Lecture Outline

- Introduction
- Hadoop
- Chord
What’s a distributed system?
What’s a distributed system?

A distributed system is a collection of loosely coupled nodes interconnected by a communication network.
What’s a distributed system?

A distributed system is a collection of loosely coupled nodes interconnected by a communication network.

In a distributed system, users access remote resources in the same way they access local resources.
Why distributed systems?
Scalability

PC
Scalability

• What if one computer is not enough?

PC
Scalability

- What if one computer is not enough?
  - Buy a bigger (server-class) computer
Scalability

- What if one computer is not enough?
  - Buy a bigger (server-class) computer

- What if the biggest computer is not enough?
Scalability

- What if one computer is not enough?
  - Buy a bigger (server-class) computer

- What if the biggest computer is not enough?
  - Buy many computers
Scalability
Scalability
Scalability

Network switches (connects nodes with each other and with other racks)
Scalability

Rack

Network switches (connects nodes with each other and with other racks)

Many nodes/BLADES (often identical)
Scalability

Rack

Network switches (connects nodes with each other and with other racks)

Many nodes/blades (often identical)

Storage device(s)
What if cluster is too big to fit into machine room?

- Build a separate building for the cluster
- Building can have lots of cooling and power
- Result: Data center
What if cluster is too big to fit into machine room?
- Build a separate building for the cluster
- Building can have lots of cooling and power
What if cluster is too big to fit into machine room?
- Build a separate building for the cluster
- Building can have lots of cooling and power
- Result: Data center
Google Data Center in Oregon
Data centers (size of a football field)
• A warehouse-sized computer
  - A single data center can easily contain 10,000 racks with 100 cores in each rack (1,000,000 cores total)
Google Data Centers World Wide
Why distributed systems?

• Resource sharing
  - E.g., Napster
Why distributed systems?

- Resource sharing
  - E.g., Napster
- Computation speedup
  - E.g., Hadoop
Why distributed systems?

- Resource sharing
  - E.g., Napster
- Computation speedup
  - E.g., Hadoop
- Reliability
  - E.g., Amazon S3
Why distributed systems?

- Resource sharing
  - E.g., Napster
- Computation speedup
  - E.g., Hadoop
- Reliability
  - E.g., S3

Issues?
Lecture Outline

• Introduction

• Hadoop

• Chord
Lecture Outline

- Introduction
- Hadoop
- Chord
What’s the Hadoop

- A project based on **GFS** and MapReduce
  - Distribute data across machines
  - Try to achieve the reliability and scalability

The Google File System
What’s the Hadoop

- A project based on GFS and MapReduce
  - Distribute data across machines
  - Try to achieve the reliability and scalability

- An open-source software framework for big data
  - Distributed storage
  - Distributed processing
Why Hadoop?
Hadoop Components
Hadoop Components

NameNode

DataNode1

DataNode2

DataNode3

DataNode4

DataNode5
Hadoop Components

- Two core components
  - The Hadoop distributed file system (HDFS)
Hadoop Components

- Two core components
  - The Hadoop distributed file system (HDFS)
Hadoop Components

- Two core components
  - The Hadoop distributed file system (HDFS)
Hadoop Components

- Two core components
  - The Hadoop distributed file system (HDFS)
Hadoop Components

- Two core components
  - The Hadoop distributed file system (HDFS)
Hadoop Components

- Two core components
  - The Hadoop distributed file system (HDFS)
  - MapReduce software framework
Hadoop Components

• Two core components
  - The Hadoop distributed file system (HDFS)
  - MapReduce software framework
Hadoop Components

- Two core components
  - The Hadoop distributed file system (HDFS)
  - MapReduce software framework
Hadoop Components

- Two core components
  - The Hadoop distributed file system (HDFS)
  - MapReduce software framework
Hadoop Components

- Two core components
  - The Hadoop distributed file system (HDFS)
  - MapReduce software framework
HDFS

• HDFS is responsible for storing data
  - Data is split into blocks and distributed across nodes
  - Each block is replicated

• Implementation of HDFS:
  - Based on Google’s GFS
  - Offers redundant storage for massive amounts of data
Getting Data in/out of HDFS

- **Hadoop API:**
  - Use `hadoop fs` to work with data in HDFS
  - `hadoop fs -copyFromLocal local_dir /hdfs_dir`
  - `hadoop fs -copyToLocal /hdfs_dir local_dir`
HDFS Example

NameNode: Stores metadata only

- NameNode holds metadata for the data files
- DataNodes hold the actual blocks
  - Each block is replicated three times

Metadata:
/usr/data/foo
  -> 1, 2, 4
/usr/data/bar
  -> 3, 5

DataNodes: Store Blocks
HDFS Example

NameNode: Stores metadata only

Metadata: /usr/data/foo
-> 1, 2, 4
/usr/data/bar
-> 3, 5

DataNodes: Store Blocks

- NameNode holds metadata for the data files
- DataNodes hold the actual blocks
  - Each block is replicated three times
- When a client wants to read file

Client

bar?
HDFS Example

NameNode: Stores metadata only

- NameNode holds metadata for the data files
- DataNodes hold the actual blocks
  - Each block is replicated three times
- When a client wants to read file

DataNodes: Store Blocks

Client

bar?

< DataNode IPs, 3, 5 >

Metadata:
/usr/data/foo
-> 1, 2, 4
/usr/data/bar
-> 3, 5
NameNode: Stores metadata only

- **Metadata:**
  - `/usr/data/foo` -> 1, 2, 4
  - `/usr/data/bar` -> 3, 5

DataNodes: Store Blocks

- **DataNode IPs:** 3, 5

- **Client**

- **NameNode** holds metadata for the data files
- **DataNodes** hold the actual blocks
  - Each block is replicated three times
- When a client wants to read file
Case Study

- Policy: Assigning blocks based on disk space

- tiger
- frog
- lion
- monkey
- python
Case Study

• Policy: Assigning blocks based on disk space
Case Study

- Policy: Assigning blocks based on disk space
Case Study

• Policy: Assigning blocks based on disk space
Case Study

• Policy: Assigning blocks based on disk space
Case Study

- Policy: Assigning blocks based on disk space
Recall: Hadoop Components

- Two core components
  - The Hadoop distributed file system (HDFS)
  - MapReduce software framework

<table>
<thead>
<tr>
<th>JobTracker</th>
<th>TaskTracker</th>
<th>TaskTracker</th>
<th>TaskTracker</th>
<th>TaskTracker</th>
<th>TaskTracker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>HDFS</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NameNode:  
DataNode1:  
DataNode2:  
DataNode3:  
DataNode4:  
DataNode5:  
MapReduce

• A method distributing a task across nodes:
  - Each node processes data stored on that node

• Consists of two phases:
  - Map
  - Reduce
MapReduce

- A method distributing a task across nodes:
  - Each node processes data stored on that node

- Consists of two phases:
  - Map
  - Reduce
Features of MapReduce

• Automatic parallelization and distribution
• A clean abstraction for programmers
  - MapReduce programs are usually written in Java

• MapReduce abstracts all the housekeeping away from the developer:
  - Developers concentrate on Map and Reduce functions
MapReduce Example: Word Count

Count the # of occurrences of each word in a large amount of input data

```
Map(input_key, input_value) {
    foreach word w in input_value:
        emit(w, 1);
}
```
MapReduce Example: Word Count

Count the # of occurrences of each word in a large amount of input data

```java
Map(input_key, input_value) {
    foreach word w in input_value:
        emit(w, 1);
}
```

• Input to the Mapper

(3414, ‘the cat sat on the mat’)
(3437, ‘the aardvark sat on the sofa’)
MapReduce Example: Word Count

Count the # of occurrences of each word in a large amount of input data

Map(input_key, input_value) {
    foreach word w in input_value:
        emit(w, 1);
}

• Input to the Mapper

(3414, ‘the cat sat on the mat’)  
(3437, ‘the aardvark sat on the sofa’)

• Output from the Mapper

(‘the’, 1), (‘cat’, 1), (‘sat’, 1), (‘on’, 1),  
(‘the’, 1), (‘mat’, 1), (‘the’, 1), (‘aardvark’, 1),  
(‘sat’, 1), (‘on’, 1), (‘the’, 1), (‘sofa’, 1)
Map Phase

Mapper Input

the cat sat on the mat
the aardvark sat on the sofa
Map Phase

Mapper Input

the cat sat on the mat
the aardvark sat on the sofa

Mapper Output

(‘the’, 1),
(‘cat’, 1),
(‘sat’, 1),
(‘on’, 1),
(‘the’, 1),
(‘mat’, 1),
(‘the’, 1),
(‘aardvark’, 1),
(‘sat’, 1),
(‘on’, 1),
(‘the’, 1),
(‘sofa’, 1)
Reducer

- After the Map, all the intermediate values for a given intermediate key are combined together into a list
Reducer

- After the Map, all the intermediate values for a given intermediate key are combined together into a list.
Reducer

Add up all the values associated with each intermediate key:

```java
Reduce(output_key, intermediate_vals) {
    set count = 0;
    foreach v in intermediate_vals:
        count += v;
    emit(output_key, count);
}
```
Reducer

Add up all the values associated with each intermediate key:

```java
Reduce(output_key, intermediate_vals) {
    set count = 0;
    foreach v in intermediate_vals:
        count += v;
    emit(output_key, count);
}
```

• Output from the Reducer

(`'the', 4), (`'sat', 2), (`'on', 2), (`'sofa', 1),
(`'mat', 1), (`'cat', 1), (`'aardvark', 1)
Map + Reduce

Mapper Input
the cat sat on the mat
the aardvark sat on the sofa

Mapping
(‘the’, 1),
(‘cat’, 1),
(‘sat’, 1),
(‘on’, 1),
(‘the’, 1),
(‘mat’, 1),
(‘the’, 1),
(‘aardvark’, 1),
(‘sat’, 1),
(‘on’, 1),
(‘the’, 1),
(‘sofa’, 1)

Shuffling
aardvark, 1
cat, 1
mat, 1
on [1, 1]
sat [1, 1]
sofa, 1
the [1, 1, 1, 1]

Reducing
aardvark, 1
cat, 1
mat, 1
on, 2
sat, 2
sofa, 1
the, 4
Why we care about counting words

- Word count is challenging over massive amounts of data
- Fundamentals of statistics often are aggregate functions
- Most aggregation functions have distributive nature
- MapReduce breaks complex tasks into smaller pieces which can be executed in parallel
Discussions
Lecture Outline

• Introduction

• Hadoop

• Chord
Lecture Outline

• Introduction
• Hadoop
• Chord
Why Chord?
Chord
Chord
Chord
Chord

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[2,3)</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>[3,5)</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>[5,1)</td>
<td>0</td>
</tr>
</tbody>
</table>
Chord

![Chord diagram]

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[2,3)</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>[3,5)</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>[5,1]</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>[4,5)</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>[5,7)</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>[7,3)</td>
<td>0</td>
</tr>
</tbody>
</table>
Importance: Complexity!

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[1,2)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>[2,4)</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>[4,0)</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[2,3)</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>[3,5)</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>[5,1)</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>[4,5)</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>[5,7)</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>[7,3)</td>
<td>0</td>
</tr>
</tbody>
</table>
Node Joins

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[1,2)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>[2,4)</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>[4,0)</td>
<td>0</td>
</tr>
</tbody>
</table>

- start: int.
- succ.

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[2,3)</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>[3,5)</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>[5,1)</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>[4,5)</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>[5,7)</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>[7,3)</td>
<td>0</td>
</tr>
</tbody>
</table>
Node Joins

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[1,2)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>[2,4)</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>[4,0)</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[2,3)</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>[3,5)</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>[5,1)</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>[4,5)</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>[5,7)</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>[7,3)</td>
<td>0</td>
</tr>
</tbody>
</table>
Node Joins

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[1,2)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>[2,4)</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>[4,0)</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[2,3)</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>[3,5)</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>[5,1)</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>[4,5)</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>[5,7)</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>[7,3)</td>
<td>0</td>
</tr>
</tbody>
</table>
Node Joins

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[1,2)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>[2,4)</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>[4,0)</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[2,3)</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>[3,5)</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>[5,1)</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>[4,5)</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>[5,7)</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>[7,3)</td>
<td>0</td>
</tr>
</tbody>
</table>
Node Joins

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[1,2)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>[2,4)</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>[4,0)</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[2,3)</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>[3,5)</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>[5,1)</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>[4,5)</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>[5,7)</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>[7,3)</td>
<td>0</td>
</tr>
</tbody>
</table>
Case Study: Node Leaves

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[1,2)</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>[2,4)</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>[4,0)</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>[7,0)</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>[0,2)</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>[2,6)</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>start</th>
<th>int.</th>
<th>succ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>[4,5)</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>[5,7)</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>[7,3)</td>
<td>0</td>
</tr>
</tbody>
</table>
Chord Issues?