Wallaroo LABS

Stream Processing with Python in Wallaroo

DataEngConf NYC, Oct. 2017

Andrew Turley, Lead Engineer
andy@WallarooLabs.com

Wallaroo Labs

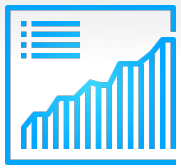
Wallroo Labs is a small startup that making it easy to develop distributed data processing applications

- ★NYC-based
- ★VC funded
- ★Working on Wallaroo for 18 months
- ★Extensive experience working on distributed data processing applications

The Challenge (in Tweets)

Finding the Trending Topics in 1M tweets
is easy to prototype on a laptop
(in between coffees)

…but months to productionize at large-scale



- Earthquake
- iPhone
- Mexico
- UN

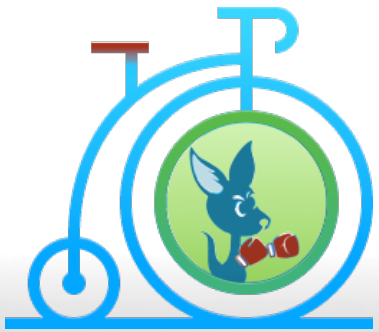
…and then the volume spikes

…and then you add sentiment analysis …and then (aargh!)

Focus on Algorithms, not Infrastructure

Wallaroo is an open-source framework for building and operating fast data, big data, and machine learning applications

- ★ Deploy rapidly, scale automatically, and operate at very low-cost
- ★ Makes the infrastructure virtually disappear
- ★ Native for Python and GoLang, no JVM needed
- ★ Ultrafast and accurate processing



Do What You're Good At

Wallaroo lets you do what you're good at, and takes care of the rest

- ★Wallaroo
  - ★Scaling
  - ★Message Delivery
  - ★Error Recovery
- ★Application Developer
  - ★Business Logic

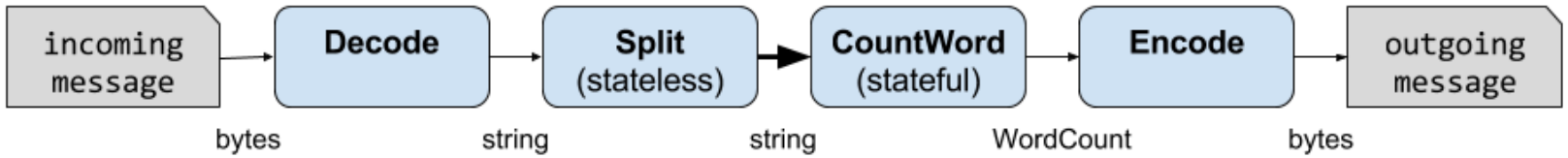
Another Challenge (in Words)

The canonical
WORD COUNT
example

- ★Receive messages that consist of one or more words
- ★Split the words into individual words
- ★Keep a running total of how many times each word occurs
- ★Output a message for each received word with the total occurrences of the word

Let's Count Some Words!

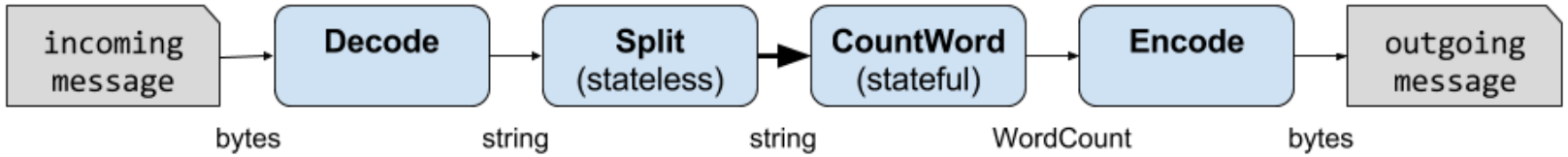
The canonical
WORD COUNT
example



- ★Get a message containing one or more words
- ★Keep a tally of how many times each word has been seen
- ★For each incoming word, output the current tally for that word

Coffee

Let's get some coffee



Word Count

Single machine prototype

```
class WordTotals:
    def __init__(self):
        self.counts = {}

    def update(self, word):
        self.counts[word] = self.counts.get(word, 0) + 1

    def get_count(self, word):
        return WordCount(word, self.counts.get(word, 0))

class WordCount(object):
    def __init__(self, word, count):
        self.word = word
        self.count = count

def decode(bs):
    return bs.encode("utf-8")

def split(words):
    return words.split()

def count_words(word, word_totals):
    word_totals.update(word)
    return word_totals.get_count(word)

def encode(word_count):
    return "{} => {}\n".format(word_count.word, word_count.count)
```

```
import socket

def main():
    word_totals = WordTotals()

    sink = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sink.connect(('127.0.0.1', 7000))

    listener = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    listener.bind(('', 5000))
    listener.listen(1)
    (source, addr) = listener.accept()
    while True:
        raw_words = decode(source.recv(512))
        words = split(raw_words)
        for w in words:
            word_count = count_words(w, word_totals)
            sink.send(encode(word_count))

if __name__ == "__main__":
    main()
```

SINGLE MACHINE PROTOTYPE

Wallaroo Handles Scaling

Run at scale and autoscale

```
class Split(object):
    def name(self):
        return "split into words"

    def compute_multi(self, data):
        punctuation = " !\"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"

        words = []

        for line in data.split("\n"):
            clean_line = line.lower().strip(punctuation)
            for word in clean_line.split(' '):
                clean_word = word.strip(punctuation)
                words.append(clean_word)

        return words

class CountWord(object):
    def name(self):
        return "Count Word"

    def compute(self, word, word_totals):
        word_totals.update(word)
        return (word_totals.get_count(word), True)

class WordTotals(object):
    def __init__(self):
        self.word_totals = {}

    def update(self, word):
        if self.word_totals.has_key(word):
            self.word_totals[word] = self.word_totals[word] + 1
        else:
            self.word_totals[word] = 1

    def get_count(self, word):
        return WordCount(word, self.word_totals[word])
```

```
class WordCount(object):
    def __init__(self, word, count):
        self.word = word
        self.count = count

class WordTotalsBuilder(object):
    def build(self):
        return WordTotals()

class WordPartitionFunction(object):
    def partition(self, data):
        if data[0] >= 'a' or data[0] <= 'z':
          return data[0]
        else:
          return "!"

class Decoder(object):
    def header_length(self):
        return 4

    def payload_length(self, bs):
        return struct.unpack(">I", bs)[0]

    def decode(self, bs):
        return bs.decode("utf-8")

class Encoder(object):
    def encode(self, data):
        output = data.word + " => " + str(data.count) + "\n"
        print output
        return output
```

RUN AT SCALE AND AUTOSCALE

Plain old Python

Wallaroo lets you
describe your business
logic in plain Python

```
class Split(object):
    def name(self):
        return "split into words"

    def compute_multi(self, data):
        punctuation = " !\"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"

        words = []

        for line in data.split("\n"):
            clean_line = line.lower().strip(punctuation)
            for word in clean_line.split(' '):
                clean_word = word.strip(punctuation)
                words.append(clean_word)

        return words
```

Plain old Python

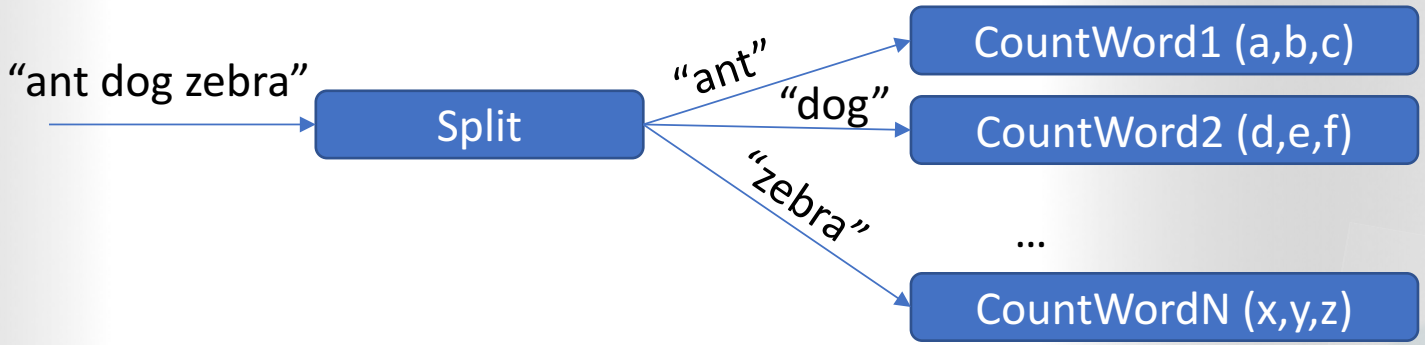
Wallaroo lets you operate on Python objects

```
class CountWord(object):
    def name(self):
        return "Count Word"

    def compute(self, word, word_totals):
        word_totals.update(word)
        return (word_totals.get_count(word), True)


class WordTotals(object):
    def __init__(self):
        self.word_totals = {}

    def update(self, word):
        if self.word_totals.has_key(word):
            self.word_totals[word] = self.word_totals[word] + 1
        else:
            self.word_totals[word] = 1

    def get_count(self, word):
        return WordCount(word, self.word_totals[word])
```

Wallaroo Handles Scaling

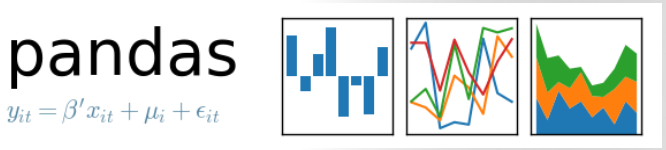
Wallaroo handles scaling for you automatically

```
class WordPartitionFunction(object):
    def partition(self, data):
        if data[0] >= 'a' or data[0] <= 'z':
          return data[0]
        else:
          return "!"
```



Python!

Leverage your existing knowledge, use your favorite libraries



Wallaroo

Wallaroo is a framework for building and operating fast data, big data, and machine learning applications that deploy rapidly, scale automatically and operate at very low-cost

wallaroolabs.com

github.com/wallaroolabs/wallaroo