MapReduce
MapReduce

MapReduce is a framework for batch processing of Big Data.
MapReduce

MapReduce is a *framework* for batch processing of Big Data.

What does that mean?
MapReduce

MapReduce is a *framework* for batch processing of Big Data.

What does that mean?
• **Framework**: A system used by programmers to build applications.
MapReduce

MapReduce is a *framework* for batch processing of Big Data.

What does that mean?

- **Framework**: A system used by programmers to build applications.
- **Batch processing**: All the data is available at the outset, and results aren't used until processing completes.
MapReduce

MapReduce is a *framework* for batch processing of Big Data.

What does that mean?

- **Framework**: A system used by programmers to build applications.
- **Batch processing**: All the data is available at the outset, and results aren't used until processing completes.
- **Big Data**: A buzzword used to describe data sets so large that they reveal facts about the world via statistical analysis.
MapReduce

MapReduce is a framework for batch processing of Big Data.

What does that mean?

- **Framework**: A system used by programmers to build applications.
- **Batch processing**: All the data is available at the outset, and results aren't used until processing completes.
- **Big Data**: A buzzword used to describe data sets so large that they reveal facts about the world via statistical analysis.

The MapReduce idea:
MapReduce

MapReduce is a *framework* for batch processing of Big Data.

What does that mean?

- **Framework**: A system used by programmers to build applications.
- **Batch processing**: All the data is available at the outset, and results aren't used until processing completes.
- **Big Data**: A buzzword used to describe data sets so large that they reveal facts about the world via statistical analysis.

The MapReduce idea:

- Data sets are too big to be analyzed by one machine.
MapReduce

MapReduce is a framework for batch processing of Big Data.

What does that mean?

- **Framework**: A system used by programmers to build applications.
- **Batch processing**: All the data is available at the outset, and results aren't used until processing completes.
- **Big Data**: A buzzword used to describe data sets so large that they reveal facts about the world via statistical analysis.

The MapReduce idea:

- Data sets are too big to be analyzed by one machine.
- When using multiple machines, systems issues abound.
MapReduce

MapReduce is a framework for batch processing of Big Data.

What does that mean?

- **Framework**: A system used by programmers to build applications.
- **Batch processing**: All the data is available at the outset, and results aren't used until processing completes.
- **Big Data**: A buzzword used to describe data sets so large that they reveal facts about the world via statistical analysis.

The MapReduce idea:

- Data sets are too big to be analyzed by one machine.
- When using multiple machines, systems issues abound.
- Pure functions enable an abstraction barrier between data processing logic and distributed system administration.
MapReduce

MapReduce is a framework for batch processing of Big Data.

What does that mean?

• **Framework:** A system used by programmers to build applications.

• **Batch processing:** All the data is available at the outset, and results aren't used until processing completes.

• **Big Data:** A buzzword used to describe data sets so large that they reveal facts about the world via statistical analysis.

The MapReduce idea:

• Data sets are too big to be analyzed by one machine.

• When using multiple machines, systems issues abound.

• Pure functions enable an abstraction barrier between data processing logic and distributed system administration.

   (Demo)
Systems
Systems research enables the development of applications by defining and implementing abstractions:
Systems

Systems research enables the development of applications by defining and implementing abstractions:

- **Operating systems** provide a stable, consistent interface to unreliable, inconsistent hardware.
Systems research enables the development of applications by defining and implementing abstractions:

- **Operating systems** provide a stable, consistent interface to unreliable, inconsistent hardware.

- **Networks** provide a simple, robust data transfer interface to constantly evolving communications infrastructure.
Systems

Systems research enables the development of applications by defining and implementing abstractions:

- **Operating systems** provide a stable, consistent interface to unreliable, inconsistent hardware.

- **Networks** provide a simple, robust data transfer interface to constantly evolving communications infrastructure.

- **Databases** provide a declarative interface to software that stores and retrieves information efficiently.
Systems research enables the development of applications by defining and implementing abstractions:

- **Operating systems** provide a stable, consistent interface to unreliable, inconsistent hardware.

- **Networks** provide a simple, robust data transfer interface to constantly evolving communications infrastructure.

- **Databases** provide a declarative interface to software that stores and retrieves information efficiently.

- **Distributed systems** provide a single-entity-level interface to a cluster of multiple machines.
Systems research enables the development of applications by defining and implementing abstractions:

- **Operating systems** provide a stable, consistent interface to unreliable, inconsistent hardware.

- **Networks** provide a simple, robust data transfer interface to constantly evolving communications infrastructure.

- **Databases** provide a declarative interface to software that stores and retrieves information efficiently.

- **Distributed systems** provide a single-entity-level interface to a cluster of multiple machines.

A unifying property of effective systems:
Systems research enables the development of applications by defining and implementing abstractions:

- **Operating systems** provide a stable, consistent interface to unreliable, inconsistent hardware.

- **Networks** provide a simple, robust data transfer interface to constantly evolving communications infrastructure.

- **Databases** provide a declarative interface to software that stores and retrieves information efficiently.

- **Distributed systems** provide a single-entity-level interface to a cluster of multiple machines.

A unifying property of effective systems:

Hide *complexity*, but retain *flexibility*
The Unix Operating System
The Unix Operating System

Essential features of the Unix operating system (and variants):
The Unix Operating System

Essential features of the Unix operating system (and variants):

- **Portability**: The same operating system on different hardware.
The Unix Operating System

Essential features of the Unix operating system (and variants):
• **Portability**: The same operating system on different hardware.
• **Multi-Tasking**: Many processes run concurrently on a machine.
The Unix Operating System

Essential features of the Unix operating system (and variants):
- **Portability**: The same operating system on different hardware.
- **Multi-Tasking**: Many processes run concurrently on a machine.
- **Plain Text**: Data is stored and shared in text format.
The Unix Operating System

Essential features of the Unix operating system (and variants):

- **Portability:** The same operating system on different hardware.
- **Multi-Tasking:** Many processes run concurrently on a machine.
- **Plain Text:** Data is stored and shared in text format.
- **Modularity:** Small tools are composed flexibly via pipes.
The Unix Operating System

Essential features of the Unix operating system (and variants):
• **Portability**: The same operating system on different hardware.
• **Multi-Tasking**: Many processes run concurrently on a machine.
• **Plain Text**: Data is stored and shared in text format.
• **Modularity**: Small tools are composed flexibly via pipes.
The Unix Operating System

Essential features of the Unix operating system (and variants):

- **Portability:** The same operating system on different hardware.
- **Multi-Tasking:** Many processes run concurrently on a machine.
- **Plain Text:** Data is stored and shared in text format.
- **Modularity:** Small tools are composed flexibly via pipes.
The Unix Operating System

Essential features of the Unix operating system (and variants):

- **Portability**: The same operating system on different hardware.
- **Multi-Tasking**: Many processes run concurrently on a machine.
- **Plain Text**: Data is stored and shared in text format.
- **Modularity**: Small tools are composed flexibly via pipes.
The Unix Operating System

Essential features of the Unix operating system (and variants):

- **Portability**: The same operating system on different hardware.
- **Multi-Tasking**: Many processes run concurrently on a machine.
- **Plain Text**: Data is stored and shared in text format.
- **Modularity**: Small tools are composed flexibly via pipes.

```
standard input ➔ process ➔ standard output

Text input
```
The Unix Operating System

Essential features of the Unix operating system (and variants):
• **Portability**: The same operating system on different hardware.
• **Multi-Tasking**: Many processes run concurrently on a machine.
• **Plain Text**: Data is stored and shared in text format.
• **Modularity**: Small tools are composed flexibly via pipes.

standard input → process → standard output

Text input

Text output
Essential features of the Unix operating system (and variants):

- **Portability**: The same operating system on different hardware.
- **Multi-Tasking**: Many processes run concurrently on a machine.
- **Plain Text**: Data is stored and shared in text format.
- **Modularity**: Small tools are composed flexibly via pipes.
The **Unix Operating System**

Essential features of the Unix operating system (and variants):

- **Portability**: The same operating system on different hardware.
- **Multi-Tasking**: Many processes run concurrently on a machine.
- **Plain Text**: Data is stored and shared in text format.
- **Modularity**: Small tools are composed flexibly via pipes.

The **standard streams** in a Unix-like operating system are conceptually similar to Python iterators.
The Unix Operating System

Essential features of the Unix operating system (and variants):

- **Portability**: The same operating system on different hardware.
- **Multi-Tasking**: Many processes run concurrently on a machine.
- **Plain Text**: Data is stored and shared in text format.
- **Modularity**: Small tools are composed flexibly via pipes.

The *standard streams* in a Unix-like operating system are conceptually similar to Python iterators.

(Demo)
Python Programs in a Unix Environment
Python Programs in a Unix Environment

The built-in \texttt{input} function reads a line from \textit{standard input}.
Python Programs in a Unix Environment

The built-in `input` function reads a line from *standard input*. The built-in `print` function writes a line to *standard output*. 
Python Programs in a Unix Environment

The built-in `input` function reads a line from *standard input*. The built-in `print` function writes a line to *standard output*.

(Demo)
Python Programs in a Unix Environment

The built-in `input` function reads a line from *standard input*. The built-in `print` function writes a line to *standard output*.

(Demo)

The values `sys.stdin` and `sys.stdout` also provide access to the Unix *standard streams* as "files."
Python Programs in a Unix Environment

The built-in `input` function reads a line from *standard input*. 
The built-in `print` function writes a line to *standard output*.

(Demo)

The values `sys.stdin` and `sys.stdout` also provide access to the Unix *standard streams* as "files."

A Python "file" is an interface that supports iteration, read, and write methods.
The built-in `input` function reads a line from *standard input*.
The built-in `print` function writes a line to *standard output*.

(Demo)

The values `sys.stdin` and `sys.stdout` also provide access to the Unix *standard streams* as "files."

A Python "file" is an interface that supports iteration, read, and write methods.

Using these "files" takes advantage of the operating system *standard stream* abstraction.
Python Programs in a Unix Environment

The built-in `input` function reads a line from `standard input`.

The built-in `print` function writes a line to `standard output`.

(Demo)

The values `sys.stdin` and `sys.stdout` also provide access to the Unix `standard streams` as "files."

A Python "file" is an interface that supports iteration, read, and write methods.

Using these "files" takes advantage of the operating system `standard stream abstraction`.

(Demo)
MapReduce Evaluation Model
MapReduce Evaluation Model

**Map phase:** Apply a *mapper* function to inputs, emitting a set of *intermediate key-value pairs.*
MapReduce Evaluation Model

**Map phase:** Apply a *mapper* function to inputs, emitting a set of *intermediate key-value pairs*.

- The *mapper* takes an iterator over inputs, such as text lines.
MapReduce Evaluation Model

**Map phase:** Apply a *mapper* function to inputs, emitting a set of *intermediate key-value pairs*.

- The *mapper* takes an iterator over inputs, such as text lines.
- The *mapper* yields zero or more *key-value pairs* per input.
MapReduce Evaluation Model

Map phase: Apply a mapper function to inputs, emitting a set of intermediate key-value pairs.

- The mapper takes an iterator over inputs, such as text lines.
- The mapper yields zero or more key-value pairs per input.

Google MapReduce
Is a Big Data framework
For batch processing
MapReduce Evaluation Model

**Map phase:** Apply a *mapper* function to inputs, emitting a set of **intermediate key-value pairs**.

- The *mapper* takes an iterator over inputs, such as text lines.
- The *mapper* yields zero or more **key-value pairs** per input.

Google MapReduce
Is a Big Data framework
For batch processing
Map phase: Apply a **mapper** function to inputs, emitting a set of **intermediate key-value pairs**.
- The **mapper** takes an iterator over inputs, such as text lines.
- The **mapper** yields zero or more **key-value pairs** per input.

Google MapReduce
Is a Big Data framework
For batch processing
MapReduce Evaluation Model

**Map phase:** Apply a *mapper* function to inputs, emitting a set of *intermediate key-value pairs*.

- The *mapper* takes an iterator over inputs, such as text lines.
- The *mapper* yields zero or more *key-value pairs* per input.

![Diagram showing mapper function and data processing]

Google MapReduce
Is a Big Data framework
For batch processing

```
<table>
<thead>
<tr>
<th>key</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>2</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>u</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
</tr>
</tbody>
</table>
```
MapReduce Evaluation Model

**Map phase:** Apply a *mapper* function to inputs, emitting a set of *intermediate key-value pairs*.

- The *mapper* takes an iterator over inputs, such as text lines.
- The *mapper* yields zero or more *key-value pairs* per input.

Google MapReduce

*Is a Big Data framework*

For batch processing

```
| o: 2 |
| a: 1 |
| u: 1 |
| e: 3 |
```

```
| i: 1 |
| a: 4 |
| e: 1 |
| o: 1 |
```
MapReduce Evaluation Model

**Map phase:** Apply a *mapper* function to inputs, emitting a set of intermediate key-value pairs.

- The *mapper* takes an iterator over inputs, such as text lines.
- The *mapper* yields zero or more key-value pairs per input.

Google MapReduce
Is a Big Data framework
For batch processing

mapper

<table>
<thead>
<tr>
<th>o: 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a: 1</td>
</tr>
<tr>
<td>u: 1</td>
</tr>
<tr>
<td>e: 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>i: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>a: 4</td>
</tr>
<tr>
<td>e: 1</td>
</tr>
<tr>
<td>o: 1</td>
</tr>
</tbody>
</table>

| a: 1 |
| o: 2 |
| e: 1 |
| i: 1 |
MapReduce Evaluation Model

**Map phase:** Apply a *mapper* function to inputs, emitting a set of *intermediate key-value pairs*.

- The *mapper* takes an iterator over inputs, such as text lines.
- The *mapper* yields zero or more *key-value pairs* per input.

Google MapReduce
Is a Big Data framework
For batch processing
MapReduce Evaluation Model

**Map phase:** Apply a *mapper* function to inputs, emitting a set of *intermediate key-value pairs*.

* The *mapper* takes an iterator over inputs, such as text lines.
* The *mapper* yields zero or more *key-value pairs* per input.

Google MapReduce
Is a Big Data framework
For batch processing

**Reduce phase:** For each *intermediate key*, apply a *reducer* function to accumulate all values associated with that key.
MapReduce Evaluation Model

**Map phase:** Apply a *mapper* function to inputs, emitting a set of *intermediate key–value pairs*.

- The *mapper* takes an iterator over inputs, such as text lines.
- The *mapper* yields zero or more *key–value pairs* per input.

Google MapReduce
Is a Big Data framework
For batch processing

![Diagram](image)

**Reduce phase:** For each *intermediate key*, apply a *reducer* function to accumulate all values associated with that key.

- The *reducer* takes an iterator over *key–value pairs*.
MapReduce Evaluation Model

**Map phase**: Apply a *mapper* function to inputs, emitting a set of *intermediate key-value pairs*.

- The *mapper* takes an iterator over inputs, such as text lines.
- The *mapper* yields zero or more *key-value pairs* per input.

Google MapReduce
Is a Big Data framework
For batch processing

**Reduce phase**: For each *intermediate key*, apply a *reducer* function to accumulate all values associated with that key.

- The *reducer* takes an iterator over *key-value pairs*.
- All pairs with a given key are consecutive.
MapReduce Evaluation Model

**Map phase:** Apply a *mapper* function to inputs, emitting a set of *intermediate key-value pairs*.

- The *mapper* takes an iterator over inputs, such as text lines.
- The *mapper* yields zero or more *key-value pairs* per input.

Google MapReduce
Is a Big Data framework
For batch processing

**Reduce phase:** For each *intermediate key*, apply a *reducer* function to accumulate all values associated with that key.

- The *reducer* takes an iterator over *key-value pairs*.
- All pairs with a given key are consecutive.
- The *reducer* yields 0 or more values, each associated with that *intermediate key*.
MapReduce Evaluation Model

Google MapReduce
Is a Big Data framework
For batch processing

**Reduce phase:** For each **intermediate key**, apply a **reducer** function to accumulate all values associated with that key.

- The **reducer** takes an iterator over **key-value pairs**.
- All pairs with a given key are consecutive.
- The **reducer** yields 0 or more values, each associated with that **intermediate key**.
MapReduce Evaluation Model

Google MapReduce
Is a Big Data framework
For batch processing

Reduce phase: For each intermediate key, apply a reducer function to accumulate all values associated with that key.

• The reducer takes an iterator over key-value pairs.
• All pairs with a given key are consecutive.
• The reducer yields 0 or more values, each associated with that intermediate key.

```
a: 4
a: 1
a: 1
e: 1
e: 3
e: 1
...
```
Google MapReduce
Is a Big Data framework
For batch processing

Reduce phase: For each intermediate key, apply a reducer function to accumulate all values associated with that key.

- The reducer takes an iterator over key-value pairs.
- All pairs with a given key are consecutive.
- The reducer yields 0 or more values, each associated with that intermediate key.
Google MapReduce is a Big Data framework for batch processing.

**Reduce phase:** For each intermediate key, apply a reducer function to accumulate all values associated with that key.

- The reducer takes an iterator over key-value pairs.
- All pairs with a given key are consecutive.
- The reducer yields 0 or more values, each associated with that intermediate key.
MapReduce Evaluation Model

Google MapReduce
Is a Big Data framework
For batch processing

Reduce phase: For each intermediate key, apply a reducer function to accumulate all values associated with that key.

• The reducer takes an iterator over key-value pairs.
• All pairs with a given key are consecutive.
• The reducer yields 0 or more values, each associated with that intermediate key.
Google MapReduce
Is a Big Data framework
For batch processing

Reduce phase: For each intermediate key, apply a reducer function to accumulate all values associated with that key.

• The reducer takes an iterator over key-value pairs.
• All pairs with a given key are consecutive.
• The reducer yields 0 or more values, each associated with that intermediate key.
Reduce phase: For each intermediate key, apply a reducer function to accumulate all values associated with that key.

- The reducer takes an iterator over key-value pairs.
- All pairs with a given key are consecutive.
- The reducer yields 0 or more values, each associated with that intermediate key.
Above-the-Line: Execution model

Input

Intermediate

Group by Key

Grouped

Output

Below-the-Line: Parallel Execution

http://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0008.html
A "task" is a Unix process running on a machine.

http://research.google.com/archive/mapreduce-osdi04-slides/index-auto-0008.html
A "task" is a Unix process running on a machine
MapReduce Assumptions
MapReduce Assumptions

**Constraints** on the *mapper* and *reducer*:
MapReduce Assumptions

**Constraints** on the *mapper* and *reducer*:

- The *mapper* must be equivalent to applying a pure function to each input independently.
MapReduce Assumptions

**Constraints** on the *mapper* and *reducer*:

- The *mapper* must be equivalent to applying a pure function to each input independently.
- The *reducer* must be equivalent to applying a pure function to the sequence of values for a key.
MapReduce Assumptions

**Constraints** on the *mapper* and *reducer*:

- The *mapper* must be equivalent to applying a pure function to each input independently.
- The *reducer* must be equivalent to applying a pure function to the sequence of values for a key.

**Benefits** of functional programming:
MapReduce Assumptions

Constraints on the mapper and reducer:

- The mapper must be equivalent to applying a pure function to each input independently.
- The reducer must be equivalent to applying a pure function to the sequence of values for a key.

Benefits of functional programming:

- When a program contains only pure functions, call expressions can be evaluated in any order, lazily, and in parallel.
MapReduce Assumptions

**Constraints** on the *mapper* and *reducer*:

- The *mapper* must be equivalent to applying a pure function to each input independently.
- The *reducer* must be equivalent to applying a pure function to the sequence of values for a key.

**Benefits** of functional programming:

- When a program contains only pure functions, call expressions can be evaluated in any order, lazily, and in parallel.
- Referential transparency: a call expression can be replaced by its value (or *vis versa*) without changing the program.
MapReduce Assumptions

**Constraints** on the *mapper* and *reducer*:

- The *mapper* must be equivalent to applying a pure function to each input independently.
- The *reducer* must be equivalent to applying a pure function to the sequence of values for a key.

**Benefits** of functional programming:

- When a program contains only pure functions, call expressions can be evaluated in any order, lazily, and in parallel.
- Referential transparency: a call expression can be replaced by its value (or *vis versa*) without changing the program.

In MapReduce, these functional programming ideas allow:
MapReduce Assumptions

**Constraints** on the mapper and reducer:
- The mapper must be equivalent to applying a pure function to each input independently.
- The reducer must be equivalent to applying a pure function to the sequence of values for a key.

**Benefits** of functional programming:
- When a program contains only pure functions, call expressions can be evaluated in any order, lazily, and in parallel.
- Referential transparency: a call expression can be replaced by its value (or *vis versa*) without changing the program.

In MapReduce, these functional programming ideas allow:
- Consistent results, however computation is partitioned.
MapReduce Assumptions

**Constraints** on the *mapper* and *reducer*:

- The *mapper* must be equivalent to applying a pure function to each input independently.
- The *reducer* must be equivalent to applying a pure function to the sequence of values for a key.

**Benefits** of functional programming:

- When a program contains only pure functions, call expressions can be evaluated in any order, lazily, and in parallel.
- Referential transparency: a call expression can be replaced by its value (or *vis versa*) without changing the program.

In MapReduce, these functional programming ideas allow:

- Consistent results, however computation is partitioned.
- Re-computation and caching of results, as needed.
Python Example of a MapReduce Application
Python Example of a MapReduce Application

The *mapper* and *reducer* are both self-contained Python programs.
Python Example of a MapReduce Application

The mapper and reducer are both self-contained Python programs.

• Read from standard input and write to standard output!
Python Example of a MapReduce Application

The *mapper* and *reducer* are both self-contained Python programs.

- Read from *standard input* and write to *standard output*!

  **Mapper**
Python Example of a MapReduce Application

The mapper and reducer are both self-contained Python programs.

- Read from standard input and write to standard output!

Mapper

```python
def emit_vowels(line):
    for vowel in 'aeiou':
        count = line.count(vowel)
        if count > 0:
            emit(vowel, count)
```
The mapper and reducer are both self-contained Python programs.

- Read from standard input and write to standard output!

**Mapper**

```python
#!/usr/bin/env python3

import sys
from ucb import main
from mapreduce import emit

def emit_vowels(line):
    for vowel in 'aeiou':
        count = line.count(vowel)
        if count > 0:
            emit(vowel, count)
```
Python Example of a MapReduce Application

The mapper and reducer are both self-contained Python programs.  
• Read from standard input and write to standard output!

```python
#!/usr/bin/env python3

import sys
from ucb import main
from mapreduce import emit

def emit_vowels(line):
    for vowel in 'aeiou':
        count = line.count(vowel)
        if count > 0:
            emit(vowel, count)
```

The mapper and reducer are both self-contained Python programs.

- Read from standard input and write to standard output!

```python
#!/usr/bin/env python3
import sys
from ucb import main
from mapreduce import emit

def emit_vowels(line):
    for vowel in 'aeiou':
        count = line.count(vowel)
        if count > 0:
            emit(vowel, count)
```

Mapper

Tell Unix: this is Python

The emit function outputs a key and value as a line of text to standard output.
Python Example of a MapReduce Application

The mapper and reducer are both self-contained Python programs.

- Read from standard input and write to standard output!

```python
#!/usr/bin/env python3

import sys
from ucb import main
from mapreduce import emit

def emit_vowels(line):
    for vowel in 'aeiou':
        count = line.count(vowel)
        if count > 0:
            emit(vowel, count)

for line in sys.stdin:
    emit_vowels(line)
```

Mapper

Tell Unix: this is Python

The emit function outputs a key and value as a line of text to standard output.
Python Example of a MapReduce Application

The mapper and reducer are both self-contained Python programs.

- Read from standard input and write to standard output!

```python
#!/usr/bin/env python3
import sys
from ucb import main
from mapreduce import emit

def emit_vowels(line):
    for vowel in 'aeiou':
        count = line.count(vowel)
        if count > 0:
            emit(vowel, count)

for line in sys.stdin:
    emit_vowels(line)
```

Mapper

Tell Unix: this is Python

The emit function outputs a key and value as a line of text to standard output

Mapper inputs are lines of text provided to standard input
Python Example of a MapReduce Application

The mapper and reducer are both self-contained Python programs.

- Read from standard input and write to standard output!
The *mapper* and *reducer* are both self-contained Python programs.

- Read from *standard input* and write to *standard output*!

**Reducer**

```python
#!/usr/bin/env python3

import sys
from ucb import main
from mapreduce import emit, group_values_by_key
```
Python Example of a MapReduce Application

The *mapper* and *reducer* are both self-contained Python programs. 
- Read from *standard input* and write to *standard output*!

**Reducer**

```python
#!/usr/bin/env python3

import sys
from ucb import main
from mapreduce import emit, group_values_by_key
```

Takes and returns iterators
Python Example of a MapReduce Application

The mapper and reducer are both self-contained Python programs.

- Read from standard input and write to standard output!

Reducer

#!/usr/bin/env python3

import sys
from ucb import main
from mapreduce import emit, group_values_by_key

Takes and returns iterators

Input: lines of text representing key-value pairs, grouped by key

Output: Iterator over (key, value_iterator) pairs that give all values for each key
Python Example of a MapReduce Application

The mapper and reducer are both self-contained Python programs.
- Read from standard input and write to standard output!

Reducer

#!/usr/bin/env python3

import sys
from ucb import main
from mapreduce import emit, group_values_by_key

for key, value_iterator in group_values_by_key(sys.stdin):
    emit(key, sum(value_iterator))

Takes and returns iterators

Input: lines of text representing key-value pairs, grouped by key
Output: Iterator over (key, value_iterator) pairs that give all values for each key
What Does the MapReduce Framework Provide
What Does the MapReduce Framework Provide

**Fault tolerance:** A machine or hard drive might crash.
What Does the MapReduce Framework Provide

**Fault tolerance:** A machine or hard drive might crash.
- The MapReduce framework automatically re-runs failed tasks.
What Does the MapReduce Framework Provide

**Fault tolerance:** A machine or hard drive might crash.  
• The MapReduce framework automatically re-runs failed tasks.

**Speed:** Some machine might be slow because it's overloaded.
What Does the MapReduce Framework Provide

**Fault tolerance:** A machine or hard drive might crash.
- The MapReduce framework automatically re-runs failed tasks.

**Speed:** Some machine might be slow because it's overloaded.
- The framework can run multiple copies of a task and keep the result of the one that finishes first.
What Does the MapReduce Framework Provide

**Fault tolerance:** A machine or hard drive might crash.  
- The MapReduce framework automatically re-runs failed tasks.

**Speed:** Some machine might be slow because it's overloaded.  
- The framework can run multiple copies of a task and keep the result of the one that finishes first.

**Network locality:** Data transfer is expensive.
What Does the MapReduce Framework Provide

**Fault tolerance:** A machine or hard drive might crash.
• The MapReduce framework automatically re-runs failed tasks.

**Speed:** Some machine might be slow because it's overloaded.
• The framework can run multiple copies of a task and keep the result of the one that finishes first.

**Network locality:** Data transfer is expensive.
• The framework tries to schedule map tasks on the machines that hold the data to be processed.
What Does the MapReduce Framework Provide

**Fault tolerance:** A machine or hard drive might crash.
- The MapReduce framework automatically re-runs failed tasks.

**Speed:** Some machine might be slow because it's overloaded.
- The framework can run multiple copies of a task and keep the result of the one that finishes first.

**Network locality:** Data transfer is expensive.
- The framework tries to schedule map tasks on the machines that hold the data to be processed.

**Monitoring:** Will my job finish before dinner?!?
What Does the MapReduce Framework Provide

**Fault tolerance:** A machine or hard drive might crash.
- The MapReduce framework automatically re-runs failed tasks.

**Speed:** Some machine might be slow because it's overloaded.
- The framework can run multiple copies of a task and keep the result of the one that finishes first.

**Network locality:** Data transfer is expensive.
- The framework tries to schedule map tasks on the machines that hold the data to be processed.

**Monitoring:** Will my job finish before dinner?!?
- The framework provides a web-based interface describing jobs.
What Does the MapReduce Framework Provide

**Fault tolerance:** A machine or hard drive might crash.
- The MapReduce framework automatically re-runs failed tasks.

**Speed:** Some machine might be slow because it's overloaded.
- The framework can run multiple copies of a task and keep the result of the one that finishes first.

**Network locality:** Data transfer is expensive.
- The framework tries to schedule map tasks on the machines that hold the data to be processed.

**Monitoring:** Will my job finish before dinner?!?
- The framework provides a web-based interface describing jobs.

(Demo)