## **Warehouse Scale Computing**

#### 1. Amdahl's Law

1) You are going to train the image classifier with 50,000 images on a WSC having more than 50,000 servers. You notice that 99% of the execution can be parallelized. What is the speedup?

$$100/(99*(1/50000) + 1) = \sim 100x$$

#### 2. Failure in a WSC

1) In this example, a WSC has 55,000 servers, and each server has four disks whose annual failure rate is 4%. How many disks will fail per hour?

2) What is the availability of the system if it does not tolerate the failure? Assume that the time to repair a disk is 30 minutes. 1/(1 + 0.5) = .66

#### 3. Performance of a WSC

	Local	Rack	Array
DRAM latency (us)	0.1	100	300
Global hit rate	90%	9%	1%
DRAM bandwidth (MiB/sec)	20,000	100	10
Disk bandwidth (MiB/ sec)	200	100	10

<sup>1)</sup> Calculate the AMAT of this WSC. What is vital for WSC performance?

locality of data to disk drives speed

$$.1(10^{-6}) + .1(100^{*}(10^{-6}) + .1(300(10^{-6})))) = 13.1(10^{-6})s$$
  
hmmmm the answer says  $(.9^{*}.1 + .09^{*}100 + .01^{*}300)(10^{-6}) = 12.09(10^{-6})...$ 

2) How long does it take to transfer 1,000 MiB a) between disks within the server, and b) between DRAM within the rack? What can you conclude from this example?

a) 5

b) 10

Keeping thing local is important to keeping things fast.

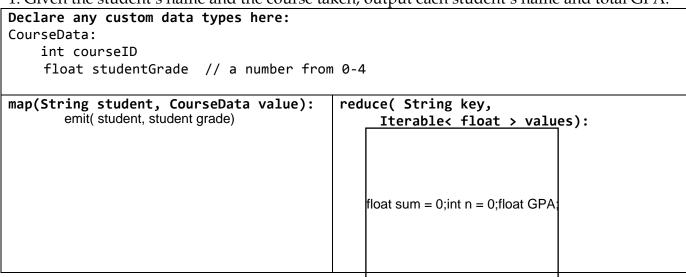
- **4. Power Usage Effectiveness (PUE) =** (Total Building Power) / (IT Equipment Power) Sources speculate Google has over 1 million servers. Assume each of the 1 million servers draw an average of 200W, the PUE is 1.5, and that Google pays an average of 6 cents per kilowatt-hour for datacenter electricity.
- 1) Estimate Google's annual power bill for its datacenters. ~157 M\$
- 2) Google reduced the PUE of a 50,000 machine datacenter from 1.5 to 1.25 without decreasing the power supplied to the servers. What's the cost savings per year?

### Map Reduce

Use pseudocode to write MapReduce functions necessary to solve the problems below. Also, make sure to fill out the correct data types. Some tips:

- The input to each MapReduce job is given by the signature of the map() function.
- The function **emit(key k, value v)** outputs the key-value pair **(k, v)**.
- The **for(var in list)** syntax can be used to iterate through **Iterable**s or you can call the **hasNext()** and **next()** functions.
- Usable data types: **int**, **float**, **String**. You may also use lists and custom data types composed of the aforementioned types.
- The method **intersection(list1, list2)** returns a list that is the intersection of list1 and list2.

1. Given the student's name and the course taken, output each student's name and total GPA.



2. Given a person's unique int ID and a list of the IDs of their friends, compute the list of mutual friends between each pair of friends in a social network.

```
Declare any custom data types here:
FriendPair:
int friendOne
int friendTwo

map(int personID, list<int> friendIDs):
for friendID in friendIDs: emit(pe

for friend in friendIDs: if personID < friendID:

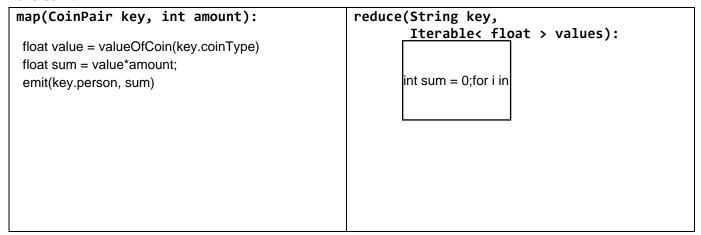
mutualFriends = intersection(friendlists[1], friendlists[2]);
emit(FriendPair key, mutualfriends

mutualFriends = intersection( values.next(), values,next())
```

3. a) Given a set of coins and each coin's owner, compute the number of coins of each denomination that a person has.

Declare any custom data types here: CoinPair: String person String coinType	
<pre>map(String person, String coinType):     CoinPair = (person, coinType)     emit(CoinPair, 1)</pre>	<pre>reduce(CoinPair key,</pre>

b) Using the output of the first MapReduce, compute the amount of money each person has. The function valueOfCoin(String coinType) returns a float corresponding to the dollar value of the coin.



# Spark

- RDD: primary abstraction of a distributed collection of items
- Transforms: RDD → RDD

map(func)	Return a new distributed dataset formed by passing each element of the source through a function <i>func</i> .
<pre>flatMap(func)</pre>	Similar to map, but each input item can be mapped to 0 or more output items (so <i>func</i> should return a Seq rather than a single item).
reduceByKey(func)	When called on a dataset of (K,V) pairs, returns a dataset of (K,V) pairs where the values for each key are aggregated using the given reduce function <i>func</i> , which must be of type (V,V) => V.

• Actions: RDD → Value

reduce(func)	Aggregate the elements of the dataset <i>regardless of keys</i> using a function <i>func</i>
	Turction june

1. Implement Problem 1 of MapReduce with Spark

```
# students: list((studentName, courseData))
studentsData = sc.parallelize(students)
out = studentsData.map(lambda (k, v): (k, (v.studentGrade, ____)))
```

2. Implement Problem 2 of MapReduce with Spark

```
def genFriendPairAndValue(pID, fIDs):
    return [((pID, fID), fIDs) if pID < fID else (fID, pID) for fID in fIDs]
def intersection(l1, l2):
    return [x for x in b1 if x in b2]
# persons: list((personID, list(friendID))
personsData = sc.parallelize(persons)</pre>
```

3. Implement Problem 3 of MapReduce with Spark

```
# coinPairs: list((person, coinType))
coinData = sc.parallelize(coinPairs)
```