

Homework 4 Report

Wilson Neira

Part I: Reading and Comments!

Introduction to Distributed System Design

Updated 4 days ago by Wilson Neira

What I found most informative in this introduction was how strongly it emphasizes that failure is normal in distributed systems, not an edge case, and how much design effort goes into detecting, handling, and recovering from partial failures rather than preventing them outright. The discussion of things like network partitions, timeouts, Heisenbugs, and the 8 fallacies helped me connect theory to why simple assumptions break down once systems are spread across machines. I had some knowledge with client-server models, networking basics, and performance bottlenecks from prior internship experience working on backend related services and infrastructure related, but this reading deepened my understanding of why mitigation strategies like retries, timeouts, replication, and simplicity are so critical in real distributed systems.

[week_4_readings](#)

Introduction to Distributed System Design

Updated 4 days ago by shrijan s shetty

The 8 Fallacies of Distributed Computing were eye-opening for me. I had been unconsciously assuming several of these were true, particularly:

1. **Latency is zero** - I never really considered how network delays impact system design decisions, like whether to make many small calls versus one large call
2. **There is one administrator** - This really made me think about how different teams, departments, or even companies might manage different parts of a distributed system, each with their own policies and priorities

The discussion around failure types was also incredibly valuable, especially the distinction between Heisenbugs and Bohrbugs. The Ken Arnold quote about designing with the expectation of failure really resonated - "When you design distributed systems, you have to say, 'Failure happens all the time.'"

From my time at National Instruments, I had hands-on experience with RPC. We needed to enable our Terminal application to communicate with a custom LabVIEW server, which required implementing RPC calls to bridge these different systems.

One major project involved migrating from WCF (Windows Communication Foundation) to gRPC when Microsoft discontinued WCF support.

[week_4_readings](#)

[Edit](#)

 0



19 views

1 Followup Discussions

 Resolved

[@446_f1](#) 



Wilson Neira 4 days ago

Hello Shrijan S Shetty!, I learned from your post more how other industries work (like RPC and moving from WCF to gRPC) makes the fallacies and failure concepts feel very real, especially how latency and multiple administrators directly shape design choices in practice. Thanks for sharing!

 1

Part II: Infrastructure set up!

```
Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems (main)
```

```
$ ssh -i "/c/Users/Owner/Documents/Audacity/Northeastern University/CS 6650 Building Scalable Distributed Systems/Week 1 Whh  
hhhat is all of this!/web-service-gin.pem" ec2-user@ec2-54-234-94-111.compute-1.amazonaws.com
```

A newer release of "Amazon Linux" is available.

Version 2023.10.20260120:

Version 2023.10.20260202:

Run `"/usr/bin/dnf check-release-update"` for full release and version update info

```
#_
~\ #####
~~ \#####\
~~ \###|
~~ \|/
~~ V~' '-> Amazon Linux 2023 (ECS Optimized)
~~~~ /
~~~~ _ _ \
~~~~ _/_/_/
~~~~ /m/'
```

For documentation, visit <http://aws.amazon.com/documentation/ecs>

Last login: Mon Feb 2 12:43:35 2026 from 98.15.96.233

```
[ec2-user@ip-172-31-19-156 ~]$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker-gs-ping	latest	643981103e08	12 days ago	1.27GB
amazon/amazon-ecs-agent	latest	192298286f1f	3 weeks ago	120MB
ebs-csi-driver	latest	702835b18362	3 weeks ago	58.8MB
ecs-service-connect-agent	interface-v1	369a1f77771a	6 weeks ago	204MB
amazon/amazon-ecs-pause	0.1.0	9dd4685d3644	11 years ago	702kB

Step 1: Configure

```
AwksAD5hC/83/r+iQsVvyjd4oxEXLMOOPE3VDE6dLCcYqAghsEAEADMMxNjAwOTk50TU2NCIMztKGuACnoTbh1srtK08C5c+5ao0DdIcLd1+atAhkRnM6YhXqQcs
iCf1E1v9i0iU04J0Jfzx8Px49KQ5+cLVeUmqW/e061x7L6GF9siR7WkhHXK10c0AAUFiFAyPikTOY8oMacEjJb11XBjY7RQc3iy2HAqjHc1mE9ZJYpB6kTDEoPNPLg
c84JjYbBkbi/vz4k+fuFvCbooJ5GuCazJvwSXPTyNkLD09rz1dSnoYyc7hkX/oS+1D564Pp8zArLqrZG814PPY+aUzAMa9bwzfftYpgNgkEVEVAJ403cn3iEoUmLg
h8BFfemwsKcjEs/4Xgbj4QAwFa810g7tp1r1e73TNSwRf1EnNCFKXTEmV08y4cEHsH05Ys1kZKQV1Rpdd77J/MBjQueAUJogBuBXJmWAdRIh+1CiNiSYr8pr
NHoNDHFsn21Yhba4/USoptTqg3W10MGZZNwXgzBxxsHzkBDkfUeUgUfGasqtb7Zxh5wJTMRD1arV8TH2Uxsf3hsIzBg2MB0xFEV3UFxRfmb407Qk4bFncW5jy
2r1Ry3Bz7+pEG8tR2tzG7JNjeySnEXtqCe/5b76L/Eat6Dpk1V541J/WUu
Default region name [us-east-1]:
Default output format [json]:
PS C:\Users\Owner\Documents\Audacity\Building-Scalable-Distributed-Systems> ssh -i "C:\Users\Owner\Documents\Audacity\Northe
```

Step 2: Setup ECR

1. Navigate to Amazon ECR
2. Create Repository

[illegible]

- ### 3. Copy the Repository URI

316009999564.dkr.ecr.us-east-1.amazonaws.com/hello-service

Step 3: Push Docker Image to ECR

1. Get your ECR repository URL

```
Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems (main)
$ ECR_URL=$(aws ecr describe-repositories \
  --repository-names hello-service \
  --region us-east-1 \
  --query 'repositories[0].repositoryUri' \
  --output text)

Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems (main)
$ echo "ECR URL: $ECR_URL"
ECR URL: 316009999564.dkr.ecr.us-east-1.amazonaws.com/hello-service
```

2. Authenticate Docker to ECR

```
Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems (main)
$ ECR_BASE=$(echo $ECR_URL | cut -d '/' -f1)

Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems (main)
$ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin $ECR_BASE
Login Succeeded

Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems (main)
$ password=$(aws ecr get-login-password --region us-east-1)

Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems (main)
$ echo "$password" | docker login --username AWS --password-stdin "$ECR_BASE"
Login Succeeded
```

3. Tag your image for ECR

```
[ec2-user@ip-172-31-19-156 ~]$ docker tag hello-service:latest "$ECR_URL:latest"
```

4. Build and push the image

```
[ec2-user@ip-172-31-19-156 ~]$ docker push "$ECR_URL:latest"
The push refers to repository [316009999564.dkr.ecr.us-east-1.amazonaws.com/hello-service]
44d3a0ac86f9: Preparing
a1153bed92ba: Preparing
8fe368f16a1e: Preparing
93bf13679d88: Preparing
bca63f61669a: Preparing
5f70bf18a086: Preparing
53bf8f783f7e: Preparing
5a797796f075: Preparing
5296ce17e466: Preparing
d7d9aa728fbb: Preparing
1a0f650a8421: Preparing
da7213941eca: Preparing
no basic auth credentials
```

5. Verify upload

```
[ec2-user@ip-172-31-19-156 ~]$ aws ecr list-images \
  --repository-name hello-service \
  --region us-east-1 \
  --query 'imageIds[*].imageTag' \
  --output table
[ec2-user@ip-172-31-19-156 ~]$
```

Step 4: Create ECS Cluster

1. Navigate to Amazon ECS
2. Create Cluster

Amazon ECR > Private registry > Repositories

Amazon Elastic Container Service

Private registry

Repositories

Settings

Public registry

Repositories

Successfully created private repository, hello-service

Private repositories (1)

Search by repository substring

Repository name	URI	Created at	Tag immutability	Encryption
hello-service	316009999564.dkr.ecr.us-east-1.amazonaws.com/hello-service	February 07, 2026, 22:15:56 (UTC-05)	Immutable	AES-256

Step 5: Create Task Definition

1. Navigate to Task Definitions
2. Configure Task Definition
3. Add Container
4. Create the Task Definition

Task definitions > hello-task > Revision 1 > Containers

Task definition successfully created

hello-task:1 has been successfully created. You can use this task definition to deploy a service or run a task.

Notifications 0 0 2 0 0

hello-task:1

Last updated February 8, 2026, 15:35 (UTC-5:00)

Deploy Actions

Overview Info

ARN
arn:aws:ecs:us-east-1:316009999564:task-definition/hello-task:1

Status
ACTIVE

Time created
February 8, 2026, 15:34 (UTC-5:00)

App environment
Fargate

Task role
LabRole

Task execution role
LabRole

Operating system/Architecture
Linux/X86_64

Network mode
awsvpc

Fault injection
Turned off

Containers JSON Task placement Volumes (0) Requires attributes Tags

Task size

Task CPU
256 units (0.25 vCPU)

Task CPU maximum allocation for containers

CPU (unit)

Task memory
512 MiB (0.5 GiB)

Task memory maximum allocation for container memory reservation

Memory (MiB)

Step 6: Run a Task

- 1. Navigate to your Cluster
- 2. Run New Task
- 3. Configure Networking
- 4. Run the Task

hello-cluster

Last updated
February 8, 2026, 16:05 (UTC-5:00)

Actions

Create with Exp

Cluster overview

ARN
arn:aws:ecs:us-east-1:316009999564:cluster/hello-cluster

Status
Active

CloudWatch monitoring
Default

Registered container instances
-

Services

Tasks

Draining
-

Active
-

Pending
-

Running
1

Services

Tasks

Infrastructure

Metrics

Scheduled tasks

Configuration

Event history

Tags

Tasks (2)

Last updated
February 8, 2026, 16:05 (UTC-5:00)

Manage tags

Stop

R

Filter tasks by property or value

Filter desired status
Any desired status

Filter launch type
Any launch type

Task

Last status

Desired status

Task definition

Health status

Created at

Started by

14673b8a38254b74ae32...

Running

Running

hello-task:1

Unknown

1 minute ago

-

Step 7: Getting the Public IP and Testing!

- 1. View Task Details

eni-0d642195b5733e34e

Network interface summary for eni-0d642195b5733e34e

Delete network interface

Actions

Network interface details

Network interface ID
eni-0d642195b5733e34e

Network interface status
In-use

VPC ID
vpc-0cea8d47d1bab81b6

Owner
316009999564

Source/dest. check
True

IP addresses

Private IPv4 address
172.31.30.78

IPv6 addresses
-

Association ID
-

IPv4 Prefix Delegation
-

Name
-

Interface type
Elastic network interface

Subnet ID
subnet-06d51511c0d8dfdb7

Requester ID
578734482556

Managed
False

Elastic Fabric Adapter
False

Secondary public IPv4 addresses
-

Elastic IP address owner
amazon

IPv6 Prefix Delegation
-

Description
arn:aws:ecs:us-east-1:316009999564:attachment/3671f4cf-e906-4a5d-812a-1d20e8b29e8e

Security groups
sg-0879c35d44d189854 (hello-service-sg)

Availability Zone
us-east-1d

Requester-managed
True

Operator
-

Public IPv4 address
3.94.195.212

Secondary private IPv4 addresses
-

MAC address
0a:ff:d3:d8:fe:91

Primary IPv6 address
-

- 2. Find Public IP

3.94.195.212

3. Test Your Service

```
Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems (main)
$ # Get albums
curl http://3.94.195.212:8080/albums

# Get album with ID 1
curl http://3.94.195.212:8080/albums/1
[
  {
    "id": "1",
    "title": "Blue Train",
    "artist": "John Coltrane",
    "price": 56.99
  },
  {
    "id": "2",
    "title": "Jeru",
    "artist": "Gerry Mulligan",
    "price": 17.99
  },
  {
    "id": "3",
    "title": "Sarah Vaughan and Clifford Brown",
    "artist": "Sarah Vaughan",
    "price": 39.99
  }
]
{
  "id": "1",
  "title": "Blue Train",
  "artist": "John Coltrane",
  "price": 56.99
}
Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems (main)
$
```

Thinking about your result...

Difference in choosing EC2 vs ECS.

EC2 is me managing an actual server/VM myself, while ECS (with Fargate) is me just deploying containers and AWS manages the servers for me.

What is a VPC Links to an external site and subnetLinks to an external site.? How did you get access into the default VPC?

A VPC is like my own private network inside AWS and a subnet is a smaller IP range inside that network in one availability zone, and I got access because AWS automatically gives my account a default VPC in each region and I selected it when launching the task.

What is TCP? How is it different than UDP?

TCP is like a reliable confirmed delivery connection where packets arrive in order, while UDP is faster but doesn't guarantee delivery or order.

How do you control resources allocated to a task?

I control task resources by setting the CPU and memory values in the ECS Task Definition (and that limits what each running task/container can use).

Part III: A Fun Experiment!

Implementation

1. Create S3

Amazon S3 > Buckets

✔ Successfully created bucket "wilson-mapreduce-lab"
To upload files and folders, or to configure additional bucket settings, choose [View details](#).

General purpose buckets

All AWS Regions

Directory buckets

General purpose buckets (1) [Info](#)

🔄

📄 Copy ARN

Empty

Delete

Create bucket

🔍 Find buckets by name

< 1 > ⚙️

Name	AWS Region	Creation date
wilson-mapreduce-lab	US East (N. Virginia) us-east-1	February 8, 2026, 16:26:26 (UTC-05:00)

2. Upload file

✔ Upload succeeded
For more information, see the [Files and folders](#) table.

Upload: status

ⓘ After you navigate away from this page, the following information is no longer available.

Summary

Destination

s3://wilson-mapreduce-lab

Succeeded

✔ 1 file, 163.9 KB (100.00%)

Failed

☹ 0 files, 0 B (0%)

Files and folders

Configuration

Files and folders (1 total, 163.9 KB)

🔍 Find by name

Name	Folder	Type	Size	Status	Error
shakespeare-hamlet.txt	-	text/plain	163.9 KB	✔ Succeeded	-

3. Build image for each code mapper, reducer, and splitter

```
docker build -t reducer-service --build-arg SERVICE=reducer .
```

View build details: [docker-desktop://dashboard/build/desktop-linux/desktop-linux/xo40frxvtlbr9154ggultw8af](#)

```
[+] Building 38.9s (15/15) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 469B                                0.0s
=> [internal] load metadata for docker.io/library/alpine:3.20      0.4s
=> [internal] load metadata for docker.io/library/golang:1.22-alpine 0.4s
=> [internal] load .dockerignore                                    0.0s
=> => transferring context: 2B                                       0.0s
=> [build 1/6] FROM docker.io/library/golang:1.22-alpine@sha256:1699c10032ca2582ec89a24a1312d986a3f094aed3d5c1147b19 0.1s
=> => resolve docker.io/library/golang:1.22-alpine@sha256:1699c10032ca2582ec89a24a1312d986a3f094aed3d5c1147b19880afe 0.1s
=> [stage-1 1/3] FROM docker.io/library/alpine:3.20@sha256:a4f4213abb84c497377b8544c81b3564f313746700372ec4fe84653e4 0.1s
=> => resolve docker.io/library/alpine:3.20@sha256:a4f4213abb84c497377b8544c81b3564f313746700372ec4fe84653e4fb03805 0.1s
=> [internal] load build context                                    0.0s
=> => transferring context: 310B                                       0.0s
=> CACHED [build 2/6] WORKDIR /app                                  0.0s
=> CACHED [build 3/6] COPY go.mod go.sum ./                          0.0s
=> CACHED [build 4/6] RUN go mod download                          0.0s
=> CACHED [build 5/6] COPY . .                                       0.0s
=> [build 6/6] RUN CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build -o /out/app ./reducer 35.8s
=> CACHED [stage-1 2/3] WORKDIR /app                                0.0s
=> [stage-1 3/3] COPY --from=build /out/app /app/app                0.1s
=> exporting to image                                              1.9s
=> => exporting layers                                                1.4s
=> => exporting manifest sha256:3bba8412d9013aba3ac08deaa9c0b93aa704c1da052cd05819aa39011d12d386 0.0s
=> => exporting config sha256:b950cef85314551a8595ea43ad3bb946dcd317a3712ca67352ded90434ac05be 0.0s
=> => exporting attestation manifest sha256:3e2f27bde32e326159a1fc58080d3dad793c9a7347216c0f665d7a42d6f4fc44 0.1s
=> => exporting manifest list sha256:dae09b1629711781744ac8c9ceda2791f756da09c4a5f521b27756f867205bdd 0.0s
=> => naming to docker.io/library/reducer-service:latest           0.0s
=> => unpacking to docker.io/library/reducer-service:latest        0.3s
```

View build details: [docker-desktop://dashboard/build/desktop-linux/desktop-linux/kc2koj9vseo401v9yh1lo74cq](#)

Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems/HW4/mapreduce-lab (main)

\$ ls

Dockerfile go.mod go.sum mapper/ reducer/ shakespeare-hamlet.txt splitter/

4. Create 3 ECR

[Amazon ECR](#) > [Private registry](#) > Repositories

Amazon Elastic
Container Service

Private registry

Repositories

Settings & Settings

Public registry

Repositories

Settings

Public gallery

Amazon ECS

Successfully created private repository, mr-reducer

Private repositories (3)

Search by repository substring

	Repository name	URI	Created at	Tag immutability	Encryption type
<input type="radio"/>	mr-mapper	316009999564.dkr.ecr.us-east-1.amazonaws.com/mr-mapper	February 08, 2026, 17:09:45 (UTC-05)	Immutable	AES-256
<input type="radio"/>	mr-reducer	316009999564.dkr.ecr.us-east-1.amazonaws.com/mr-reducer	February 08, 2026, 17:09:53 (UTC-05)	Immutable	AES-256
<input type="radio"/>	mr-splitter	316009999564.dkr.ecr.us-east-1.amazonaws.com/mr-splitter	February 08, 2026, 17:09:28 (UTC-05)	Immutable	AES-256

5. Tag + push each image

```
Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems/HW4/mapreduce-lab (main)
$ ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
BASE="316009999564.dkr.ecr.us-east-1.amazonaws.com"
```


```
docker tag splitter-service:latest $BASE/mr-splitter:latest
docker tag mapper-service:latest $BASE/mr-mapper:latest
docker tag reducer-service:latest $BASE/mr-reducer:latest
```

```
docker push $BASE/mr-splitter:latest
docker push $BASE/mr-mapper:latest
docker push $BASE/mr-reducer:latest
The push refers to repository [316009999564.dkr.ecr.us-east-1.amazonaws.com/mr-splitter]
1f1feb02db15: Pushed
811d8fb87323: Pushed
fc332be8cea2: Pushed
76eb174b37c3: Pushed
latest: digest: sha256:9bba91c6d79cf92555dfcba09a757f57b2bb2e216fd66c895bb2f464d1da5069 size: 855
The push refers to repository [316009999564.dkr.ecr.us-east-1.amazonaws.com/mr-mapper]
1f1feb02db15: Pushed
66239dc487ad: Pushed
52e27d47c2ba: Pushed
76eb174b37c3: Pushed
latest: digest: sha256:282ce5f9f5d45aa0316ef5c90f3f5a9bba502726bb12761f6cb7e9d10ed8aca3 size: 855
The push refers to repository [316009999564.dkr.ecr.us-east-1.amazonaws.com/mr-reducer]
df6c8daab628: Pushed
e9c7f95ffbb7: Pushed
1f1feb02db15: Pushed
76eb174b37c3: Pushed
latest: digest: sha256:dae09b1629711781744ac8c9ceda2791f756da09c4a5f521b27756f867205bdd size: 855
```

```
Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems/HW4/mapreduce-lab (main)
```

6. Create task definitions (3 total)

Task definitions (4) [Info](#)

Last updated
February 8, 2026, 17:35 (UTC-5:00) 

Filter status: Active

	Task definition	Status of last revision
<input type="radio"/>	hello-task	✓ Active
<input type="radio"/>	mr-mapper-task	✓ Active
<input type="radio"/>	mr-reducer-task	✓ Active
<input type="radio"/>	mr-splitter-task	✓ Active

7. Run Tasks

> Clusters > hello-cluster > Tasks

ARN
arn:aws:ecs:us-east-1:316009999564:cluster/hello-cluster

Status
Active

CloudWatch monitoring
Default

Registered container instances
-

Services
Draining
-

Active
-

Tasks
Pending
-

Running
4

Services | **Tasks** | Infrastructure | Metrics | Scheduled tasks | Configuration | Event history | Tags

Tasks (4)
February 8, 2026, 17:41 (UTC-5:00) Last updated
Manage tags Stop Run new task

Q Filter tasks by property or value

Filter desired status
Any desired status

Filter launch type
Any launch type

<input type="checkbox"/>	Task	Last status	Desired status	Task definition	Health status	Created at	Started by	Started at
<input type="checkbox"/>	14673b8a38254b74ae32...	Running	Running	hello-task:1	Unknown	2 hours ago	-	2 hours ago
<input type="checkbox"/>	3176e1288c13447f9b98...	Running	Running	mr-reducer-task:1	Unknown	50 seconds ago	-	31 seconds ago
<input type="checkbox"/>	425d51ad10304c2f8fa7a...	Running	Running	mr-splitter-task:1	Unknown	3 minutes ago	-	3 minutes ago
<input type="checkbox"/>	d86ca0fb1cf94dea9d908...	Running	Running	mr-mapper-task:2	Unknown	2 minutes ago	-	2 minutes ago

8. Split

```
Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems/HW4/mapreduce-lab (main)
$ curl "http://3.238.198.242:8080/split?s3=s3://wilson-mapreduce-lab/shakespeare-hamlet.txt"
{"chunks":["s3://wilson-mapreduce-lab/mr/chunks/shakespeare-hamlet_20260208T225253Z_chunk00.txt","s3://wilson-mapreduce-lab/mr/chunks/shakespeare-hamlet_20260208T225253Z_chunk01.txt","s3://wilson-mapreduce-lab/mr/chunks/shakespeare-hamlet_20260208T225253Z_chunk02.txt"]}
```

mapreduce-lab > mr/ > chunks/

chunks/

Objects | Properties

Objects (9)
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explain permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	shakespeare-hamlet_20260208T225004Z_chunk00.txt	txt	February 8, 2026, 17:50:05 (UTC-05:00)	53.3 KB	Standard
<input type="checkbox"/>	shakespeare-hamlet_20260208T225004Z_chunk01.txt	txt	February 8, 2026, 17:50:05 (UTC-05:00)	57.4 KB	Standard
<input type="checkbox"/>	shakespeare-hamlet_20260208T225004Z_chunk02.txt	txt	February 8, 2026, 17:50:05 (UTC-05:00)	53.2 KB	Standard
<input type="checkbox"/>	shakespeare-hamlet_20260208T225155Z_chunk00.txt	txt	February 8, 2026, 17:51:56 (UTC-05:00)	53.3 KB	Standard
<input type="checkbox"/>	shakespeare-hamlet_20260208T225155Z_chunk01.txt	txt	February 8, 2026, 17:51:56 (UTC-05:00)	57.4 KB	Standard
<input type="checkbox"/>	shakespeare-hamlet_20260208T225155Z_chunk02.txt	txt	February 8, 2026, 17:51:56 (UTC-05:00)	53.2 KB	Standard

9. Map

Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems/HW4/mapreduce-lab (main)

```
$ curl "http://54.237.79.199:8080/map?s3=s3://wilson-mapreduce-lab/mr/chunks/shakespeare-hamlet_20260208T225253Z_chunk00.txt"
```

```
curl "http://54.237.79.199:8080/map?s3=s3://wilson-mapreduce-lab/mr/chunks/shakespeare-hamlet_20260208T225253Z_chunk01.txt"
curl "http://54.237.79.199:8080/map?s3=s3://wilson-mapreduce-lab/mr/chunks/shakespeare-hamlet_20260208T225253Z_chunk02.txt"
{"out":"s3://wilson-mapreduce-lab/mr/maps/mr_chunks_shakespeare-hamlet_20260208T225253Z_chunk00_txt_20260208T225519Z.json"}
{"out":"s3://wilson-mapreduce-lab/mr/maps/mr_chunks_shakespeare-hamlet_20260208T225253Z_chunk01_txt_20260208T225520Z.json"}
{"out":"s3://wilson-mapreduce-lab/mr/maps/mr_chunks_shakespeare-hamlet_20260208T225253Z_chunk02_txt_20260208T225521Z.json"}
```

[mapreduce-lab](#) > [mr/](#) > [maps/](#)

maps/

Objects

Properties

Objects (3)



Copy S3 URI

Copy URL

Download

Open ↗

Delete

Actions ▼

Create

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	mr_chunks_shakespeare-hamlet_20260208T225253Z_chunk00_txt_20260208T225519Z.json	json	February 8, 2026, 17:55:20 (UTC-05:00)	23.9 KB	Standard
<input type="checkbox"/>	mr_chunks_shakespeare-hamlet_20260208T225253Z_chunk01_txt_20260208T225520Z.json	json	February 8, 2026, 17:55:21 (UTC-05:00)	26.8 KB	Standard
<input type="checkbox"/>	mr_chunks_shakespeare-hamlet_20260208T225253Z_chunk02_txt_20260208T225521Z.json	json	February 8, 2026, 17:55:22 (UTC-05:00)	24.4 KB	Standard

10. Reduce

Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems/HW4/mapreduce-lab (main)

```
$ curl "http://50.19.146.108:8080/reduce?in=s3://wilson-mapreduce-lab/mr/maps/mr_chunks_shakespeare-hamlet_20260208T225253Z_chunk00_txt_20260208T225519Z.json&in=s3://wilson-mapreduce-lab/mr/maps/mr_chunks_shakespeare-hamlet_20260208T225253Z_chunk01_txt_20260208T225520Z.json&in=s3://wilson-mapreduce-lab/mr/maps/mr_chunks_shakespeare-hamlet_20260208T225253Z_chunk02_txt_20260208T225521Z.json"
{"out":"s3://wilson-mapreduce-lab/mr/reduce/final_20260208T225747Z.json","files":3}
```

[mapreduce-lab](#) > [mr/](#) > [reduce/](#)

reduce/

Objects

Properties

Objects (1)



Copy S3 URI

Copy URL

Download

Open ↗

Delete

Actions ▼

Create

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	final_20260208T225747Z.json	json	February 8, 2026, 17:57:48 (UTC-05:00)	73.0 KB	Standard

11. Verify the final JSON produced by reducer and it matches with the real results!

```
PS C:\Users\Owner\Documents\Audacity\Building-Scalable-Distributed-Systems\HW4\mapreduce-lab> & C:/Users/Owner/AppData/Local/Programs/Python/Python310/python.exe c:/Users/Owner/Documents/Audacity/Building-Scalable-Distributed-Systems/HW4/mapreduce-lab/verify_json.py
Total unique words truth: 4815
Total unique words reduced: 4817
Total mismatched words: 0
PASS
```

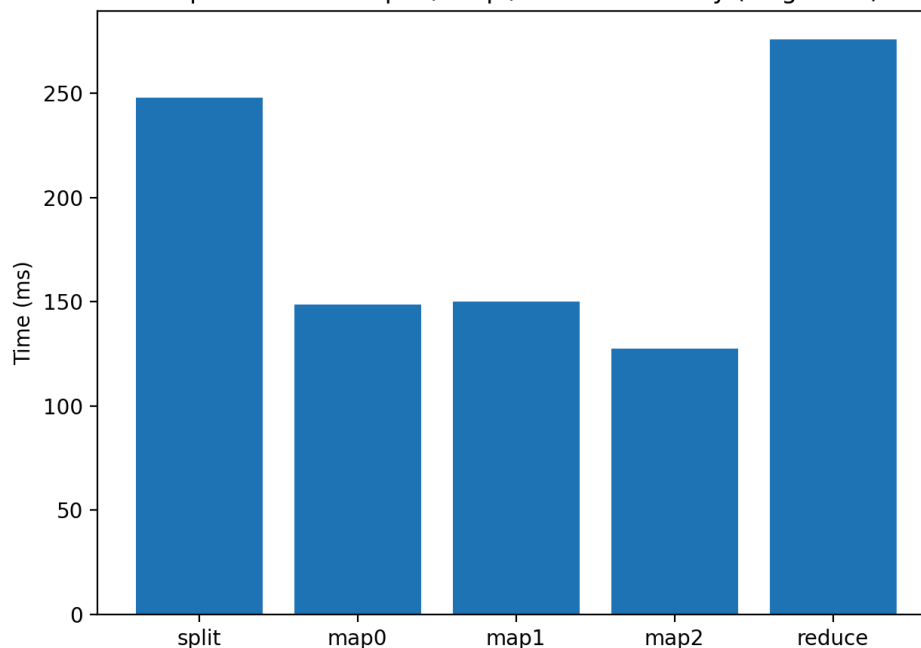
12. Performance Plots

```
Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems/HW4/mapreduce-lab (main)
$ curl -s -o /dev/null -w "split ms=%{time_total}\n" "http://3.238.198.242:8080/split?s3=s3://wilson-mapreduce-lab/shakespeare-hamlet.txt"
split ms=0.247902

Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems/HW4/mapreduce-lab (main)
$ curl -s -o /dev/null -w "map0 ms=%{time_total}\n" "http://54.237.79.199:8080/map?s3=s3://wilson-mapreduce-lab/mr/chunks/shakespeare-hamlet_20260208T225253Z_chunk00.txt"
map0 ms=0.148496
$ curl -s -o /dev/null -w "map1 ms=%{time_total}\n" "http://54.237.79.199:8080/map?s3=s3://wilson-mapreduce-lab/mr/chunks/shakespeare-hamlet_20260208T225253Z_chunk01.txt"
map1 ms=0.150162
$ curl -s -o /dev/null -w "map2 ms=%{time_total}\n" "http://54.237.79.199:8080/map?s3=s3://wilson-mapreduce-lab/mr/chunks/shakespeare-hamlet_20260208T225253Z_chunk02.txt"
map2 ms=0.127319

Owner@DESKTOP-5FCT03U MINGW64 ~/Documents/Audacity/Building-Scalable-Distributed-Systems/HW4/mapreduce-lab (main)
$ curl -s -o /dev/null -w "reduce ms=%{time_total}\n" "http://50.19.146.108:8080/reduce?in=s3://wilson-mapreduce-lab/mr/maps/mr_chunks_shakespeare-hamlet_20260208T225253Z_chunk00_txt_20260208T225519Z.json&in=s3://wilson-mapreduce-lab/mr/maps/mr_chunks_shakespeare-hamlet_20260208T225253Z_chunk01_txt_20260208T225520Z.json&in=s3://wilson-mapreduce-lab/mr/maps/mr_chunks_shakespeare-hamlet_20260208T225253Z_chunk02_txt_20260208T225521Z.json"
reduce ms=0.275870
```

MapReduce Lab: Split / Map / Reduce Latency (single run)



This shows that splitting and reducing take the most time, while the three map tasks each run faster and at about the same speed, which proves the work is being divided and processed in parallel.

Results

What happen if one of the mapper failed? How would you recover?

The final reduce would be missing that mapper's output, so I'd re-run that mapper on the same chunk until it successfully writes its JSON to S3, then run reduce again with all the outputs.

How can you scale this system into 10 or 100 mappers?

I'd split the file into 10/100 chunks and run 10/100 mapper tasks in parallel, then have the reducer read all the mapper output URLs from S3 (or a list the splitter writes) instead of hardcoding just three.

What was the challenging part of coordinating tasks manually?

Keeping track of all the moving pieces (IPs, S3 paths, correct query params, and the right order of split -> map -> reduce) without making a tiny typo that breaks everything.

What to hand in and when!

Did you get a speed up? How much?!

Yes, I did get a speed up because the three mapper tasks ran in parallel instead of one after the other, so the total map phase time was about the time of the slowest mapper (~0.15s) instead of the sum of all three (~0.43s), which gives roughly a 3x speedup compared to running them sequentially.