Reduce: Compute the coefficient and evaluate by substituting for

Map: Generate key-value pair from with key = where

Combine: Collect the “like terms”

Large Set (up to 1,000,000\*):

Split set *S* into smaller subsets and distribute the smaller sets to the map processors (\* the total size takes into account the size of each element)

Final Summation:

Collect all terms and return the sum

= 1

= 1

[(1, v), (1, v)]

[(1, v), (1, v)]

[(0, v)]

**Local Combine:**

[(0, v)]

Reduce:

= 0

Map:

[d, e, f]

Map:

[a, b, c]

Map:

[g, h]

(0, v)

(1, v)

(1, v)

(1, v)

(0, v)

(1, v)

(1, v)

(1, v)

Split Set S:

[a, b, c, d, e, f, g, h]

*x* = 2 w = is\_consonant()

= 1

= 0

[(1, v), (1, v)]

Map:

[a, b, c]

Map:

[d, e, f]

Map:

[ig, h]

(0, v)

(1, v)

(1, v)

(1, v)

(0, v)

(1, v)

(1, v)

(1, v)

**Local Combine:**

[(0, v)]

[(1, v), (1, v)]

[(0, v)]

[(1, v), (1, v)]

[(1, v), (1, v)]

Split Set S:

[a, b, c, d, e, f, g, h]

*x* = 2 w = is\_consonant()

= 1

Reduce:

= 0

Example. Consider the following weight function

Then let S = {‘a’, ’b’, ’c’, ’d’, ’e’, ’f’, ’g’}

At *x* = 2, the sum of the series is (2 + 10 = 12)

Reduce:

= 0

= 1

(1, v)

[(1, v), … , (1, v)]

**Global Combine:**

[(0, v), (0, v)]

(1, v)

Map:

[a, b, c]

Map:

[d, e, f]

Map:

[ig, h]

Split Set S:

[a, b, c, d, e, f, g, h]

*x* = 2 w = is\_consonant()

(0, v)

(1, v)

(1, v)

(1, v)

(0, v)

(1, v)