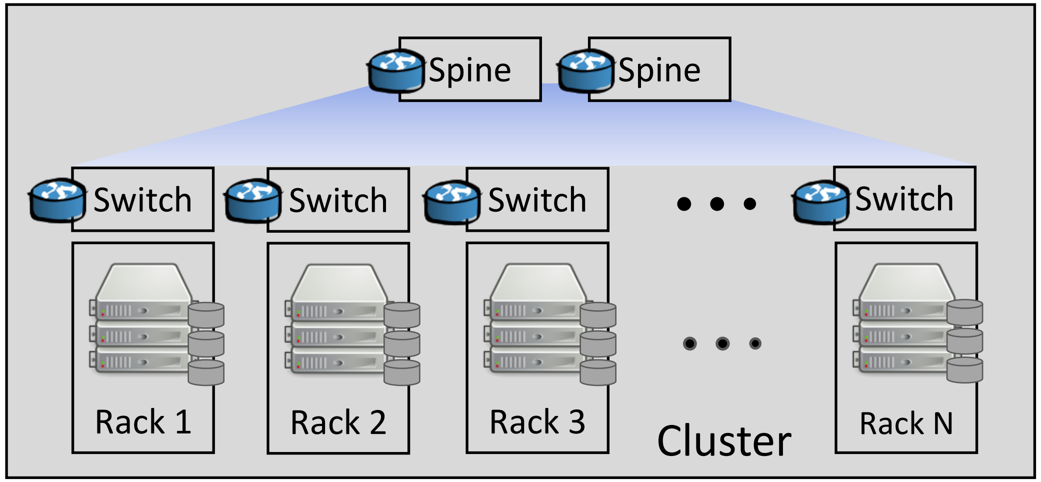
Apache Hadoop

Prof. Ítalo Cunha

昆山杜克大学
DUKE KUNSHAN
UNIVERSITY

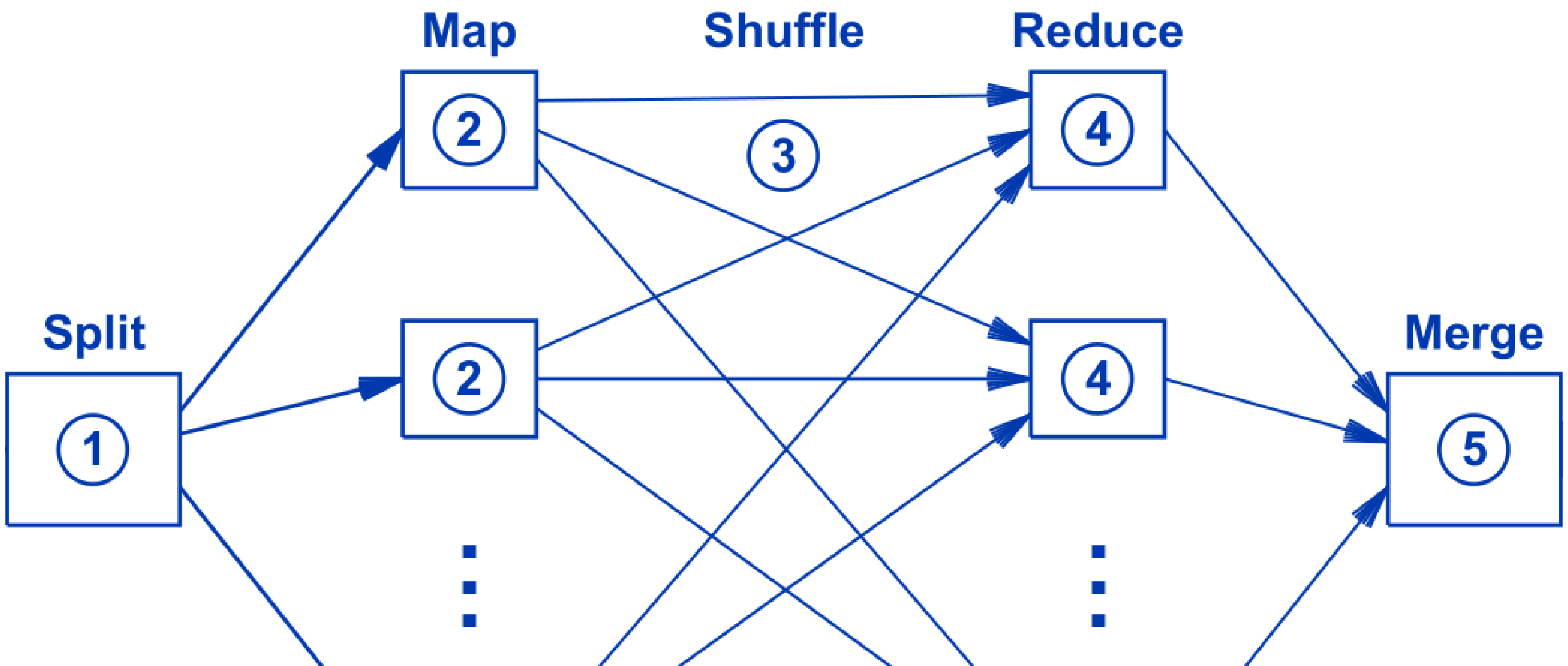
Hadoop hardware cluster model

- Hadoop assumes servers with compute, memory, *and disks*
- Thousands of nodes to handle large volumes of data
- Lots of disk to support large datasets, replication, and I/O



Hadoop processing module

- Hadoop has a “main” node and “worker” nodes
- Worker nodes can run Map and Reduce instances
- Split, shuffle, and merge usually provided by the platform



Hadoop processing component

- Hadoop has a “main” node and “worker” nodes
- Worker nodes can run Map and Reduce instances
- Split, shuffle, and merge usually provided by the platform
  - Reading and writing data, transmitting data over the network
- Instance spawning, monitoring, failure detection, and recovery
  - With thousands of nodes, one might fail during the execution of a task

Hadoop processing component

- Hadoop has a “main” node and “worker” nodes
- Worker nodes can run Map and Reduce instances
- Split, shuffle, and merge usually provided by the platform
  - Reading and writing data, transmitting data over the network
- Instance spawning, monitoring, failure detection, and recovery
  - With thousands of nodes, one might fail during the execution of a task

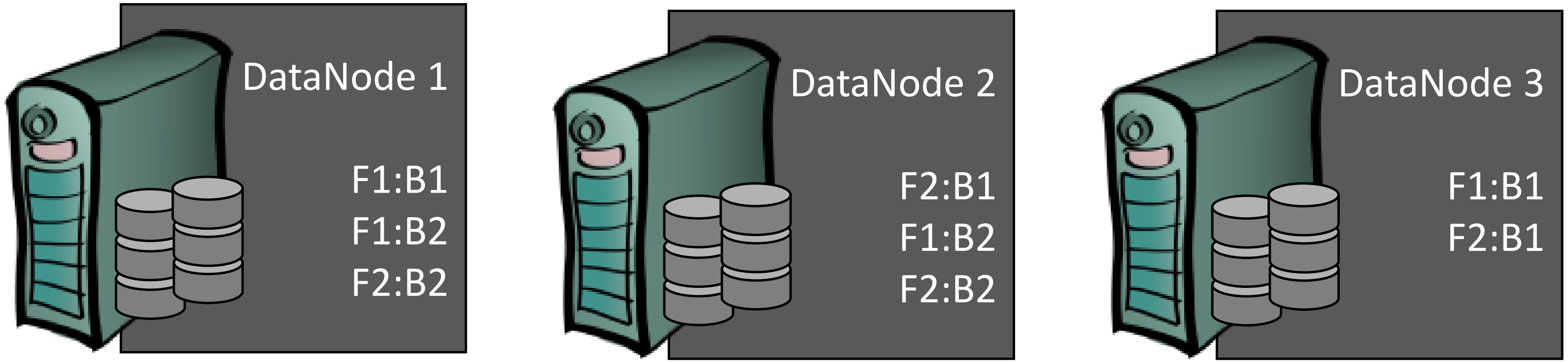
- We only need to write the Map and Reduce classes

HDFS (Hadoop Distributed File System)

- Hadoop handles massive volumes of data
- Data is managed within HDFS
- Specifically tailored to Hadoop’s needs

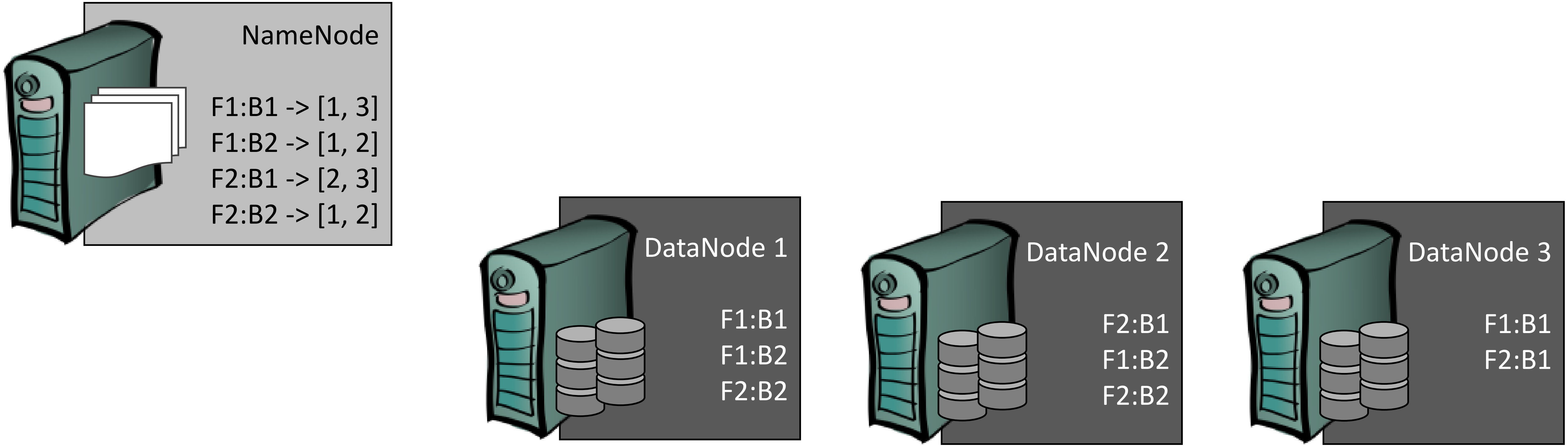
NameNode and DataNodes

- HDFS splits files into large (128MB) blocks
  - DataNodes store blocks
  - NameNode indexes files → blocks → DataNodes



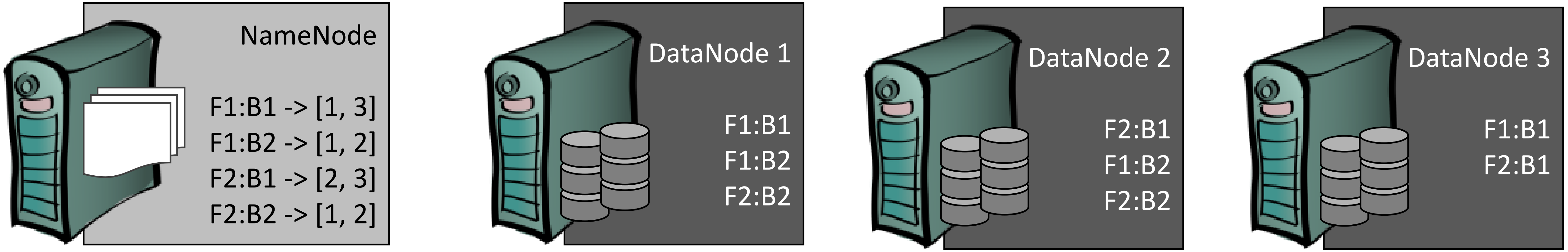
NameNode and DataNodes

- HDFS splits files into large (128MB) blocks
  - DataNodes store blocks
  - NameNode indexes files → blocks → DataNodes



Replication for fault resilience

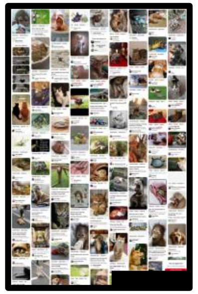
- Each block is replicated three times by default
  - Allows rebuilding if a disk or server fails
- Hadoop is aware of cluster hierarchy, and distributes blocks across different failure groups
  - For example, servers in a rack share power and the top-of-rack switch, spread blocks across servers in multiple racks



HDFS and MapReduce

- HDFS and MapReduce are tightly linked
- Programmer can split data to fit block size
  - A block is large enough to store many records, allows for improved performance
  - Records may not align with block size, but not a lot of wasted space

Block 1



Block 2

Example: Many figure files in a block, will waste some space at the end where we cannot fit a figure

HDFS and MapReduce

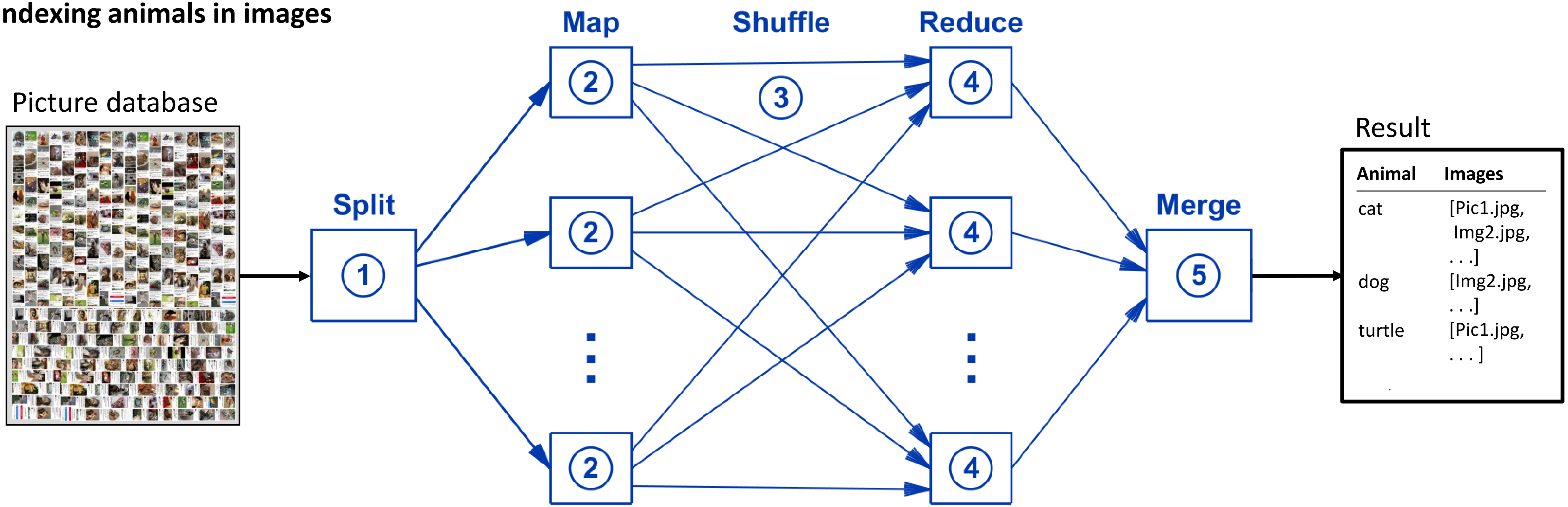
- HDFS and MapReduce are tightly linked
- Programmer can split data to fit block size
  - A block is large enough to store many records, allows for improved performance
  - Records may not align with block size, but not a lot of wasted space
- File semantics optimized for MapReduce
  - MapReduce reads data sequentially
  - HDFS only has read and append operations: Cannot modify existing content
- Colocation of computation and data
  - Hadoop tries to spawn Map instances on DataNodes where the data is stored
  - Reduce network overhead

HDFS and MapReduce

- HDFS and MapReduce are tightly linked
- Programmer can split data to fit block size
  - A block is large enough to store many records, allows for improved performance
  - Records may not align with block size, but not a lot of wasted space
- File semantics optimized for MapReduce
  - MapReduce reads data sequentially
  - HDFS only has read and append operations: Cannot modify existing content
- Colocation of computation and data
  - Hadoop tries to spawn Map instances on DataNodes where the data is stored
  - Reduce network overhead

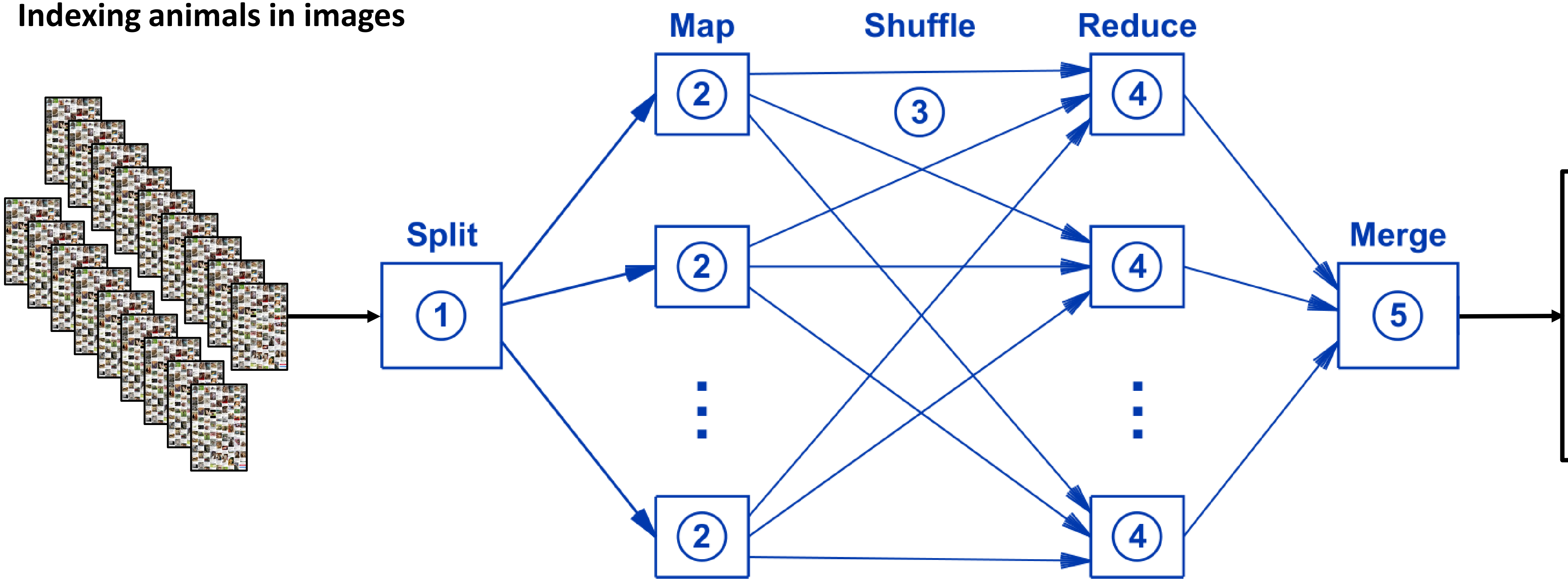
HDFS and MapReduce

Indexing animals in images



HDFS and MapReduce

Indexing animals in images



Result

Animal	Images
cat	[Pic1.jpg, Img2.jpg, . . .]
dog	[Img2.jpg, . . .]
turtle	[Pic1.jpg, . . .]

HDFS and MapReduce

Indexing animals in images