Data-Parallel to Distributed Data-Parallel

Big Data Analysis with Scala and Spark

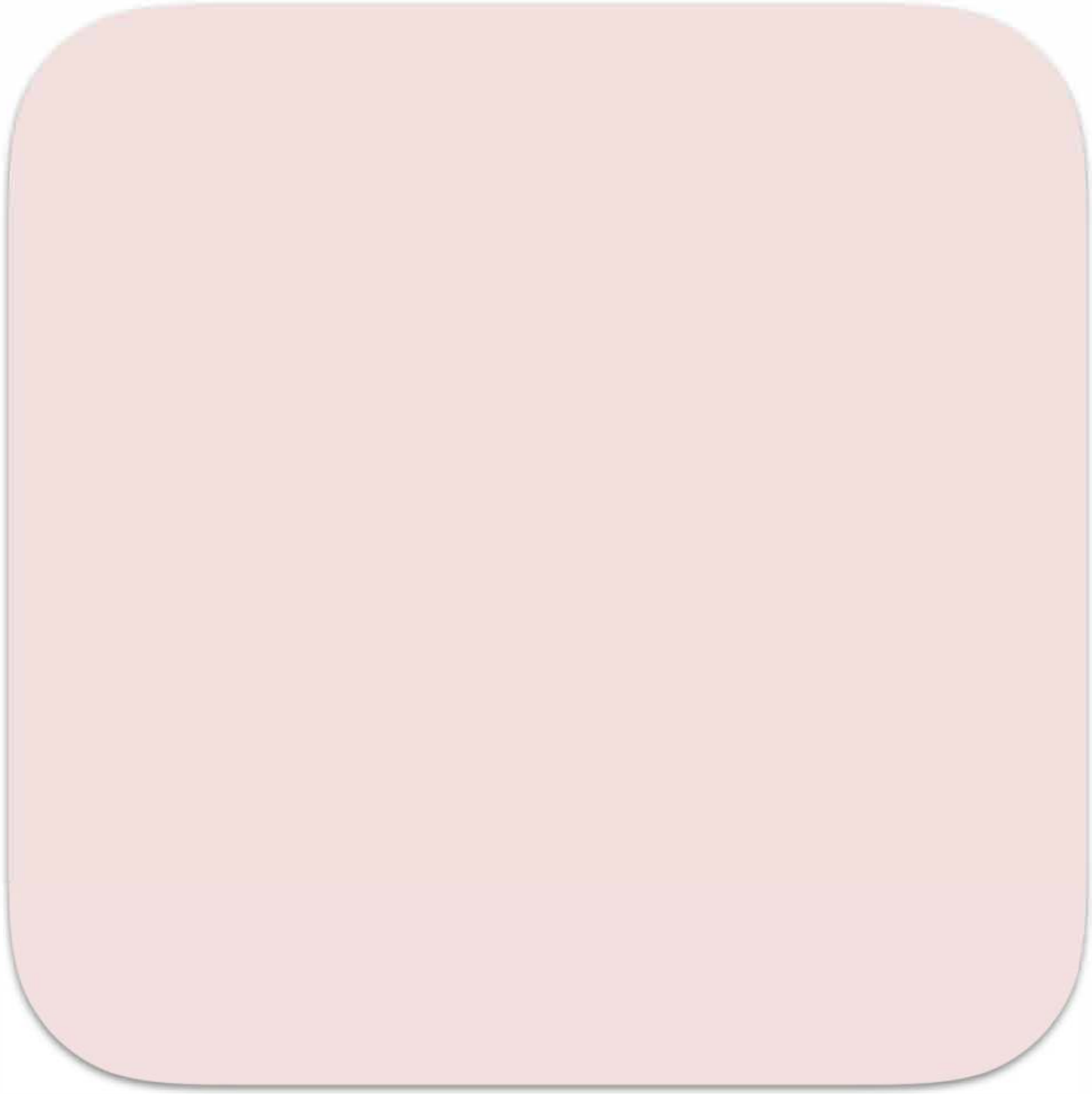
Heather Miller

Visualizing **Shared Memory** Data Parallelism

What does data-parallel look like?

Visualizing **Shared Memory** Data Parallelism

What does data-parallel look like?



Compute Node
(Shared Memory)

Visualizing **Shared Memory** Data Parallelism

What does data-parallel look like?

Compute Node
(Shared Memory)

Visualizing **Shared Memory** Data Parallelism

What does data-parallel look like?

```
val res =
  jar.map(jellyBean => doSomething(jellyBean))
```

`jar`

Collection

Compute Node

(Shared Memory)

Visualizing **Shared Memory** Data Parallelism

What does data-parallel look like?

```
val res =
  jar.map(jellyBean => doSomething(jellyBean))
```

Compute Node
(Shared Memory)

Shared memory data parallelism:

- Split the data.
- Workers/threads independently operate on the data shards in parallel.
- Combine when done (if necessary).

Visualizing **Shared Memory** Data Parallelism

What does data-parallel look like?

```scala
val res =
  jar.map(jellyBean => doSomething(jellyBean))
```

Compute Node
(Shared Memory)

Shared memory data parallelism:

- Split the data.
- Workers/threads independently operate on the data shards in parallel.
- Combine when done (if necessary).

Visualizing Shared Memory Data Parallelism

What does data-parallel look like?

```
val res =
  jar.map(jellyBean => doSomething(jellyBean))
```

Compute Node
(Shared Memory)

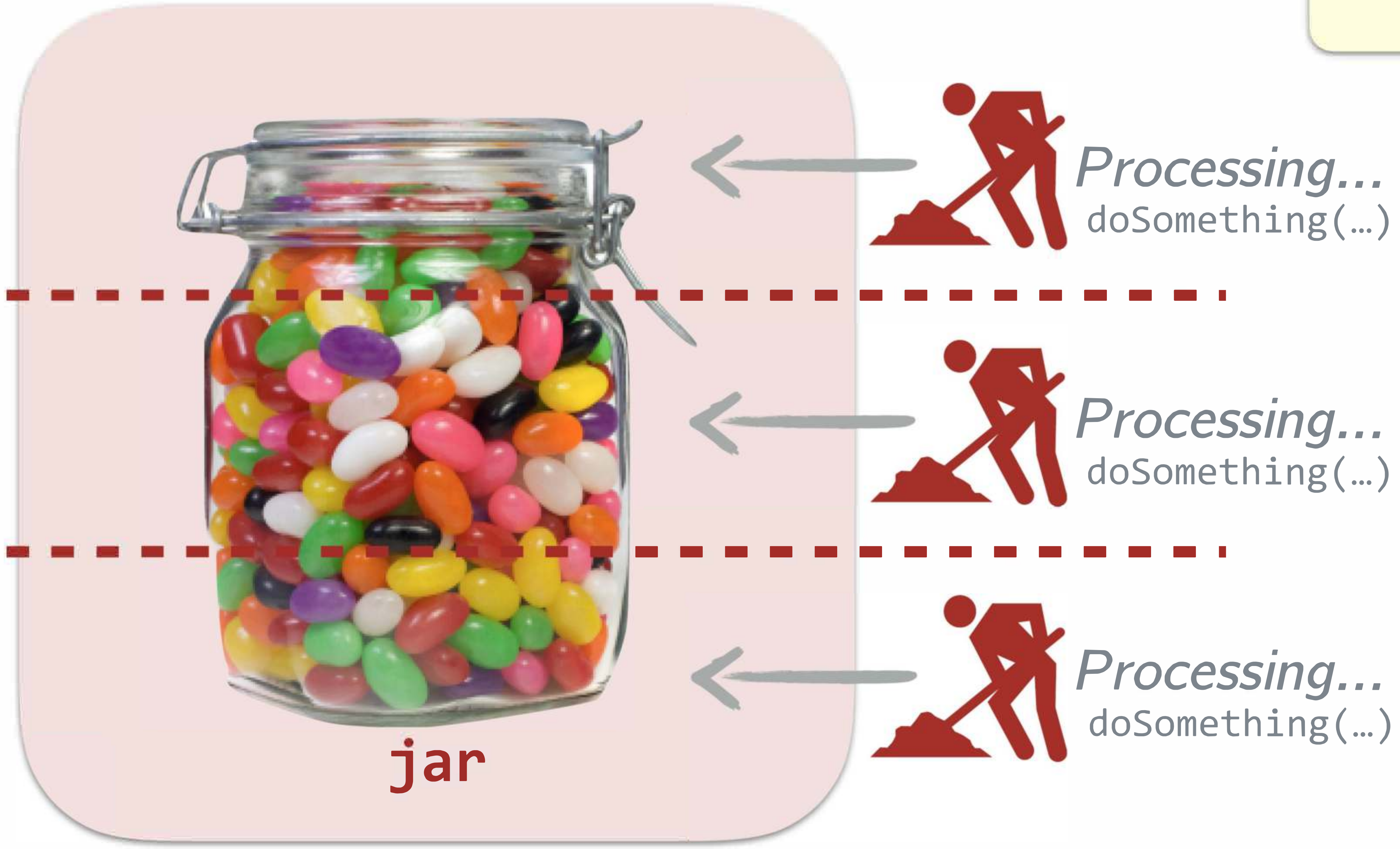
Shared memory data parallelism:

- Split the data.
- Workers/threads independently operate on the data shards in parallel.
- Combine when done (if necessary).

Visualizing **Shared Memory** Data Parallelism

What does data-parallel look like?

```
val res =
  jar.map(jellyBean => doSomething(jellyBean))
```



Compute Node
(Shared Memory)

Shared memory data parallelism:

- Split the data.
- Workers/threads independently operate on the data shards in parallel.
- Combine when done (if necessary).

Visualizing Shared Memory Data Parallelism

What does data-parallel look like?

```
val res =
  jar.map(jellyBean => doSomething(jellyBean))
```

Compute Node
(Shared Memory)

Shared memory data parallelism:

- Split the data.
- Workers/threads independently operate on the data shards in parallel.
- Combine when done (if necessary).

Scala's Parallel Collections is a collections abstraction over shared memory data-parallel execution.

Visualizing **Distributed** Data-Parallelism

What does **distributed** data-parallel look like?

Shared memory data parallelism:

- Split the data.
- Workers/threads independently operate on the data shards in parallel.
- Combine when done (if necessary).

Visualizing **Distributed** Data-Parallelism

What does **distributed** data-parallel look like?

Distributed
~~Shared-memory~~ data parallelism:

- Split the data over several nodes.
- Nodes independently operate on the data shards in parallel.
- Combine when done (if necessary).

Visualizing **Distributed** Data-Parallelism

What does **distributed** data-parallel look like?

Distributed data parallelism:

- Split the data over several nodes.
- Nodes independently operate on the data shards in parallel.
- Combine when done (if necessary).

Visualizing **Distributed** Data-Parallelism

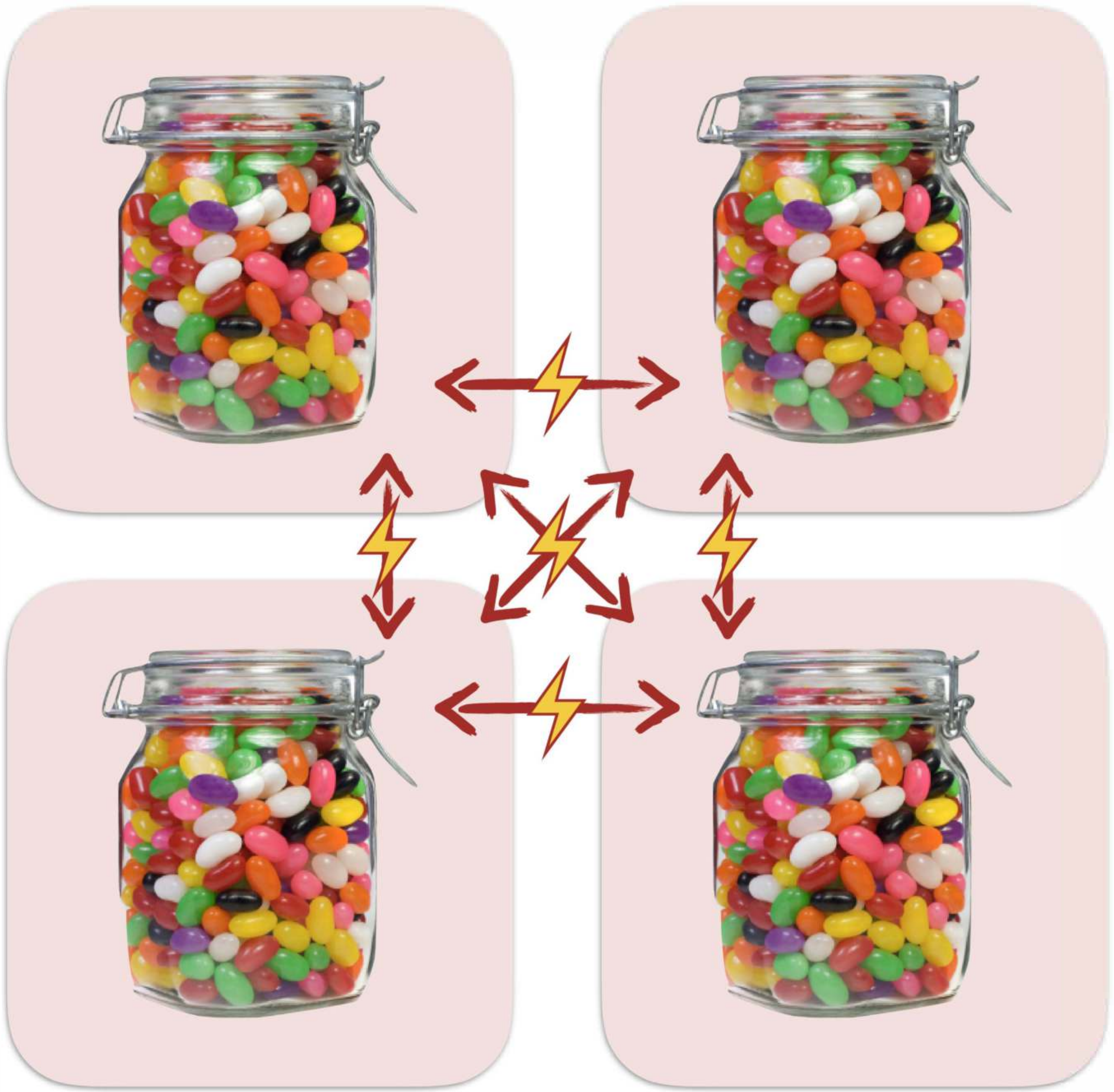
What does **distributed** data-parallel look like?

Distributed data parallelism:

- Split the data over several nodes.
- Nodes independently operate on the data shards in parallel.
- Combine when done (if necessary).

Visualizing **Distributed** Data-Parallelism

What does **distributed** data-parallel look like?



Distributed data parallelism:

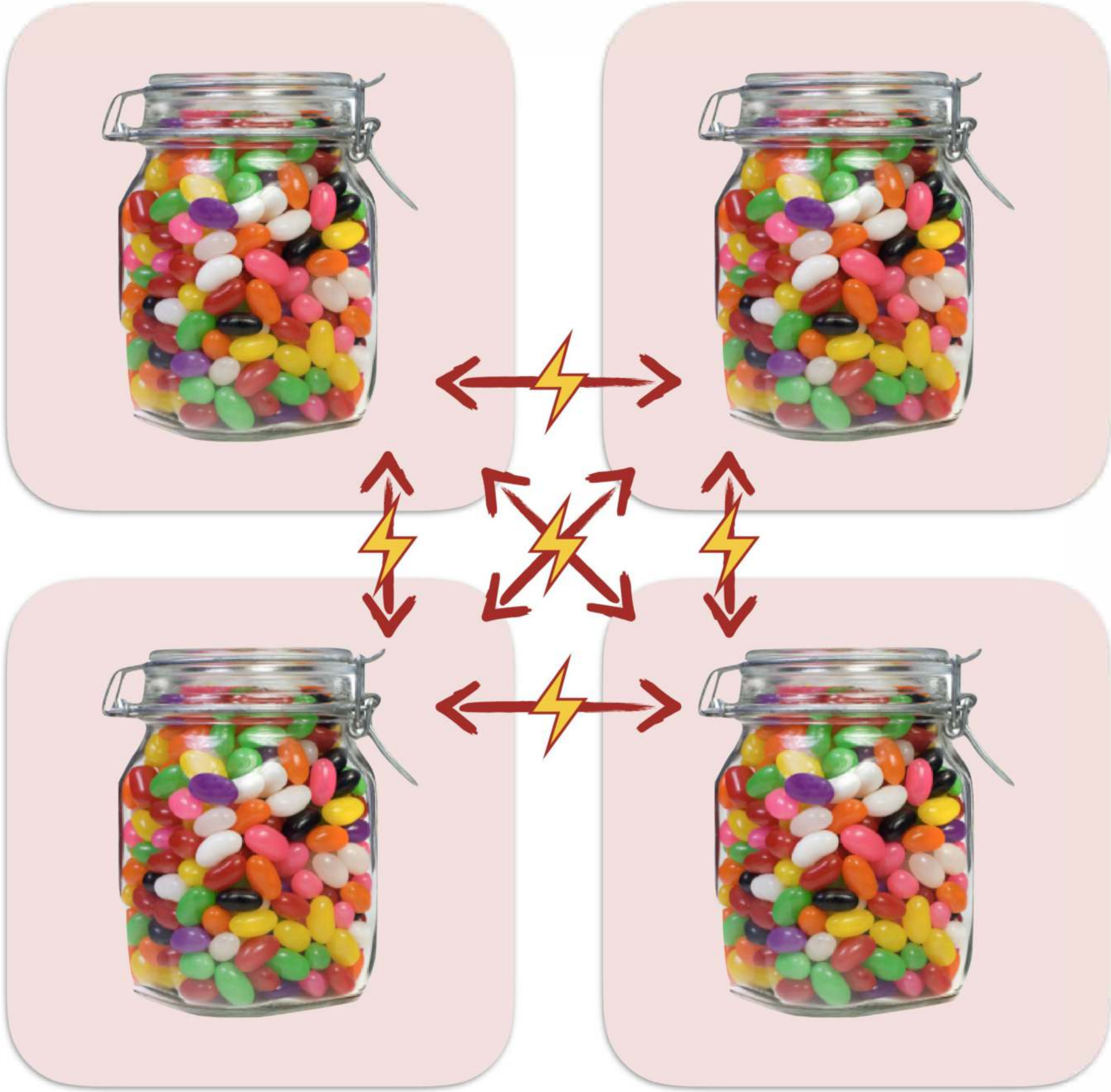
- Split the data over several nodes.
- Nodes independently operate on the data shards in parallel.
- Combine when done (if necessary).

New concern:

Now we have to worry about network latency between workers.

Visualizing **Distributed** Data-Parallelism

What does **distributed** data-parallel look like?



```scala
val res =
  jar.map(jellyBean => doSomething(jellyBean))
```

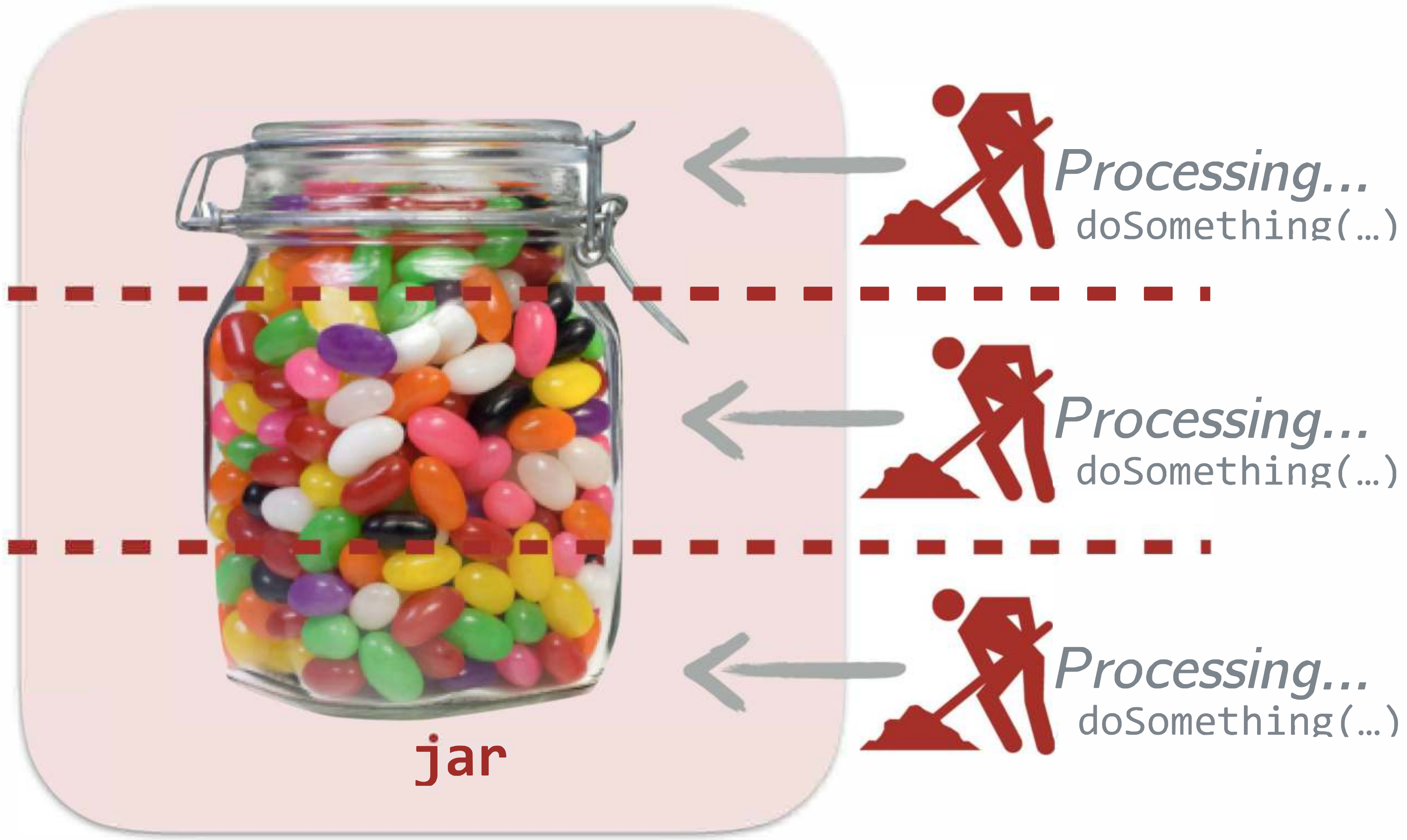
Distributed data parallelism:

- Split the data over several nodes.
- Nodes independently operate on the data shards in parallel.
- Combine when done (if necessary).

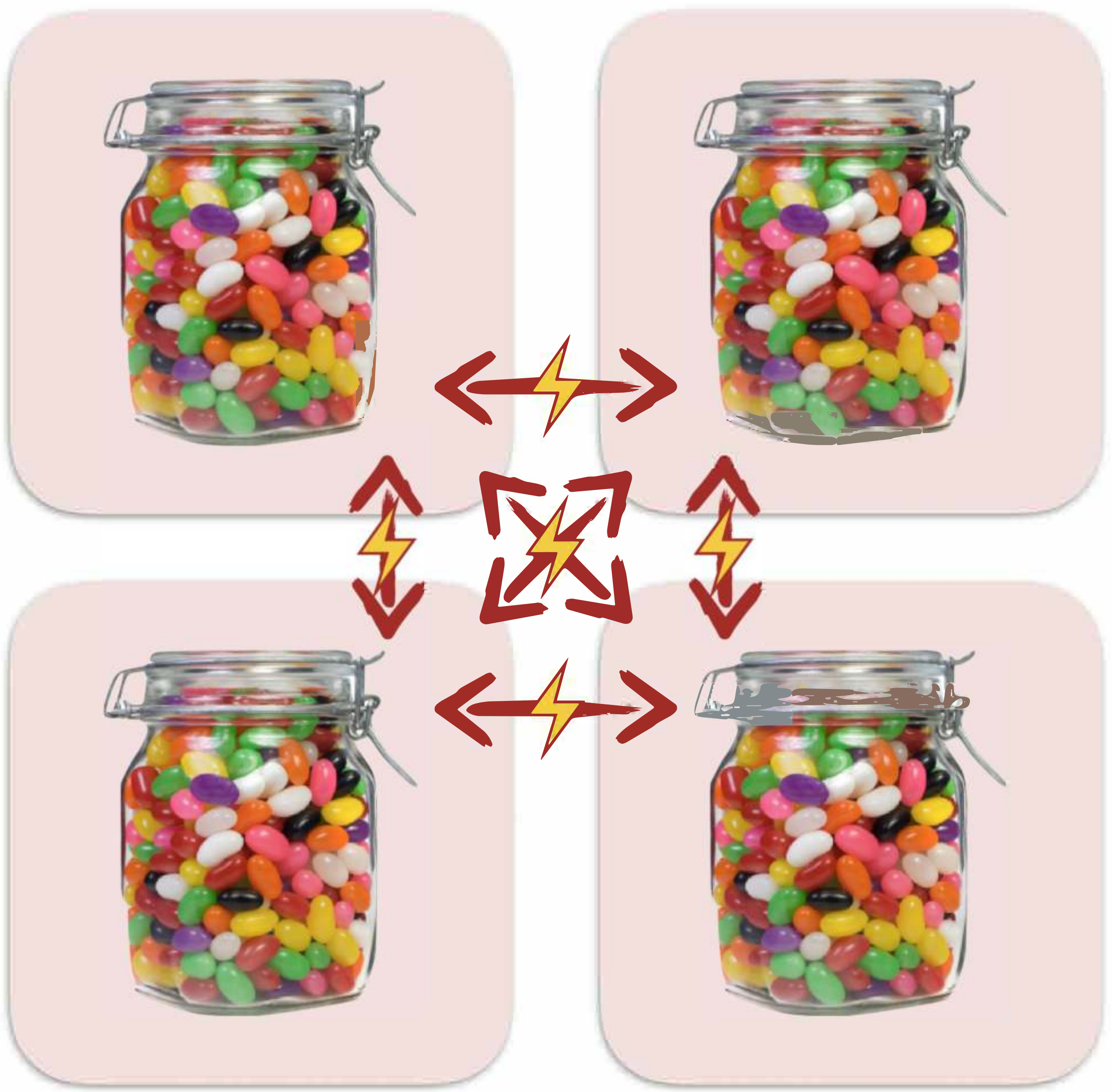
However, like parallel collections, we can keep collections abstraction over distributed data-parallel execution.

Data-Parallel to **Distributed** Data-Parallel

Shared memory:



Distributed:

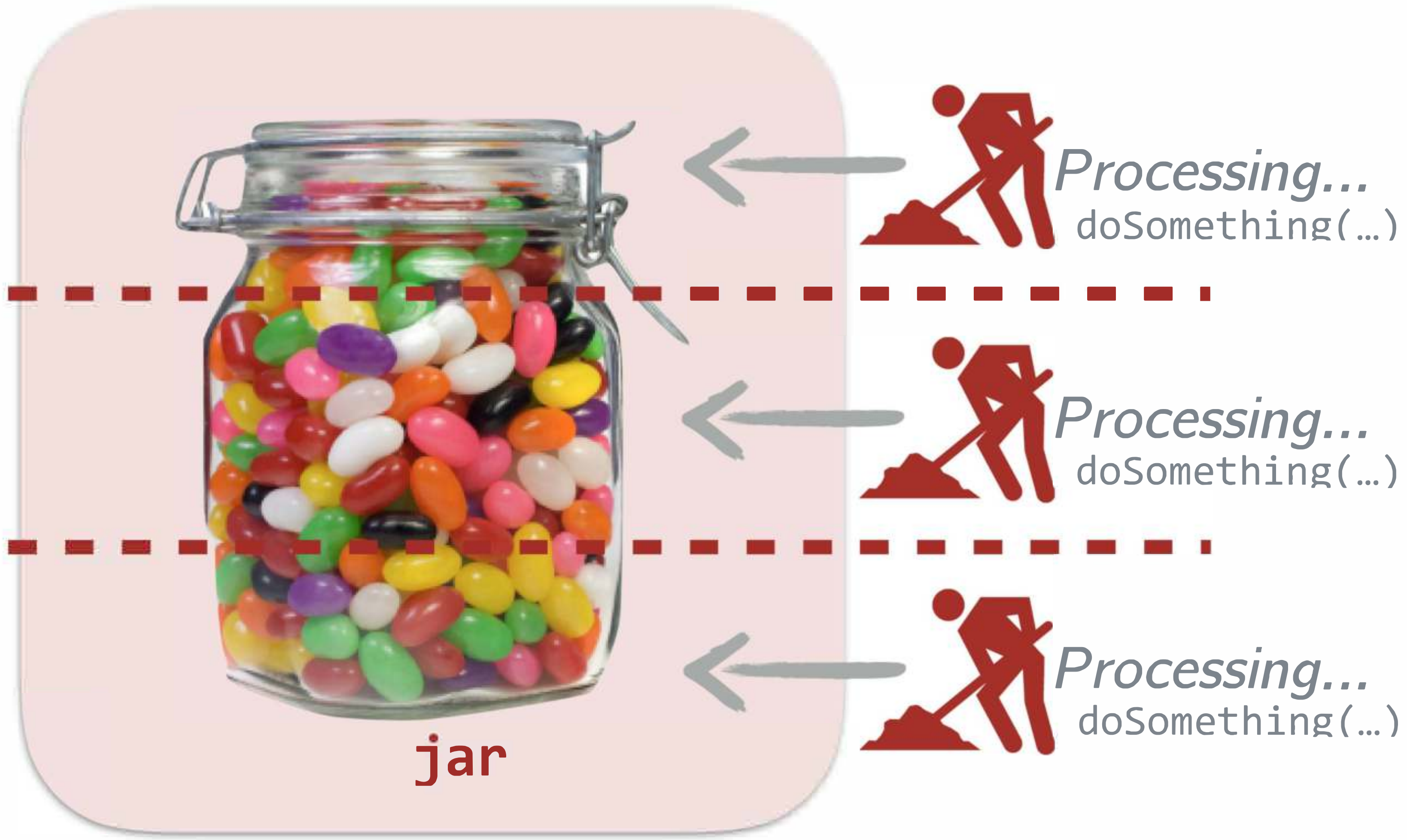


Shared memory case: Data-parallel programming model. Data partitioned in memory and operated upon in parallel.

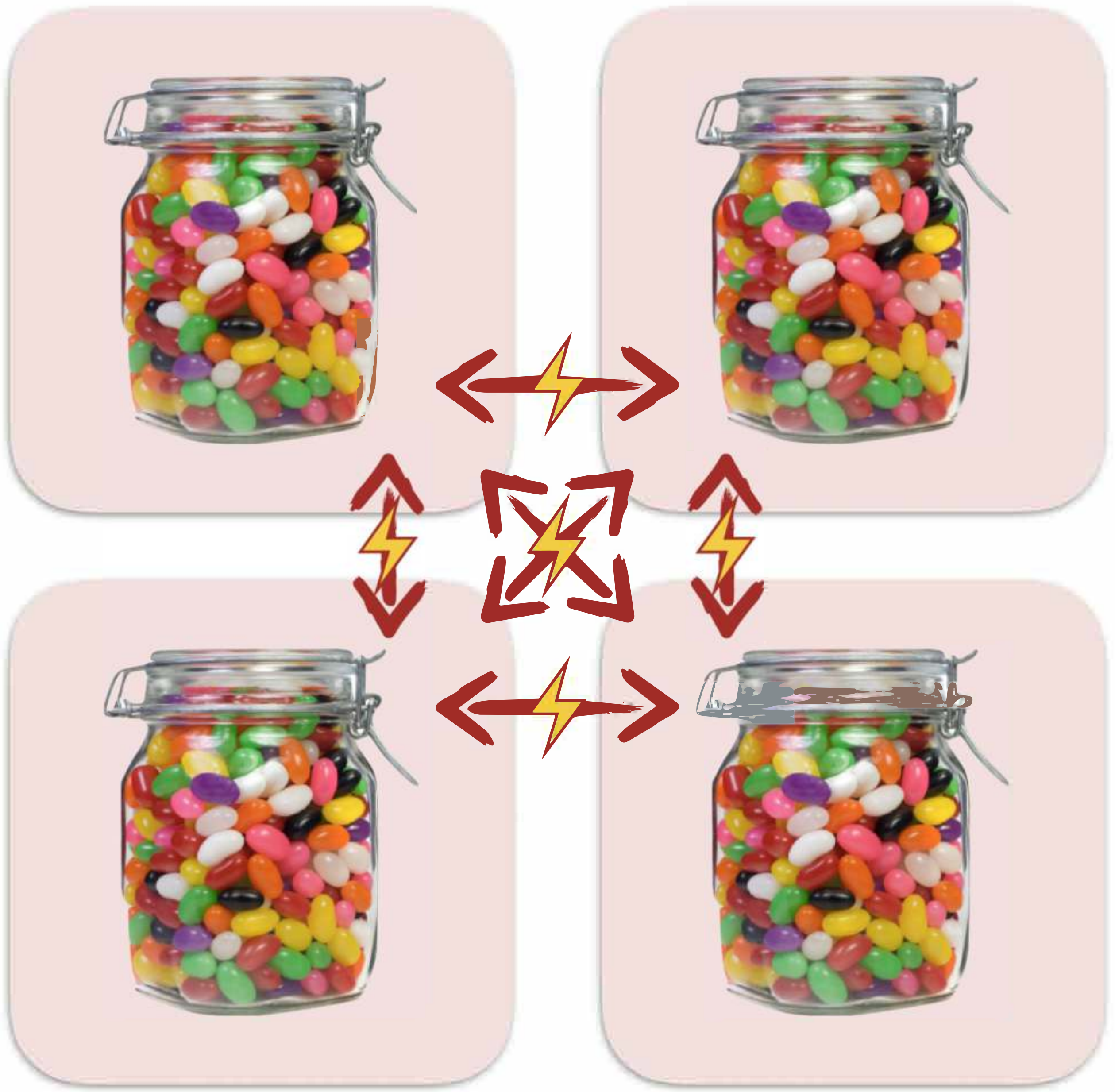
Distributed case: Data-parallel programming model. Data partitioned between machines, network in between, operated upon in parallel.

Data-Parallel to **Distributed** Data-Parallel

Shared memory:



Distributed:



Overall, most all properties we learned about related to shared memory data-parallel collections can be applied to their distributed counterparts.
E.g., watch out for non-associative reduction operations!

.reduce(_-_)

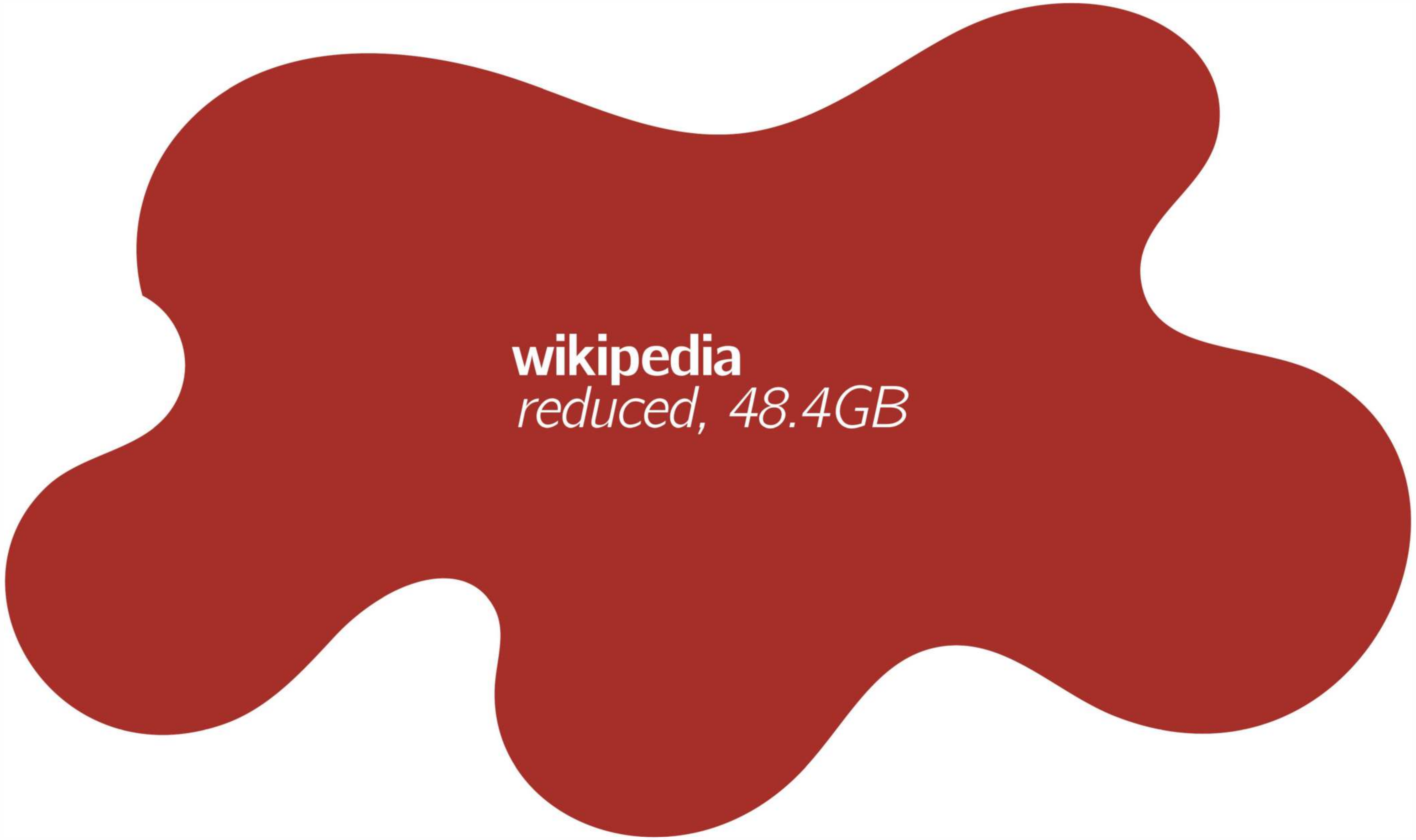
However, must now consider **latency** when using our model.

Apache Spark

Throughout this part of the course we will use the **Apache Spark** framework for distributed data-parallel programming.

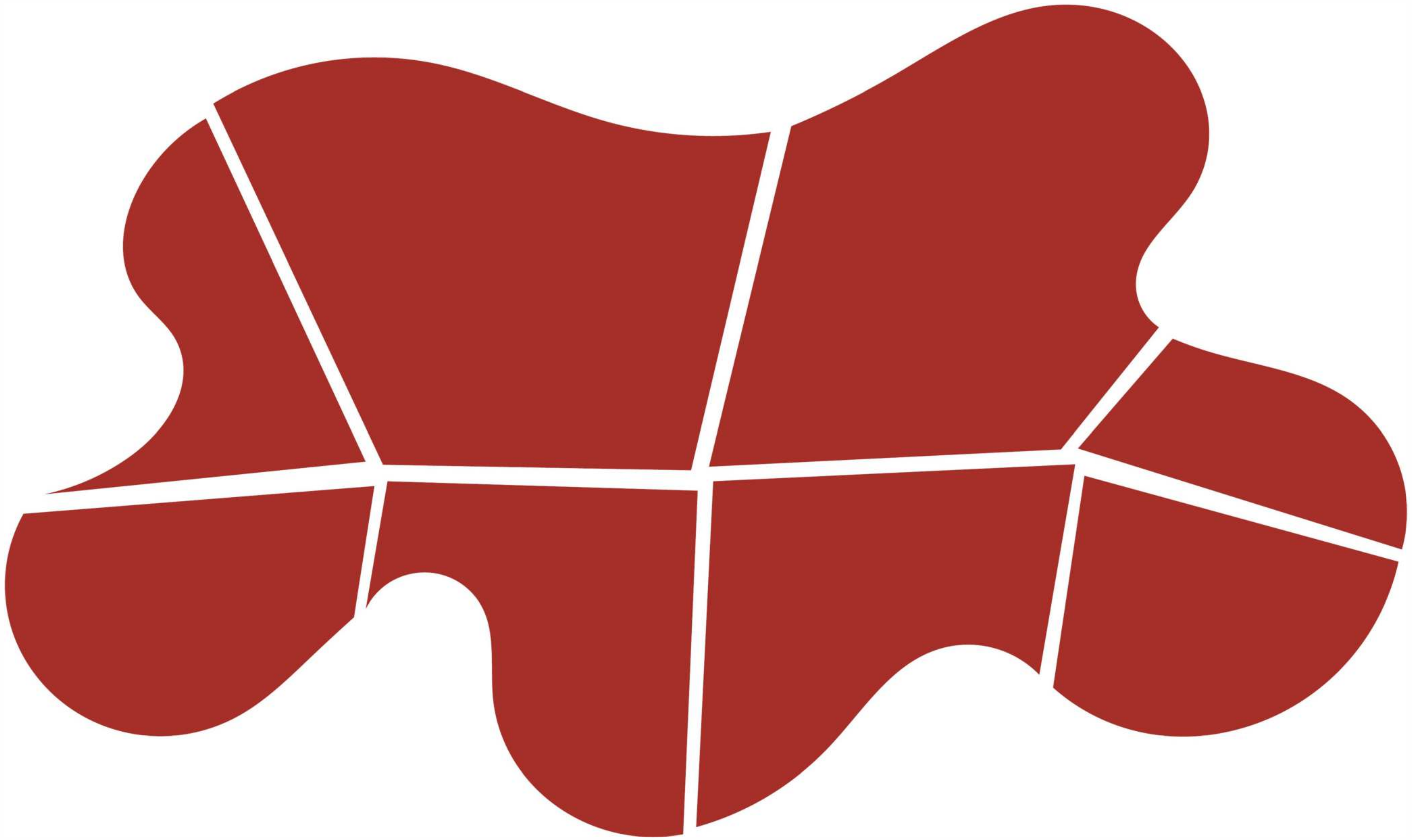
Spark implements a distributed data parallel model called **Resilient Distributed Datasets (RDDs)**

Distributed Data-Parallel: High Level Illustration



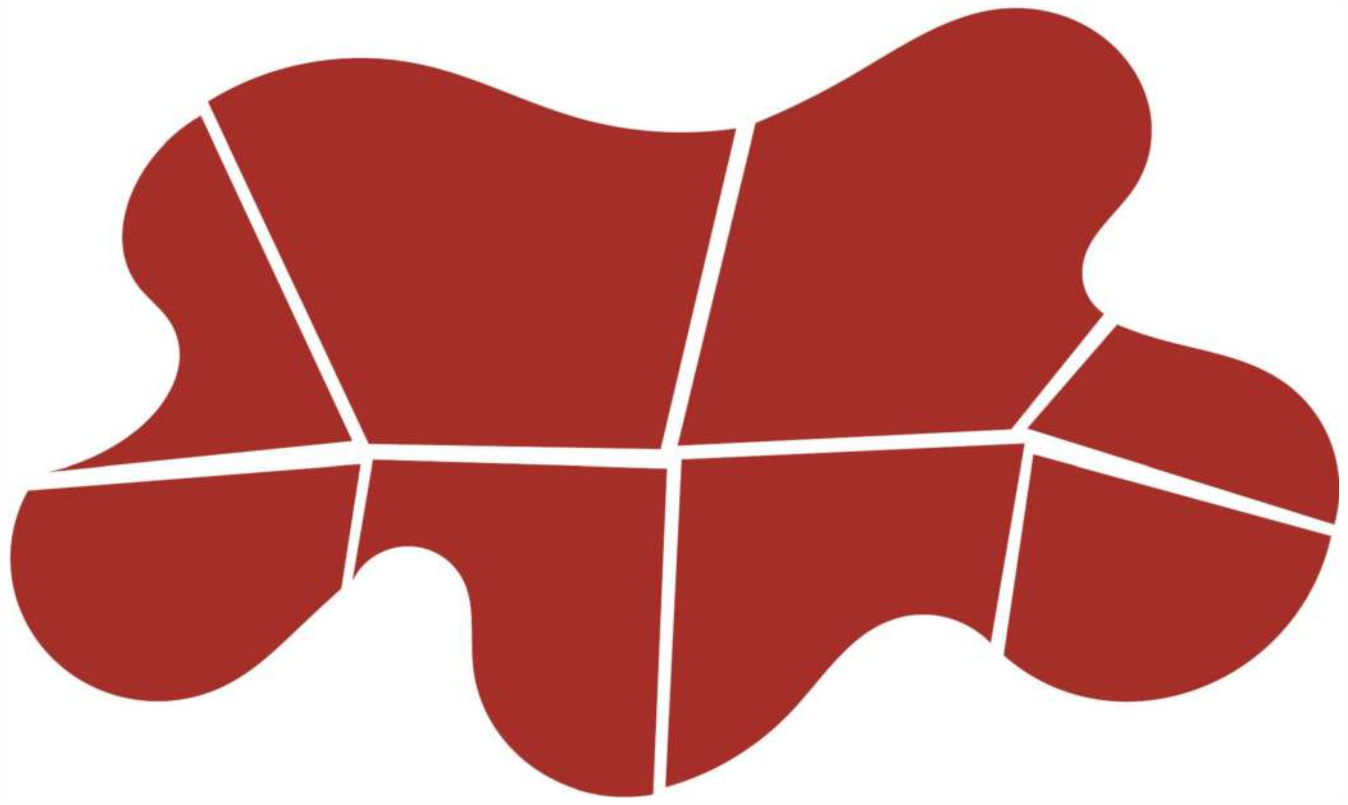
Given some large dataset that can't fit into memory on a single node...

Distributed Data-Parallel: High Level Illustration



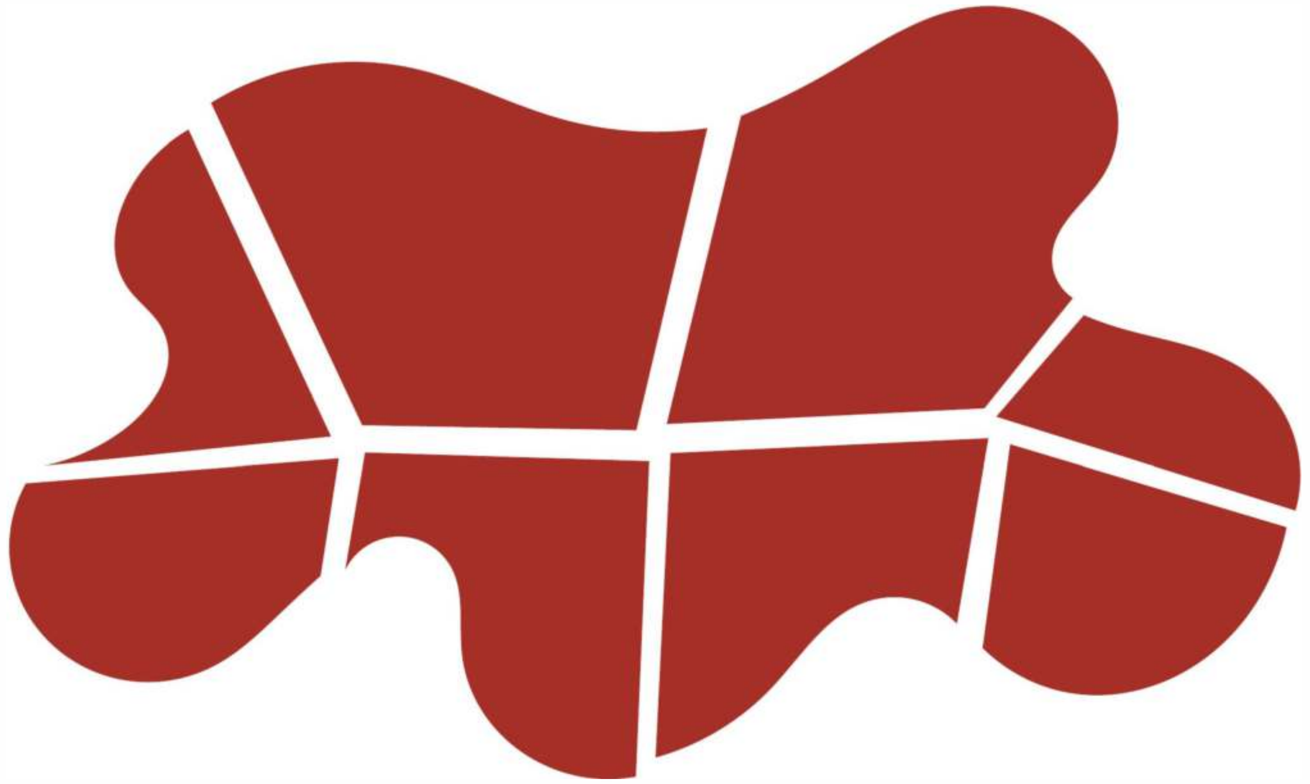
Chunk up the data...

Distributed Data-Parallel: High Level Illustration



Chunk up the data...

Distributed Data-Parallel: High Level Illustration



Distribute it over your cluster of machines.

Distributed Data-Parallel: High Level Illustration

Distribute it over your cluster of machines.

Distributed Data-Parallel: High Level Illustration

From there, think of your distributed data like a single collection...

```scala
val wiki: RDD[WikiArticle] = ...
```

Example:

Transform the text (not titles) of all wiki articles to lowercase.

```scala
wiki.map {
  article => article.text.toLowerCase
}
```