

61A Lecture 36

Monday, April 27

Announcements

Announcements

- Recursive Art Contest Entries due Monday 4/27 @ 11:59pm

Announcements

- Recursive Art Contest Entries due Monday 4/27 @ 11:59pm
 - Email your code & a screenshot of your art to cs61a-tae@imail.eecs.berkeley.edu (Albert)

Announcements

- Recursive Art Contest Entries due Monday 4/27 @ 11:59pm
 - Email your code & a screenshot of your art to cs61a-tae@imail.eecs.berkeley.edu (Albert)
- Homework 9 (4 pts) due Wednesday 4/29 @ 11:59pm

Announcements

- Recursive Art Contest Entries due Monday 4/27 @ 11:59pm
 - Email your code & a screenshot of your art to cs61a-tae@imail.eecs.berkeley.edu (Albert)
- Homework 9 (4 pts) due Wednesday 4/29 @ 11:59pm
 - Homework Party Tuesday 5pm–6:30pm on Tuesday 4/28 in 2050 VLSB

Announcements

- Recursive Art Contest Entries due Monday 4/27 @ 11:59pm
 - Email your code & a screenshot of your art to cs61a-tae@imail.eecs.berkeley.edu (Albert)
- Homework 9 (4 pts) due Wednesday 4/29 @ 11:59pm
 - Homework Party Tuesday 5pm–6:30pm on Tuesday 4/28 in 2050 VLSB
 - Go to lab next week for help on the SQL homework! (There's also a lab.)

Announcements

- Recursive Art Contest Entries due Monday 4/27 @ 11:59pm
 - Email your code & a screenshot of your art to cs61a-tae@imail.eecs.berkeley.edu (Albert)
- Homework 9 (4 pts) due Wednesday 4/29 @ 11:59pm
 - Homework Party Tuesday 5pm–6:30pm on Tuesday 4/28 in 2050 VLSB
 - Go to lab next week for help on the SQL homework! (There's also a lab.)
- Quiz 4 (SQL) released on Tuesday 4/28 is due Thursday 4/30 @ 11:59pm

Unix

Computer Systems

Computer Systems

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

Computer Systems

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

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

Computer 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 robust data transfer interface to constantly evolving communications infrastructure

Computer 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 robust data transfer interface to constantly evolving communications infrastructure
- **Databases** provide a declarative interface to software that stores and retrieves information efficiently

Computer 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 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 unified interface to a cluster of multiple machines

Computer 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 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 unified interface to a cluster of multiple machines

A unifying property of effective systems:

Computer 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 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 unified 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.

“We should have some ways of coupling programs like [a] garden hose – screw in another segment when it becomes necessary to massage data in another way,” Doug McIlroy in 1964.

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.

“We should have some ways of coupling programs like [a] garden hose – screw in another segment when it becomes necessary to massage data in another way,” Doug McIlroy in 1964.

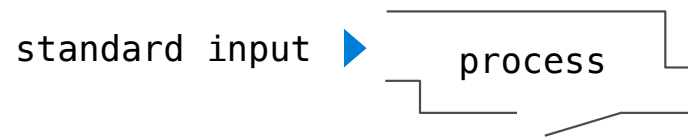


The Unix Operating System

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.

“We should have some ways of coupling programs like [a] garden hose – screw in another segment when it becomes necessary to message data in another way,” Doug McIlroy in 1964.

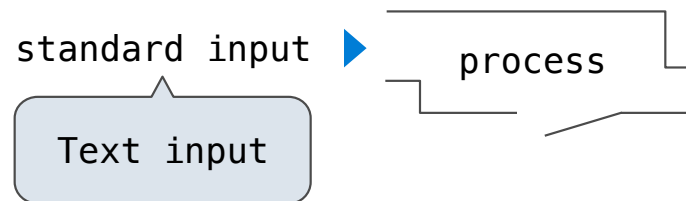


The Unix Operating System

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.

“We should have some ways of coupling programs like [a] garden hose – screw in another segment when it becomes necessary to massage data in another way,” Doug McIlroy in 1964.

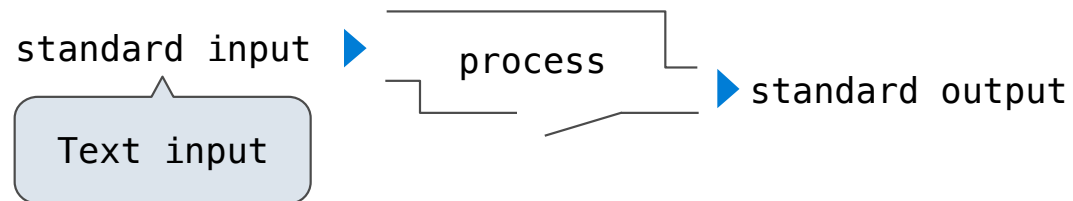


The Unix Operating System

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.

“We should have some ways of coupling programs like [a] garden hose – screw in another segment when it becomes necessary to massage data in another way,” Doug McIlroy in 1964.

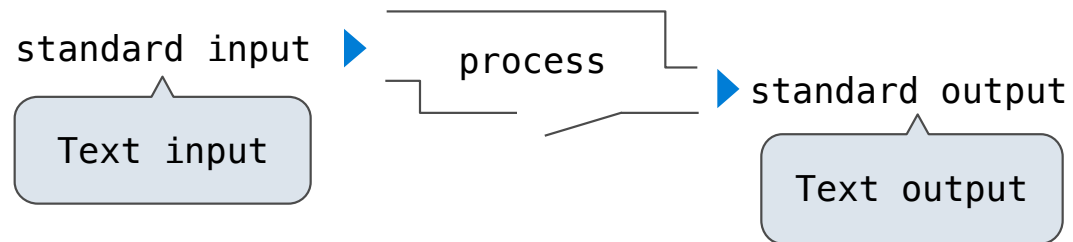


The Unix Operating System

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.

“We should have some ways of coupling programs like [a] garden hose – screw in another segment when it becomes necessary to massage data in another way,” Doug McIlroy in 1964.

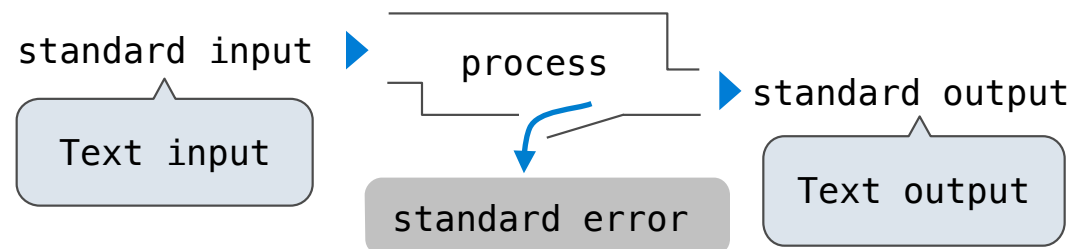


The Unix Operating System

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.

“We should have some ways of coupling programs like [a] garden hose – screw in another segment when it becomes necessary to massage data in another way,” Doug McIlroy in 1964.

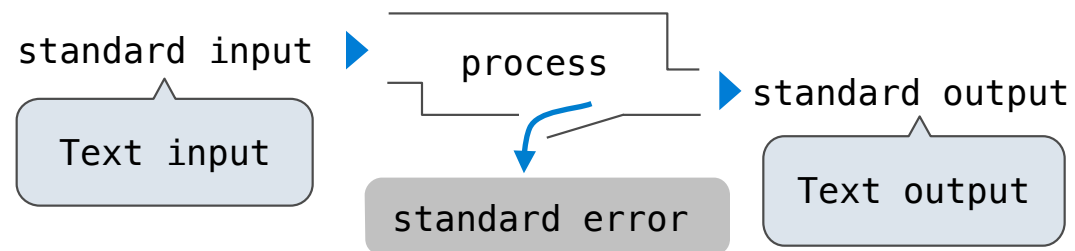


The Unix Operating System

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.

“We should have some ways of coupling programs like [a] garden hose – screw in another segment when it becomes necessary to massage data in another way,” Doug McIlroy in 1964.



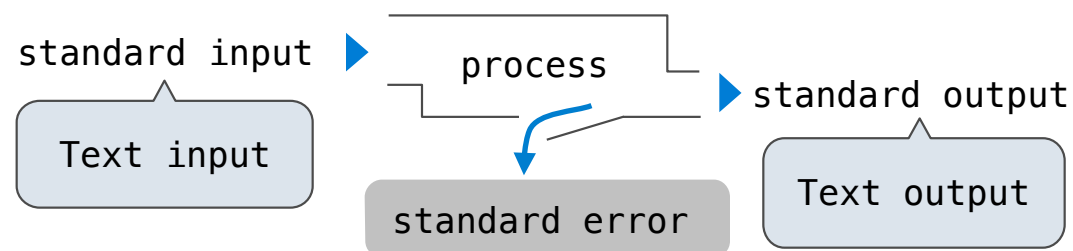
The standard streams in a Unix-like operating system are similar to Python iterators.

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.

“We should have some ways of coupling programs like [a] garden hose – screw in another segment when it becomes necessary to massage data in another way,” Doug McIlroy in 1964.



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

(Demo)

```
ls hw* | grep -v html | cut -f 1 -d '.' | cut -c 3- | sort -n
```


Python Programs in a Unix Environment

Python Programs in a Unix Environment

The built-in `input` function reads a line from 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 `sys.stdin` and `sys.stdout` values 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 `sys.stdin` and `sys.stdout` values provide access to the Unix standard streams as files

A Python file has an interface that supports iteration, `read`, and `write` methods

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

A Python file has an interface that supports iteration, `read`, and `write` methods

Using these "files" takes advantage of the operating system text processing 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 `sys.stdin` and `sys.stdout` values provide access to the Unix standard streams as files

A Python file has an interface that supports iteration, `read`, and `write` methods

Using these "files" takes advantage of the operating system text processing abstraction

(Demo)

MapReduce

Big Data Processing

Big Data Processing

MapReduce is a framework for batch processing of big data.

Big Data Processing

MapReduce is a framework for batch processing of big data.

- **Framework:** A system used by programmers to build applications

Big Data Processing

MapReduce is a framework for batch processing of big data.

- **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 Processing

MapReduce is a framework for batch processing of big data.

- **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:** Used to describe data sets so large and comprehensive that they can reveal facts about a whole population, usually from statistical analysis

Big Data Processing

MapReduce is a framework for batch processing of big data.

- **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:** Used to describe data sets so large and comprehensive that they can reveal facts about a whole population, usually from statistical analysis

The MapReduce idea:

Big Data Processing

MapReduce is a framework for batch processing of big data.

- **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:** Used to describe data sets so large and comprehensive that they can reveal facts about a whole population, usually from statistical analysis

The MapReduce idea:

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

Big Data Processing

MapReduce is a framework for batch processing of big data.

- **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:** Used to describe data sets so large and comprehensive that they can reveal facts about a whole population, usually from statistical analysis

The MapReduce idea:

- Data sets are too big to be analyzed by one machine
- Using multiple machines has the same complications, regardless of the application/analysis

Big Data Processing

MapReduce is a framework for batch processing of big data.

- **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:** Used to describe data sets so large and comprehensive that they can reveal facts about a whole population, usually from statistical analysis

The MapReduce idea:

- Data sets are too big to be analyzed by one machine
- Using multiple machines has the same complications, regardless of the application/analysis
- Pure functions enable an abstraction barrier between data processing logic and coordinating a distributed application

Big Data Processing

MapReduce is a framework for batch processing of big data.

- **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:** Used to describe data sets so large and comprehensive that they can reveal facts about a whole population, usually from statistical analysis

The MapReduce idea:

- Data sets are too big to be analyzed by one machine
- Using multiple machines has the same complications, regardless of the application/analysis
- Pure functions enable an abstraction barrier between data processing logic and coordinating a distributed application

(Demo)

MapReduce Evaluation Model

MapReduce Evaluation Model

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

MapReduce Evaluation Model

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

- The mapper takes an iterable value containing inputs, such as lines of text

MapReduce Evaluation Model

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

- The mapper takes an iterable value containing inputs, such as lines of text
- The mapper yields zero or more key-value pairs for each input

MapReduce Evaluation Model

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

- The mapper takes an iterable value containing inputs, such as lines of text
- The mapper yields zero or more key-value pairs for each input

Google MapReduce

Is a Big Data framework

For batch processing

MapReduce Evaluation Model

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

- The mapper takes an iterable value containing inputs, such as lines of text
- The mapper yields zero or more key-value pairs for each input

Google MapReduce
Is a Big Data framework
For batch processing



MapReduce Evaluation Model

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

- The mapper takes an iterable value containing inputs, such as lines of text
- The mapper yields zero or more key-value pairs for each input



MapReduce Evaluation Model

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

- The mapper takes an iterable value containing inputs, such as lines of text
- The mapper yields zero or more key-value pairs for each input

Google MapReduce

Is a Big Data framework
For batch processing



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

MapReduce Evaluation Model

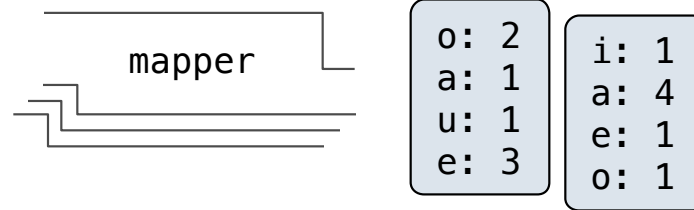
Map phase: Apply a *mapper* function to all inputs, emitting intermediate key-value pairs

- The mapper takes an iterable value containing inputs, such as lines of text
- The mapper yields zero or more key-value pairs for each input

Google MapReduce

Is a Big Data framework

For batch processing



MapReduce Evaluation Model

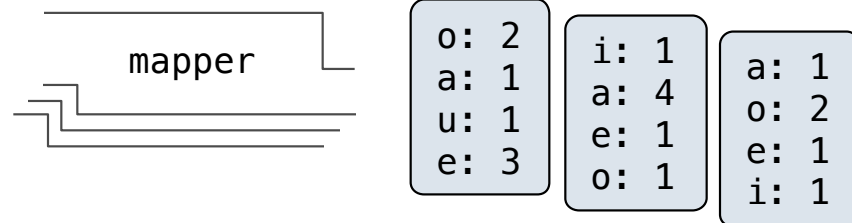
Map phase: Apply a *mapper* function to all inputs, emitting intermediate key-value pairs

- The mapper takes an iterable value containing inputs, such as lines of text
- The mapper yields zero or more key-value pairs for each input

Google MapReduce

Is a Big Data framework

For batch processing

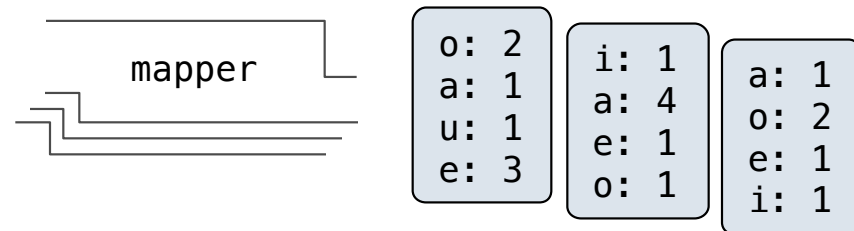


MapReduce Evaluation Model

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

- The mapper takes an iterable value containing inputs, such as lines of text
- The mapper yields zero or more key-value pairs for each input

Google MapReduce
Is a Big Data framework
For batch processing

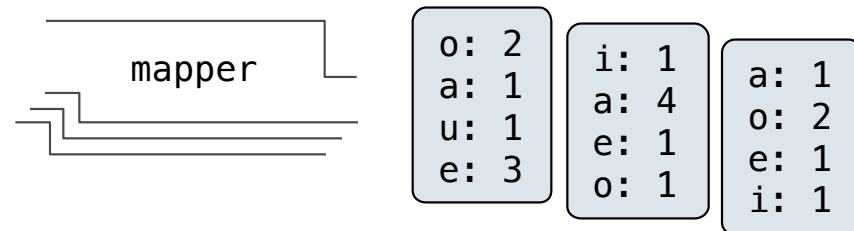


MapReduce Evaluation Model

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

- The mapper takes an iterable value containing inputs, such as lines of text
- The mapper yields zero or more key-value pairs for each input

Google MapReduce
Is a Big Data framework
For batch processing



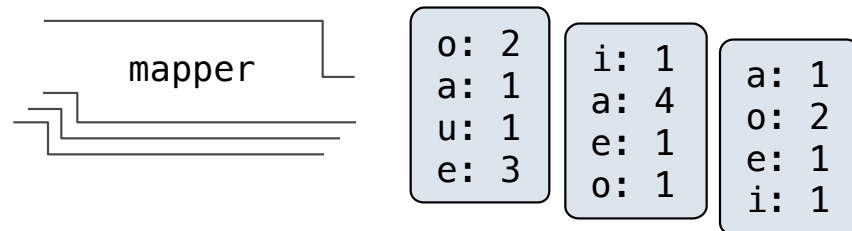
Reduce phase: For each intermediate key, apply a *reducer* function to accumulate all values associated with that key

MapReduce Evaluation Model

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

- The mapper takes an iterable value containing inputs, such as lines of text
- The mapper yields zero or more key-value pairs for each 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

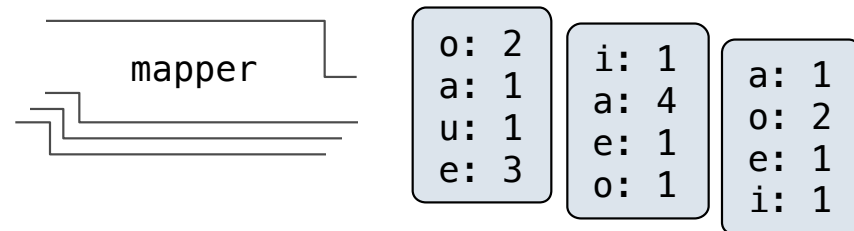
- The reducer takes an iterable value containing intermediate key-value pairs

MapReduce Evaluation Model

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

- The mapper takes an iterable value containing inputs, such as lines of text
- The mapper yields zero or more key-value pairs for each 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

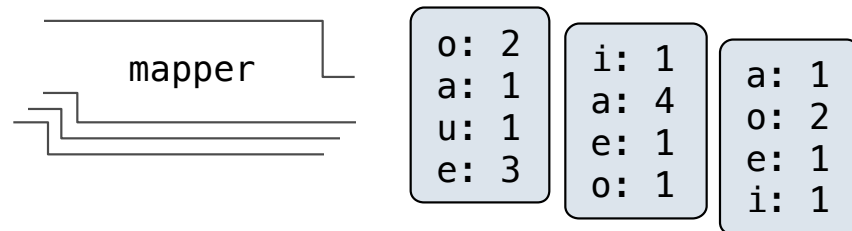
- The reducer takes an iterable value containing intermediate key-value pairs
- All pairs with the same key appear consecutively

MapReduce Evaluation Model

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

- The mapper takes an iterable value containing inputs, such as lines of text
- The mapper yields zero or more key-value pairs for each input

Google MapReduce
Is a Big Data framework
For batch processing

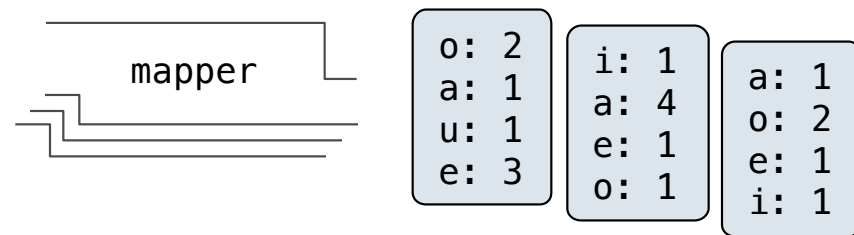


Reduce phase: For each intermediate key, apply a *reducer* function to accumulate all values associated with that key

- The reducer takes an iterable value containing intermediate key-value pairs
- All pairs with the same key appear consecutively
- The reducer yields zero or more values, each associated with that intermediate key

MapReduce Evaluation Model

Google MapReduce
Is a Big Data framework
For batch processing

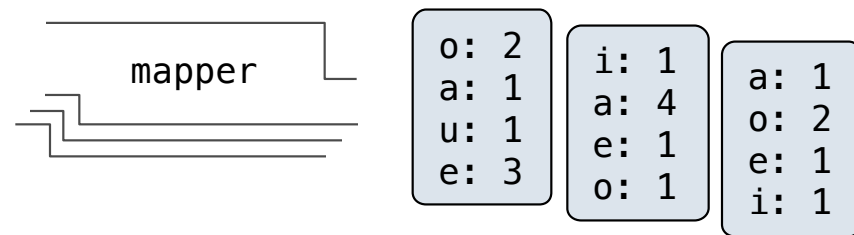


Reduce phase: For each intermediate key, apply a *reducer* function to accumulate all values associated with that key

- The reducer takes an iterable value containing intermediate key-value pairs
- All pairs with the same key appear consecutively
- The reducer yields zero or more values, each associated with that intermediate key

MapReduce Evaluation Model

Google MapReduce
Is a Big Data framework
For batch processing



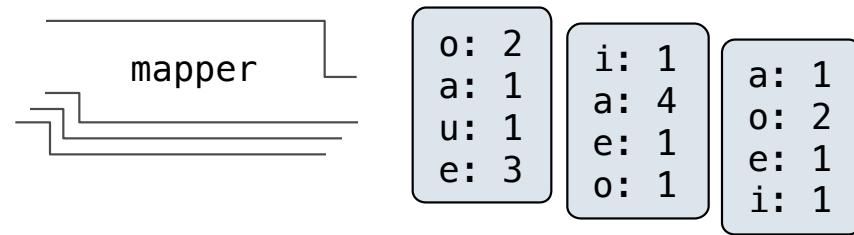
Reduce phase: For each intermediate key, apply a *reducer* function to accumulate all values associated with that key

- The reducer takes an iterable value containing intermediate key-value pairs
- All pairs with the same key appear consecutively
- The reducer yields zero or more values, each associated with that intermediate key

```
a: 4
a: 1
a: 1
e: 1
e: 3
e: 1
...
```

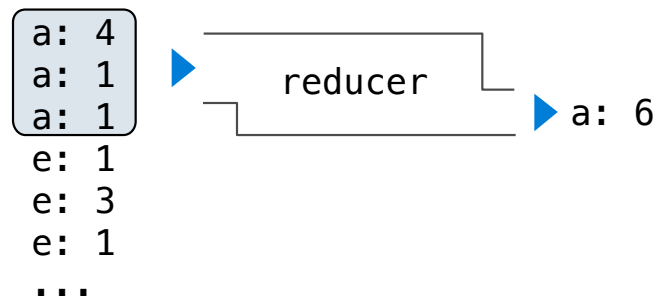
MapReduce Evaluation Model

Google MapReduce
Is a Big Data framework
For batch processing



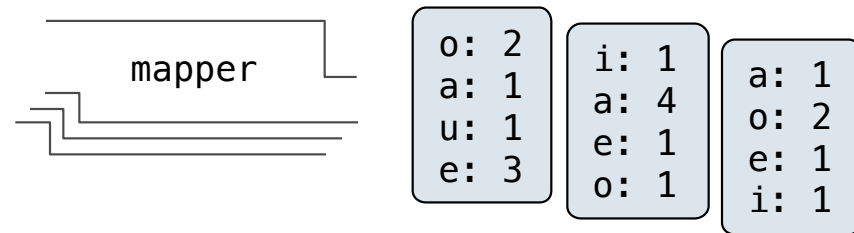
Reduce phase: For each intermediate key, apply a *reducer* function to accumulate all values associated with that key

- The reducer takes an iterable value containing intermediate key-value pairs
- All pairs with the same key appear consecutively
- The reducer yields zero or more values, each associated with that intermediate key



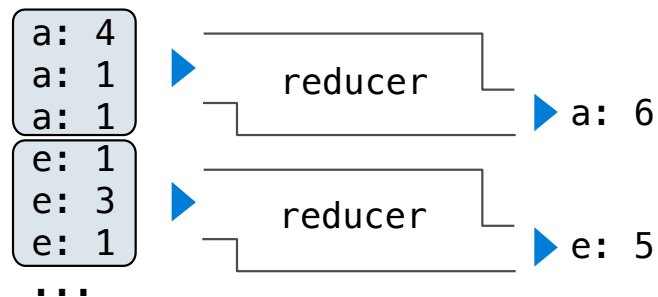
MapReduce Evaluation Model

Google MapReduce
Is a Big Data framework
For batch processing



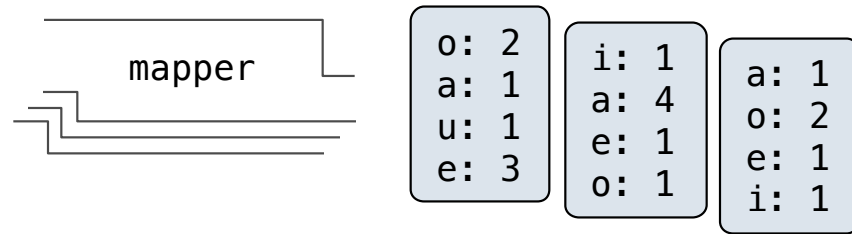
Reduce phase: For each intermediate key, apply a *reducer* function to accumulate all values associated with that key

- The reducer takes an iterable value containing intermediate key-value pairs
- All pairs with the same key appear consecutively
- The reducer yields zero or more values, each associated with that intermediate key



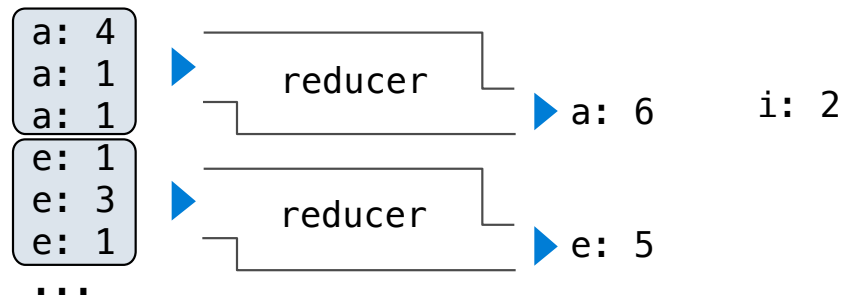
MapReduce Evaluation Model

Google MapReduce
Is a Big Data framework
For batch processing



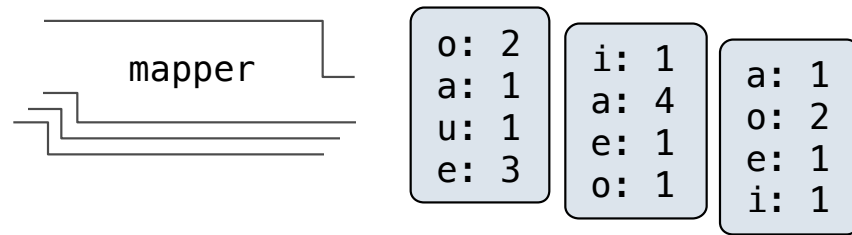
Reduce phase: For each intermediate key, apply a *reducer* function to accumulate all values associated with that key

- The reducer takes an iterable value containing intermediate key-value pairs
- All pairs with the same key appear consecutively
- The reducer yields zero or more values, each associated with that intermediate key



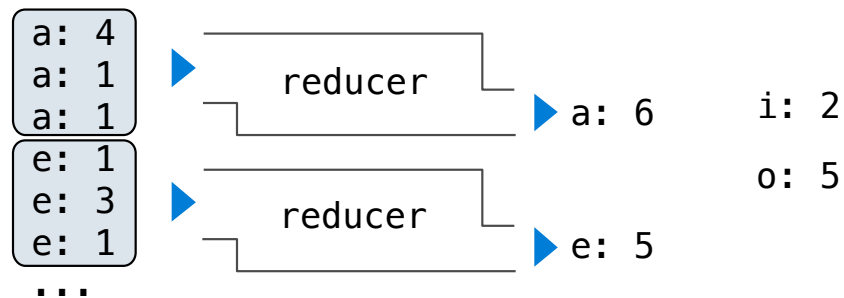
MapReduce Evaluation Model

Google MapReduce
Is a Big Data framework
For batch processing



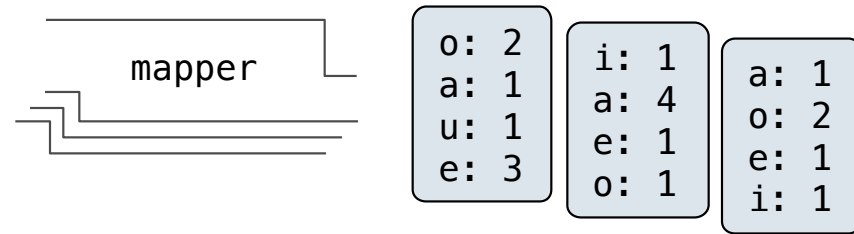
Reduce phase: For each intermediate key, apply a *reducer* function to accumulate all values associated with that key

- The reducer takes an iterable value containing intermediate key-value pairs
- All pairs with the same key appear consecutively
- The reducer yields zero or more values, each associated with that intermediate key



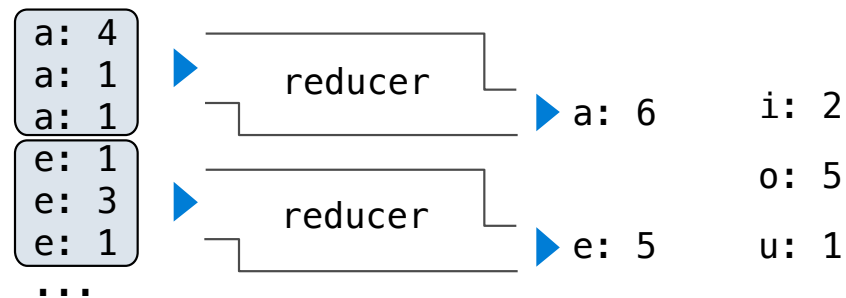
MapReduce Evaluation Model

Google MapReduce
Is a Big Data framework
For batch processing



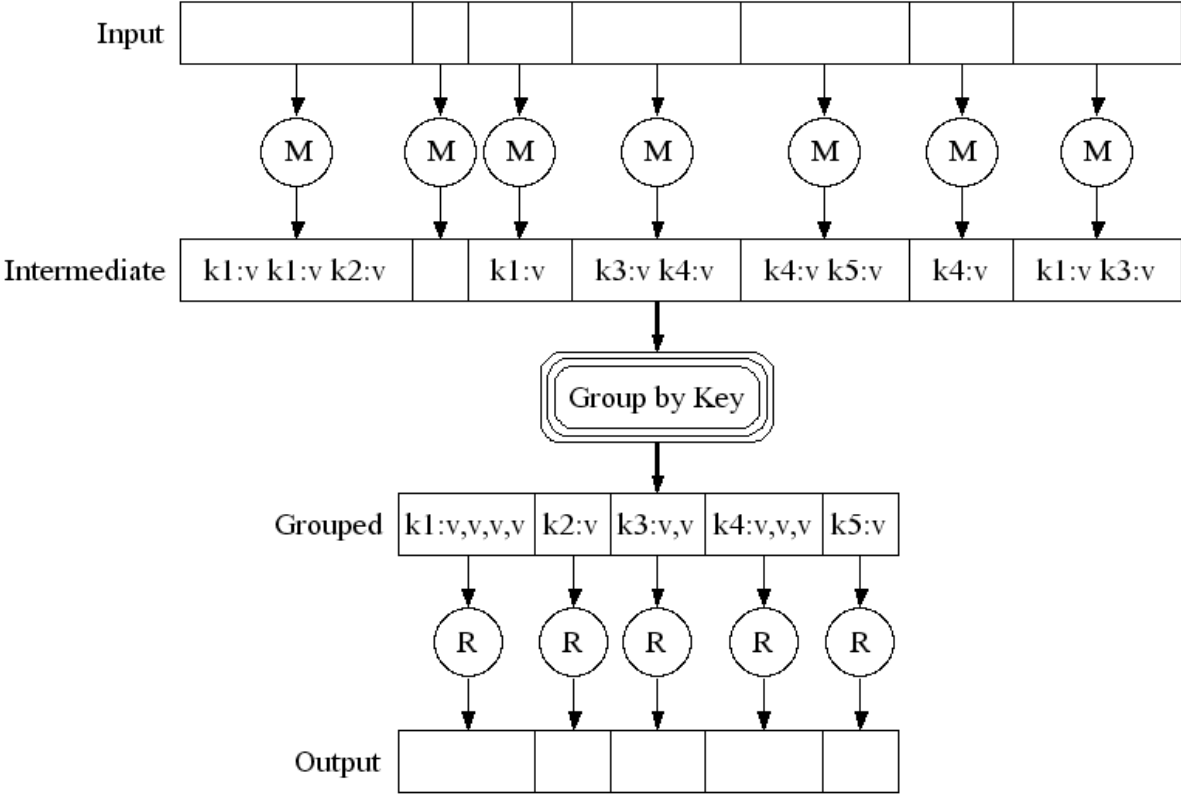
Reduce phase: For each intermediate key, apply a *reducer* function to accumulate all values associated with that key

- The reducer takes an iterable value containing intermediate key-value pairs
- All pairs with the same key appear consecutively
- The reducer yields zero or more values, each associated with that intermediate key

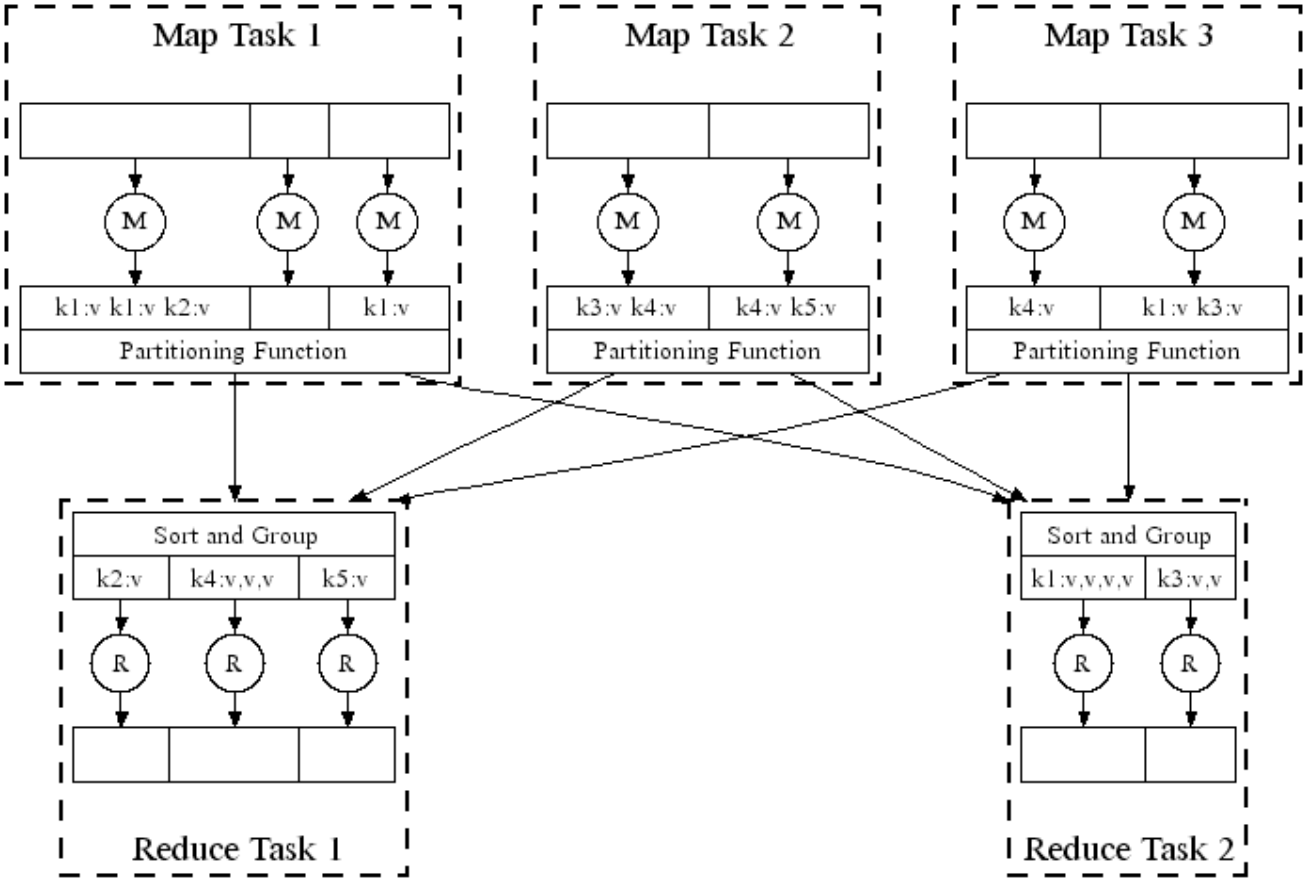


MapReduce Execution Model

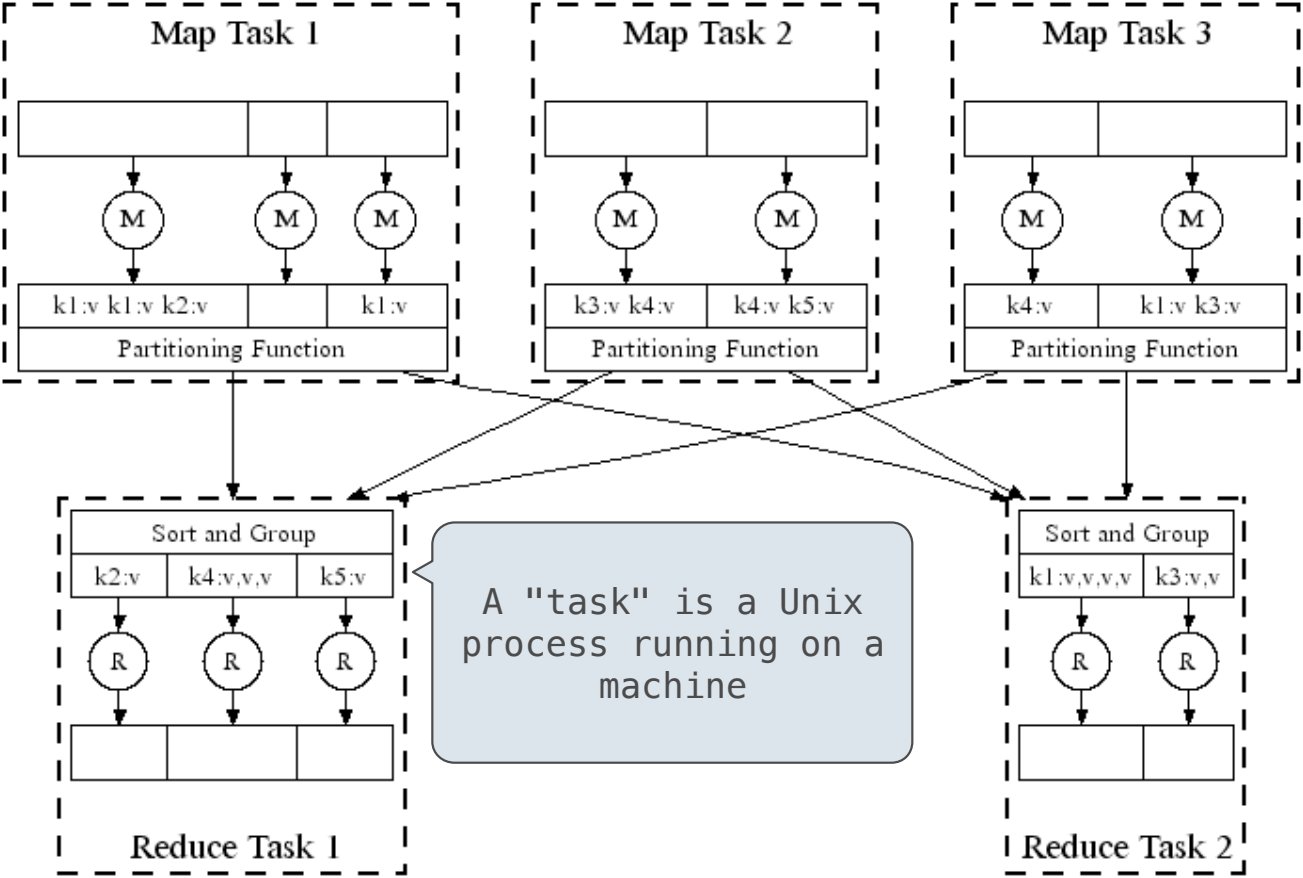
Execution Model



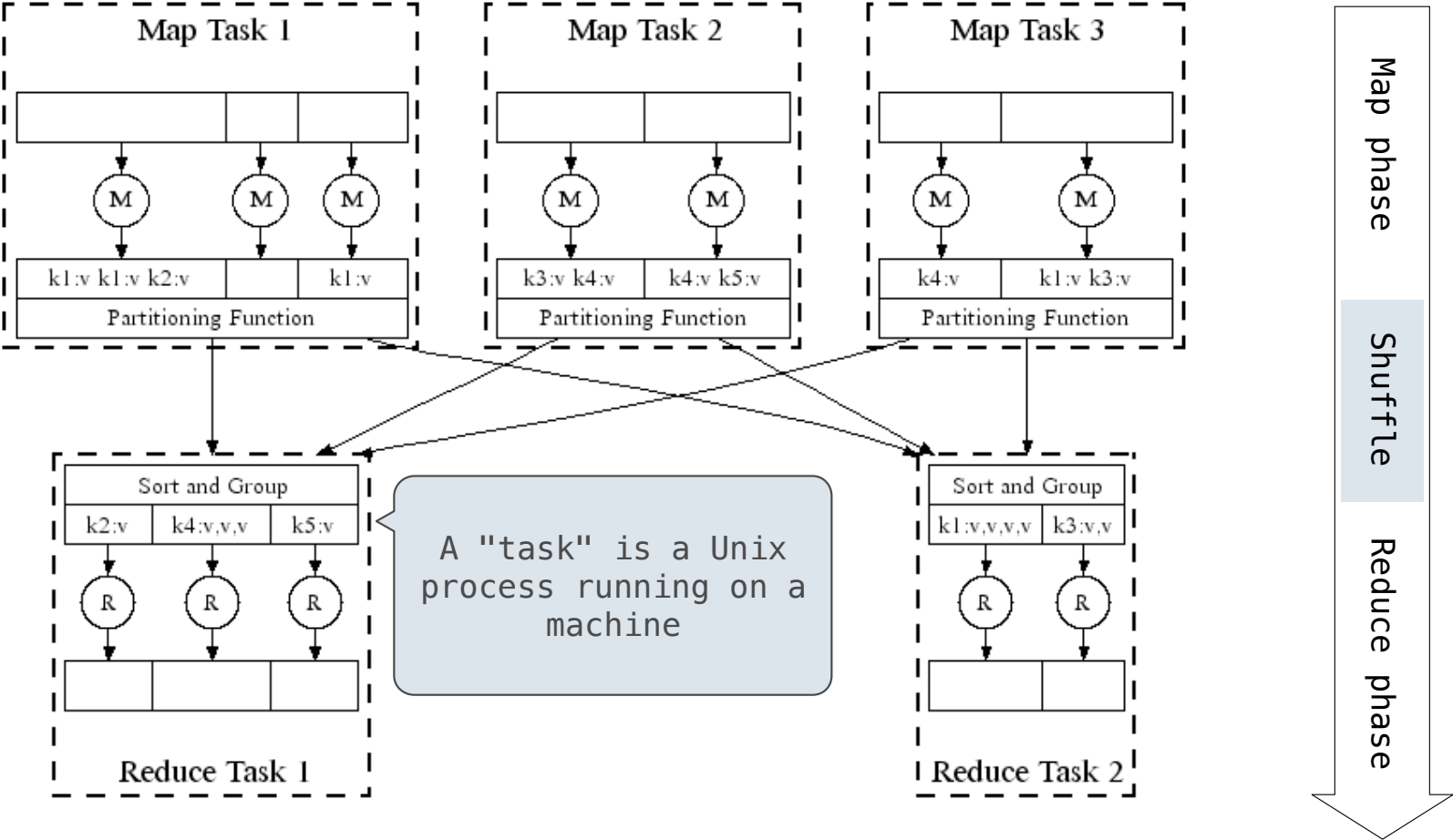
Parallel Execution Implementation



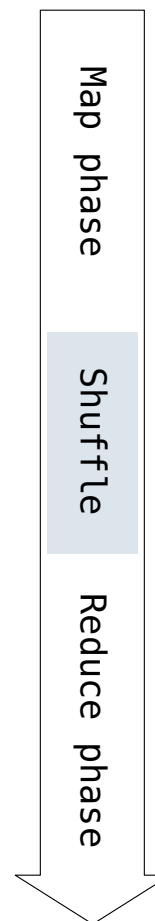
Parallel Execution Implementation



Parallel Execution Implementation

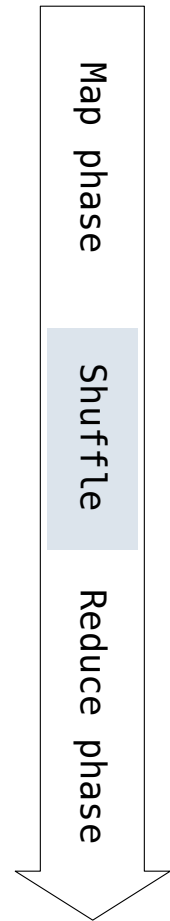


MapReduce Assumptions



MapReduce Assumptions

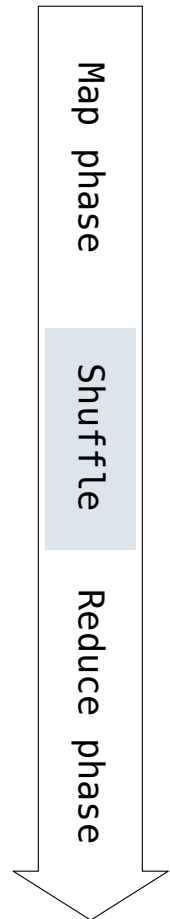
Constraints on the *mapper* and *reducer*:



MapReduce Assumptions

Constraints on the *mapper* and *reducer*:

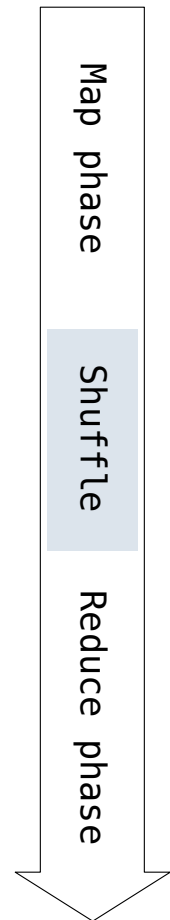
- The mapper must be equivalent to applying a deterministic pure function to each input independently



MapReduce Assumptions

Constraints on the *mapper* and *reducer*:

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

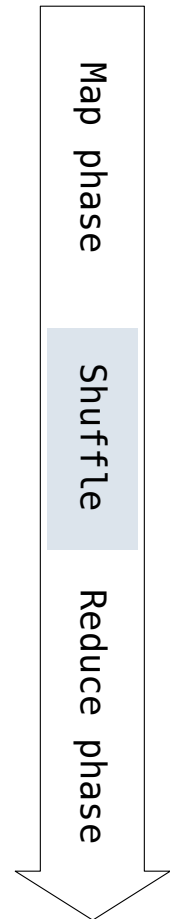


MapReduce Assumptions

Constraints on the *mapper* and *reducer*:

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

Benefits of functional programming:



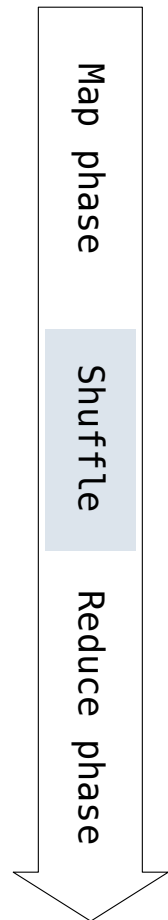
MapReduce Assumptions

Constraints on the *mapper* and *reducer*:

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

Benefits of functional programming:

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



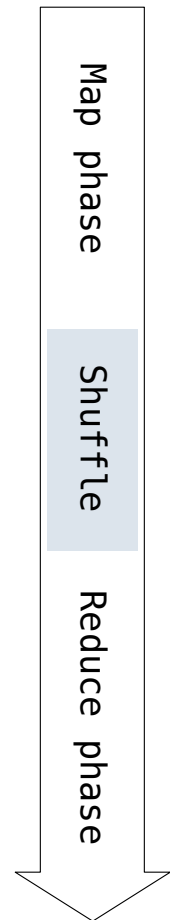
MapReduce Assumptions

Constraints on the *mapper* and *reducer*:

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

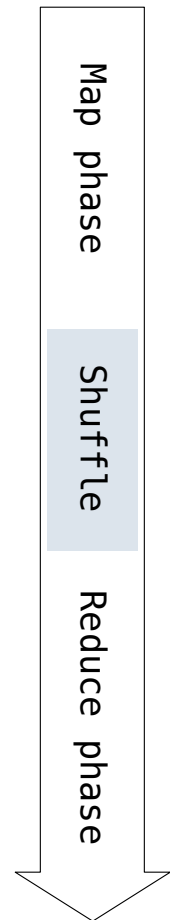
Constraints on the *mapper* and *reducer*:

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

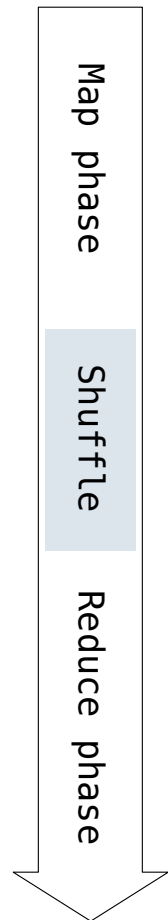
- The mapper must be equivalent to applying a deterministic pure function to each input independently
- The reducer must be equivalent to applying a deterministic pure function to the sequence of values for each key

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*:

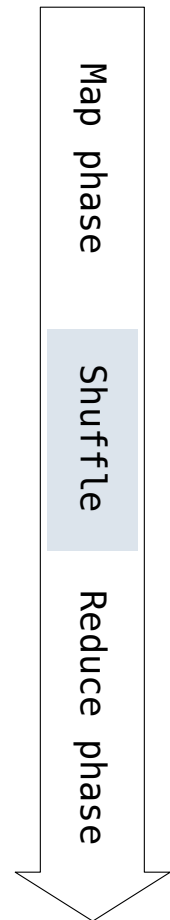
- The mapper must be equivalent to applying a deterministic pure function to each input independently
- The reducer must be equivalent to applying a deterministic pure function to the sequence of values for each key

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



MapReduce Applications

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

- They 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

- They 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

- They read from standard input and write to standard output

Mapper

```
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

- They read from standard input and write to standard output

Mapper

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

import sys
from mr import emit

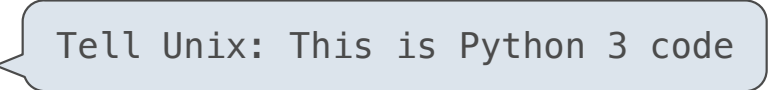
def emit_vowels(line):
    for vowel in 'aeiou':
        count = line.count(vowel)
        if count > 0:
            emit(vowel, count)
```

Python Example of a MapReduce Application

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

- They read from standard input and write to standard output

Mapper

```
#!/usr/bin/env python3   
  
import sys  
from mr 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

- They read from standard input and write to standard output

Mapper

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

Tell Unix: This is Python 3 code

```
import sys
```

```
from mr import emit
```

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

```
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

- They read from standard input and write to standard output

Mapper

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

Tell Unix: This is Python 3 code

```
import sys
```

```
from mr import emit
```

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

```
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)
```

Python Example of a MapReduce Application

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

- They read from standard input and write to standard output

Mapper

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

Tell Unix: This is Python 3 code

```
import sys
```

```
from mr import emit
```

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

```
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 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

- They read from standard input and write to standard output

Mapper

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

Tell Unix: This is Python 3 code

```
import sys
```

```
from mr import emit
```

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

```
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 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

- They read from standard input and write to standard output

Mapper

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

Tell Unix: This is Python 3 code

```
import sys
```

```
from mr import emit
```

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

```
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 inputs are lines of text provided to standard input

(Demo)

Python Example of a MapReduce Application

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

- They read from standard input and write to standard output

Reducer

Python Example of a MapReduce Application

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

- They read from standard input and write to standard output

Reducer

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

import sys
from mr import emit, values_by_key
```


Python Example of a MapReduce Application

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

- They read from standard input and write to standard output

Reducer

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



Takes and returns iterators

```
import sys
from mr import emit, values_by_key
```


Python Example of a MapReduce Application

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

- They read from standard input and write to standard output

Reducer

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

Takes and returns iterators

```
import sys
from mr import emit, values_by_key
```

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

- They read from standard input and write to standard output

Reducer

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

Takes and returns iterators

```
import sys
from mr import emit, values_by_key
```

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

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

Python Example of a MapReduce Application

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

- They read from standard input and write to standard output

Reducer

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

Takes and returns iterators

```
import sys
from mr import emit, values_by_key
```

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

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

(Demo)

MapReduce Benefits

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)