

# Distributed Programming Models: MapReduce

Questions Answered in this Lecture:

- How do we abstract away details of parallel programs?
- How does MapReduce work?
- When is it useful?

# Announcements

- P4A due Friday!
- Reading: Google tutorial on MapReduce
  - <https://ai.google/research/pubs/pub36249>

# Acknowledgements

- A lot of this information comes from Google
- Also Paul Krzyzanowski's notes from Rutgers
- See also here

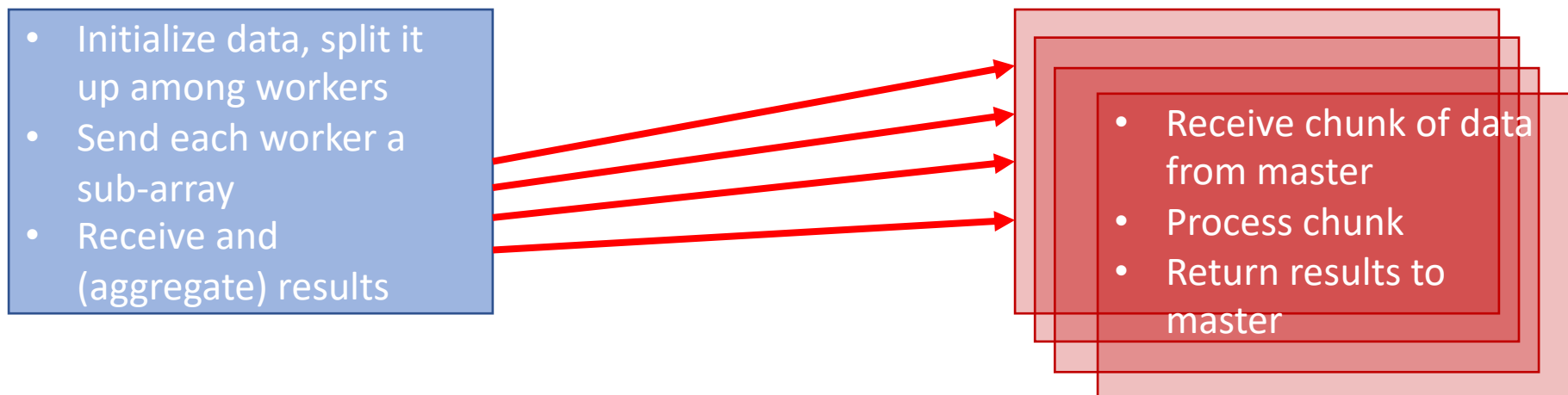
[https://hadoop.apache.org/docs/r1.2.1/mapred\\_tutorial.html](https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html)

# Background

- We've seen a lot about concurrency and parallelism, and why they make sense to use
- But we also know how it can be hard to get right

# Simplest example: *embarrassingly parallel*

- ***No dependencies*** between data
- Split our data into chunks (or shards)
- Distribute chunks among workers (threads, processes, nodes, etc.)
- *Master* process coordinates others (*workers*)



# The Power of Abstraction

- **Many programs work this way** when written with parallelism in mind
- But **they all repeat a lot of work** (creating threads, waiting on barriers, waiting on CV's, getting intermediate results, terminating workers, etc.)
- Can we abstract away the details? (yes)
- We **build a *framework***, and how we *use* (program to the interface of) that framework we'll refer to as a *programming model*

# MapReduce

- Created by Google in 2004
- Inspired by collection-oriented functions in functional languages (e.g. LISP)
  - **map(function, collection)**
    - Applies function to each value in the collection
    - (map 'length '(()) (a) (a b) (a b c))) => (0 1 2 3)
  - **reduce(function, collection)**
    - Combines all values using a binary function (+, \*, etc.)
    - (reduce #' + '(1 2 3 4 5)) => 15

# MapReduce

- Framework for parallel computing (on clusters)
- Programmers use a simple API
- Don't have to worry about
  - Parallelization – Creating threads, managing them
  - Domain decomposition – who gets what data
  - Load Balancing – When/where to schedule workers
  - Fault Tolerance
- Can use it to process tera/petabytes on big machines



# MapReduce

- Map(input chunk) → Intermediate key/value pairs
  - Framework automatically **partitions input data into M chunks**
  - Discard unneeded data and generate (key, value) sets
  - **Framework groups intermediate values** with same key and passes them to *Reduce*
- Reduce(intermediate key/value pairs) → result files
  - Input key & set of values which it maps to
    - **Merge these values together to form a smaller set of values**
- Important: Reducers are distributed by **partitioning key space into R pieces** (using a partitioning function)

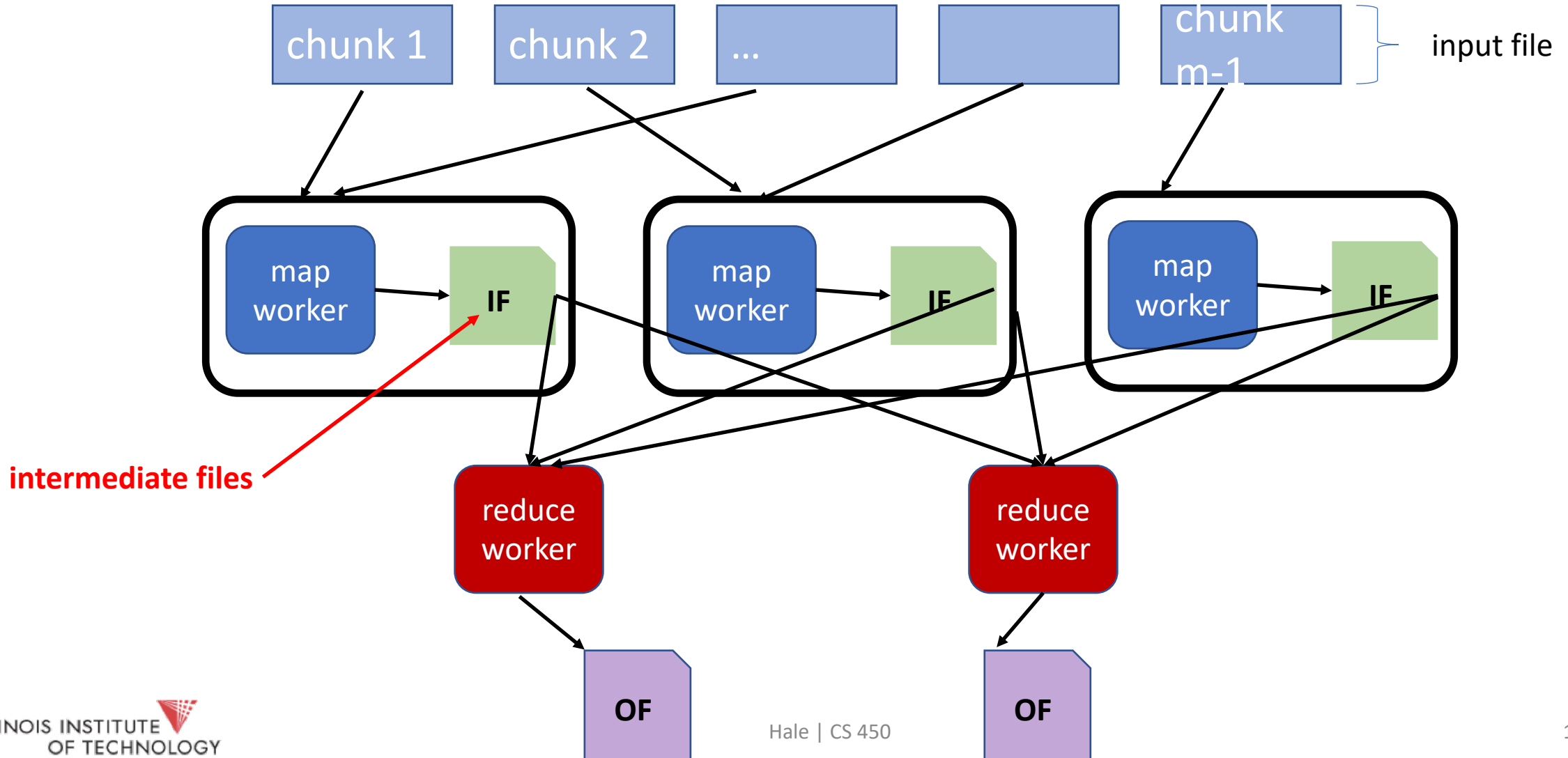
- Map
  - Get source data (parse into key, value pair)
  - Write key to intermediate file (emit)
- Partition
  - Identify which of R reducers will handle which keys
  - Map data to target it to one of the R reducers

**Map Worker**

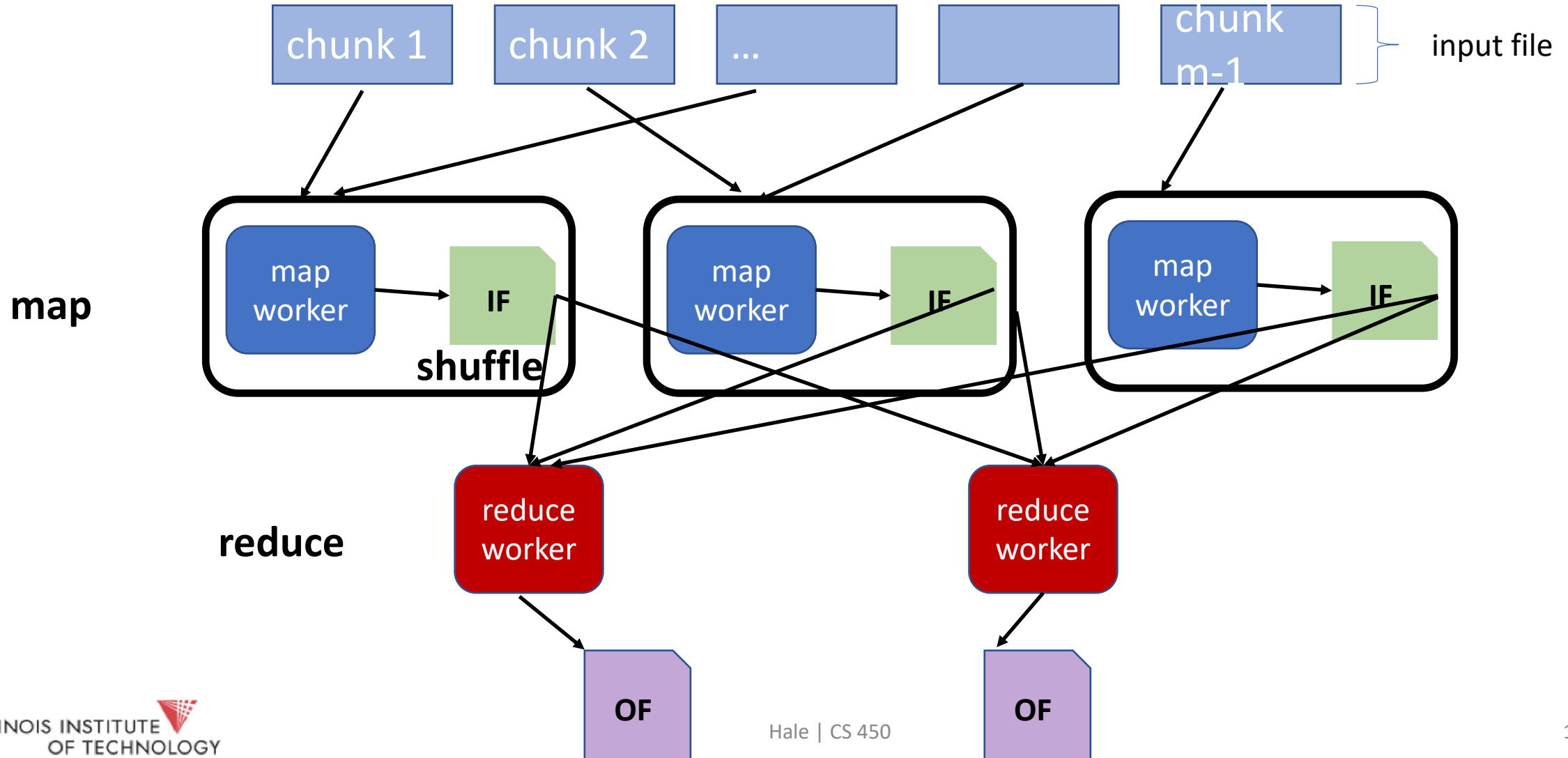
- Shuffle and Sort
  - Shuffle: Fetch the relevant partition of the output from all mappers
  - Sort by keys (note that there may be more than one value for each key)
- Reduce
  - Input is the sorted output of mappers
  - Call Reduce function per key so that reducer can aggregate all values

**Reduce Worker**

# The complete picture



# The complete picture



# Example: word count

- Map
  - Process data, output each word and a count (1 each occurrence)
- Reduce
  - Sort: sort by keys (words)
  - Reduce: Sum together counts for each key (word)

```
map (String key, String value):
```

```
    // key: doc name, value: contents
```

```
    for each word w in value:
```

```
        EmitIntermediate(w, "1");
```

```
reduce (String key, Iterator values):
```

```
    // key: a word; values: a list of counts
```

```
    int result = 0;
```

```
    for each v in values:
```

```
        result += Int(v);
```

```
    Emit(String(result));
```

"At what period," continues Grote, "these poems, or indeed any other Greek poems, first began to be written, must be matter of conjecture, though there is ground for assurance that it was before the time of Solon. If, in the absence of evidence, we may venture upon naming any more determinate period, the question a once suggests itself, What were the purposes which, in that state of society, a manuscript at its first commencement must have been intended to answer? For whom was a written Iliad necessary? Not for the rhapsodes; for with them it was not only planted in the memory, but also interwoven with the feelings, and conceived in conjunction with all those flexions and intonations of voice, pauses, and other oral artifices which were required for emphatic delivery



at 1  
at 1  
any 1  
began 1  
continues 1  
first 1  
Greek 1  
Grote 1  
indeed 1  
or 1  
other 1  
other 1  
period 1  
poems 1  
these 1  
what 1  
what 1

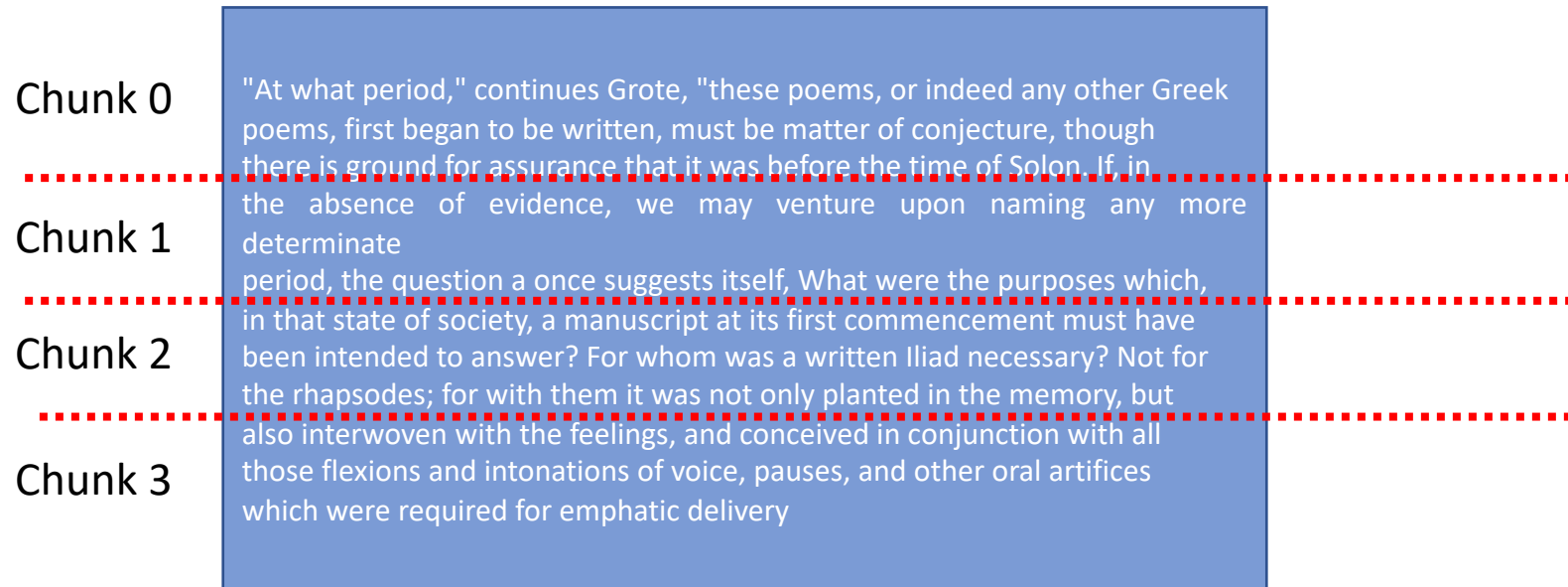


at 2  
any 1  
began 1  
continues 1  
first 1  
Greek 1  
Grote 1  
indeed 1  
or 1  
other 2  
period 1  
poems 1  
these 1  
what 2

# Under the Hood: Pre-map

"At what period," continues Grote, "these poems, or indeed any other Greek poems, first began to be written, must be matter of conjecture, though there is ground for assurance that it was before the time of Solon. If, in the absence of evidence, we may venture upon naming any more determinate period, the question a once suggests itself, What were the purposes which, in that state of society, a manuscript at its first commencement must have been intended to answer? For whom was a written Iliad necessary? Not for the rhapsodes; for with them it was not only planted in the memory, but also interwoven with the feelings, and conceived in conjunction with all those flexions and intonations of voice, pauses, and other oral artifices which were required for emphatic delivery

# Under the Hood: Partition Input





# Under the Hood: Partition Input

Key: chunk\_0.in, val=

"At what period," continues Grote, "these poems, or indeed any other Greek poems, first began to be written, must be matter of conjecture, though there is ground for assurance that it was before the time of Solon. If, in

Key: chunk\_1.in, val=

the absence of evidence, we may venture upon naming any more determinate period, the question a once suggests itself, What were the purposes which,

Key: chunk\_2.in, val=

in that state of society, a manuscript at its first commencement must have been intended to answer? For whom was a written Iliad necessary? Not for the rhapsodes; for with them it was not only planted in the memory, but

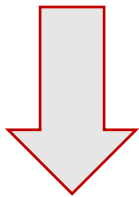
Key: chunk\_3.in, val=

also interwoven with the feelings, and conceived in conjunction with all those flexions and intonations of voice, pauses, and other oral artifices which were required for emphatic delivery

# Under the Hood: Partition Input

Key: chunk\_0.in, val=

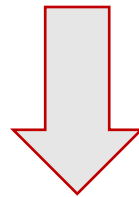
"At what period," continues Grote, "these poems, or indeed any other Greek poems, first began to be written, must be matter of conjecture, though there is ground for assurance that it was before the time of Solon. If, in



Mapper 0

Key: chunk\_1.in, val=

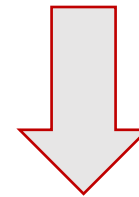
the absence of evidence, we may venture upon naming any more determinate period, the question a once suggests itself, What were the purposes which,



Mapper 1

Key: chunk\_2.in, val=

in that state of society, a manuscript at its first commencement must have been intended to answer? For whom was a written Iliad necessary? Not for the rhapsodes; for with them it was not only planted in the memory, but



Mapper 2

Key: chunk\_3.in, val=

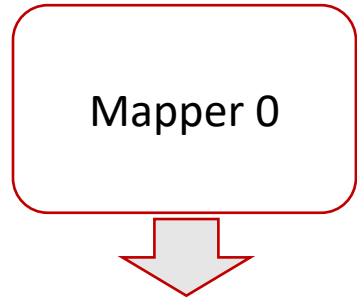
also interwoven with the feelings, and conceived in conjunction with all those flexions and intonations of voice, pauses, and other oral artifices which were required for emphatic delivery



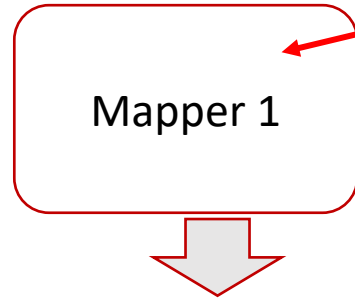
Mapper 3

# Under the Hood: Map

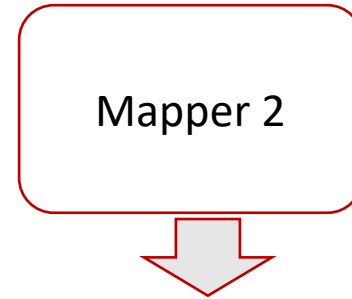
Programmer writes this!



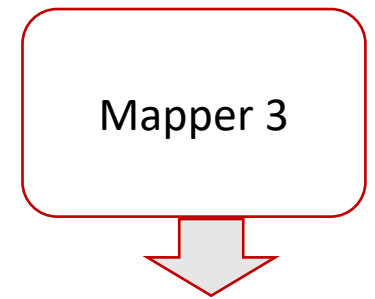
Word	Count
poems	1
poems	1
be	1
be	1
of	1
of	1
at	1
what	1
period	1
continues	1
grote	1
these	1
or	1
indeed	1



Word	Count
the	1
the	1
the	1
absence	1
of	1
evidence	1
we	1
may	1
venture	1
upon	1
naming	1
any	1



...



# Under the Hood: Keyspace Partitioning (simplified)

Keyspace Partition	Who Owns It
"a..." -> "m..."	Reducer 0
"n..." -> "z..."	Reducer 1

# Under the Hood: Shuffle (simplified)

Word	Count
poems	1
poems	1
be	1
be	1
of	1
of	1
at	1
what	1
period	1
continues	1
grote	1
these	1
or	1
indeed	1
any	1
other	1
greek	1
first	1
began	1

Word	Count
the	1
the	1
the	1
absence	1
of	1
evidence	1
we	1
may	1
venture	1
upon	1
naming	1
any	1
more	1
what	1
period	1
question	1
a	1

Partition 0

be: {1, 1}  
at: 1  
grote: 1  
any: {1, 1}  
...

# Under the Hood: Shuffle (simplified)

Word	Count
poems	1
poems	1
be	1
be	1
of	1
of	1
at	1
what	1
period	1
continues	1
grote	1
these	1
or	1
indeed	1
any	1
other	1
greek	1
first	1
began	1

Word	Count
the	1
the	1
the	1
absence	1
of	1
evidence	1
we	1
may	1
venture	1
upon	1
naming	1
any	1
more	1
what	1
period	1
question	1
a	1

Partition 0

be: {1, 1}  
 at: 1  
 grote: 1  
 any: {1, 1}  
 ...

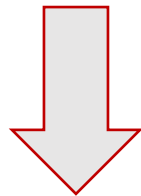
Partition 1

poems: {1, 1}  
 what: {1, 1}  
 question: 1  
 ...

# Under the Hood: Sort Keys

Partition 0

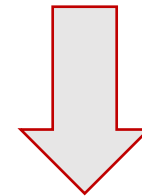
```
be: {1, 1}
at: 1
grote: 1
any: {1, 1}
...
```



```
any: {1, 1}
at: 1
be: {1, 1}
grote: 1
...
```

Partition 1

```
poems: {1, 1}
what: {1, 1}
question: 1
...
```



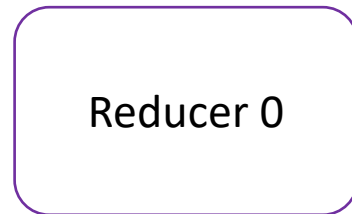
```
poems: {1, 1}
question: 1
what: {1, 1}
...
```

Sort

# Under the Hood: Reduce

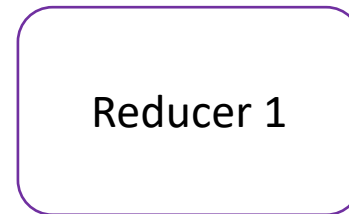
Sorted Partition 0

```
any: {1, 1}
at: 1
be: {1, 1}
grote: 1
...
```



Sorted Partition 1

```
poems: {1, 1}
question: 1
what: {1, 1}
...
```





# Under the Hood: Reduce

Sorted Partition 0

```
any: {1, 1}
at: 1
be: {1, 1}
grote: 1
...
```

Reducer 0

Sorted Partition 1

```
poems: {1, 1}
question: 1
what: {1, 1}
...
```

Reducer 1

← Programmer writes this!

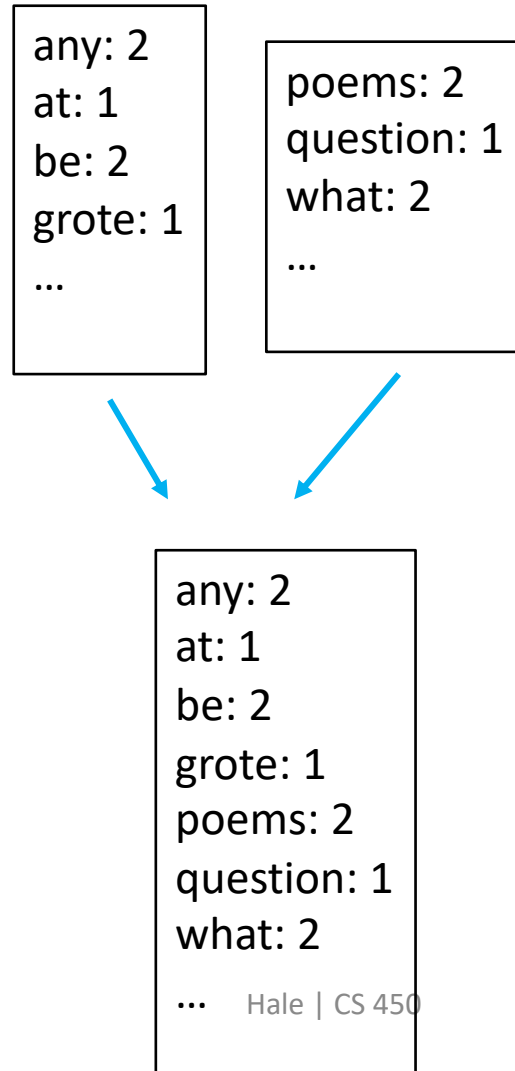


```
any: 2
at: 1
be: 2
grote: 1
...
```



```
poems: 2
question: 1
what: 2
...
```

# Up to the user: Merge



# Fault Tolerance

- How to detect failure?
  - Master pings workers periodically (heartbeats)
  - If they don't respond, reset map/reduce tasks and give them to someone else
- How can we ensure persistence?

# Locality

- MapReduce typically uses a distributed file system for input data (Google File System or BigTable)
- How do we reduce network overhead?
  - Schedule map tasks near their input data!

# Other Examples

- Distributed grep
- Map: emit a line if it matches pattern
- Reduce: copy intermediate data to output

