DISTRIBUTED FILE SYSTEM ON INTERNET

基于互联网的小型分布式文件系统

组长：罗丽薇 组员：邱子悦 袁一玮 余致远

INTRODUCTION

简介

CLOUD STORAGE

- 云存储需求
  - 多设备同步
  - 备份重要数据

商业云存储

- 内容遭审查
- 令人窒息的带宽

OUR PROJECT

- 基于互联网
- 易于部署
- 利用闲置硬盘/带宽
- 避免审查
- 提高文件安全系数

INVESTIGATION

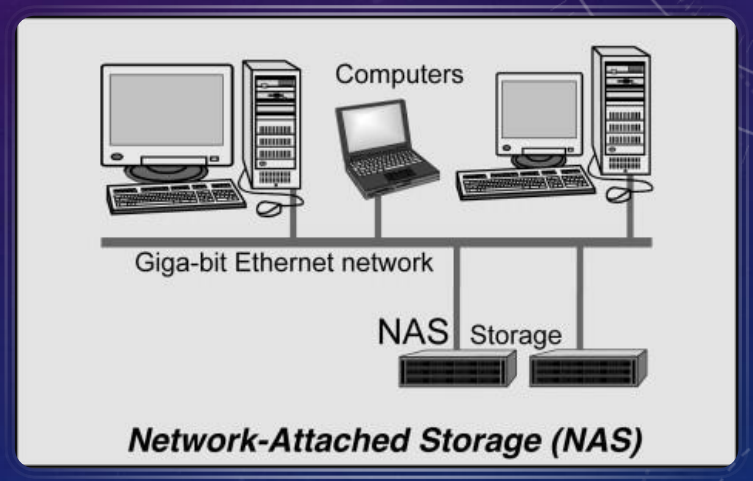
背景 & 调研

IPFS - INTERPLANETARY FILE SYSTEM

- 分布式存储
- 基于区块链
- P2P
- 哈希码寻址
- Filecoin 激励

NAS - NETWORK ATTACHED STORAGE

- 整合分布数据
- 家用网盘



小结

	IPFS	Our FS
数据安全性	无身份验证，拿到哈希值就能拿到文件	目录节点可进行身份验证，用户只能访问自己的文件
数据可靠性	无法保证足够可用源	目录节点可协调冗余备份，保证备份充足

	NAS	Our FS
设备限制	专有设备	普通设备运行客户端即可，跨平台兼容
扩容限制	单台扩容有限，多台难以无缝合并	无缝扩展，无上限

SENIORS' WORK

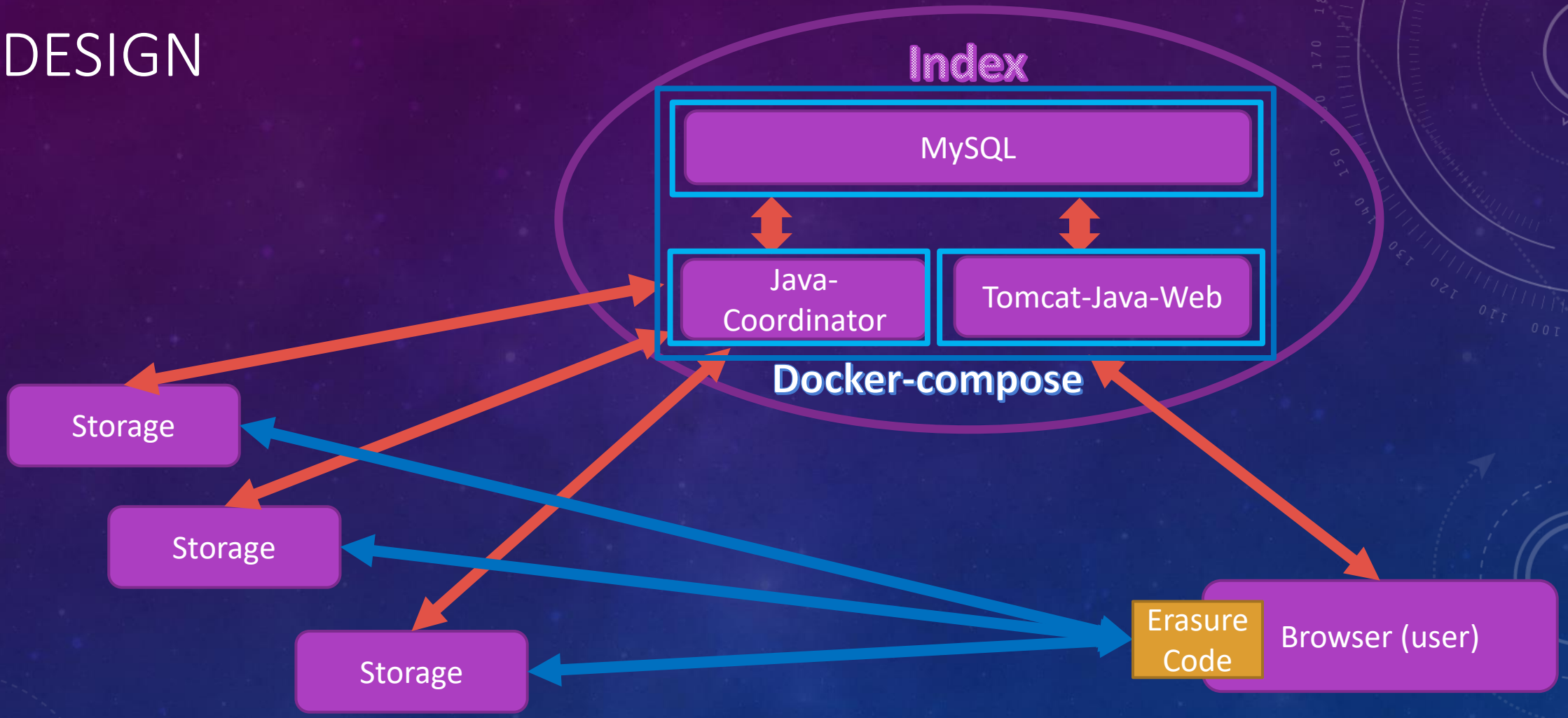
- 不利：
  - 中心化的数据传输
  - 服务器配置繁杂
- 借鉴 idea：
  - Java 客户端实现跨平台
  - Web 管理界面
  - 纠删码冗余

Repo: https://github.com/IngramWang/DFS_OSH2017_USTC

STRUCTURE

项目结构

DESIGN

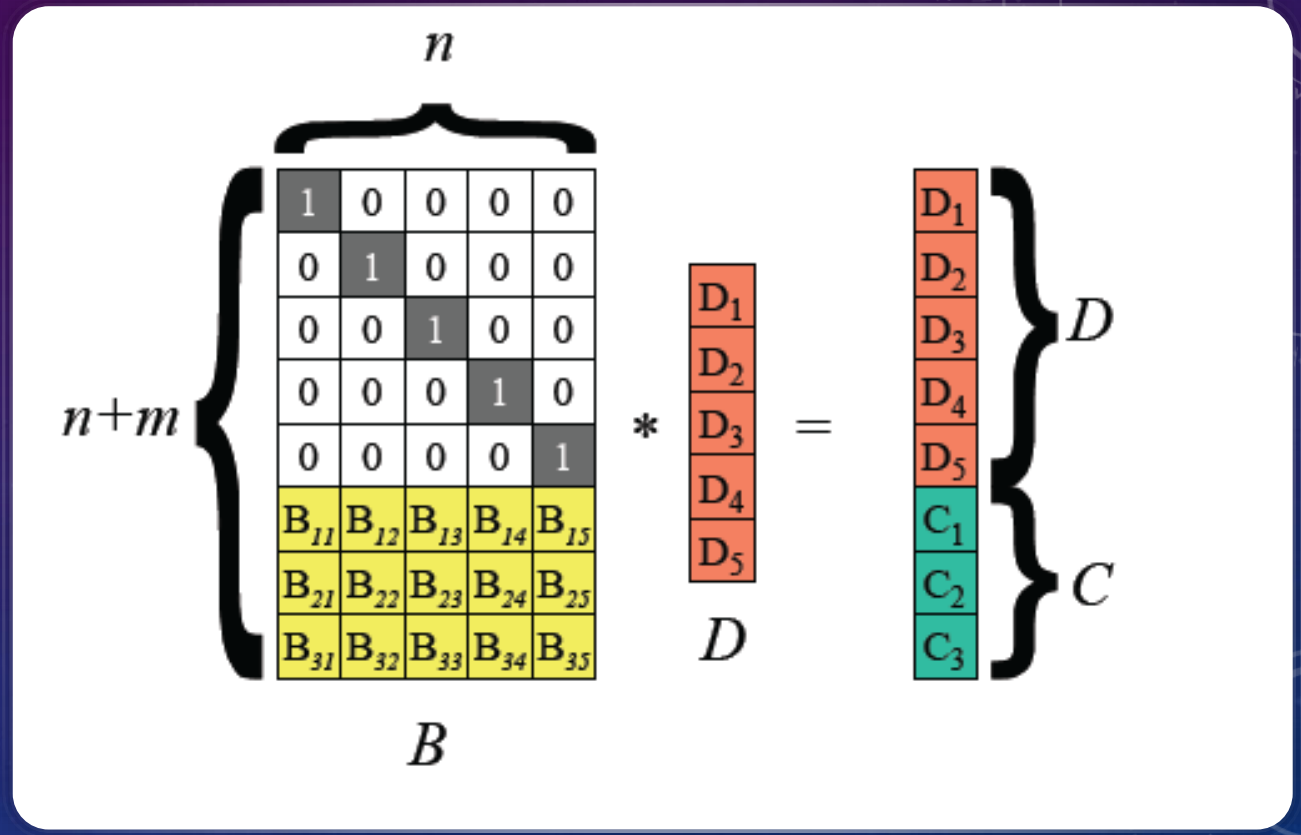


Repo：https://github.com/OSH-2020/x-dontpanic

SCALING

STABILITY

- Erasure code



STABILITY

- 丢失概率低
- 减少额外空间开销

设单个存储节点故障率为 p。使用纠删码时文件分成 n 块并冗余 m 块，$n+m$ 块分给不同的节点。备份冗余数量为 k，分别放在不同节点。

单点存储丢失概率：p

副本冗余丢失概率：p^k

纠删码冗余丢失概率：

$$\sum_{i=m+1}^{n+m} \binom{n+m}{i} p^i (1-p)^{n+m-i}$$

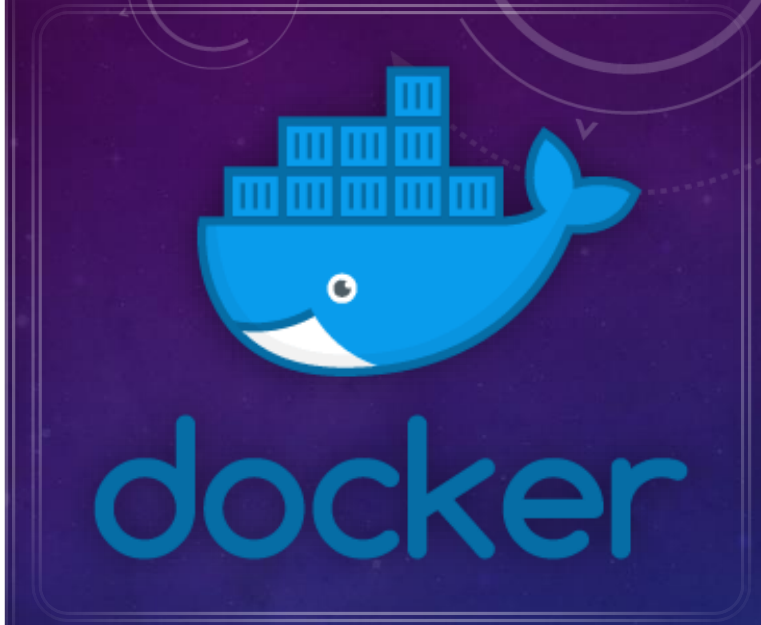
$p=1\%, n=5, m=5$：

纠删码冗余存储丢失概率：2.03e-10

达到同样概率所需备份数量：5

纠删码节约空间：file size*3

Java



Docker

Web

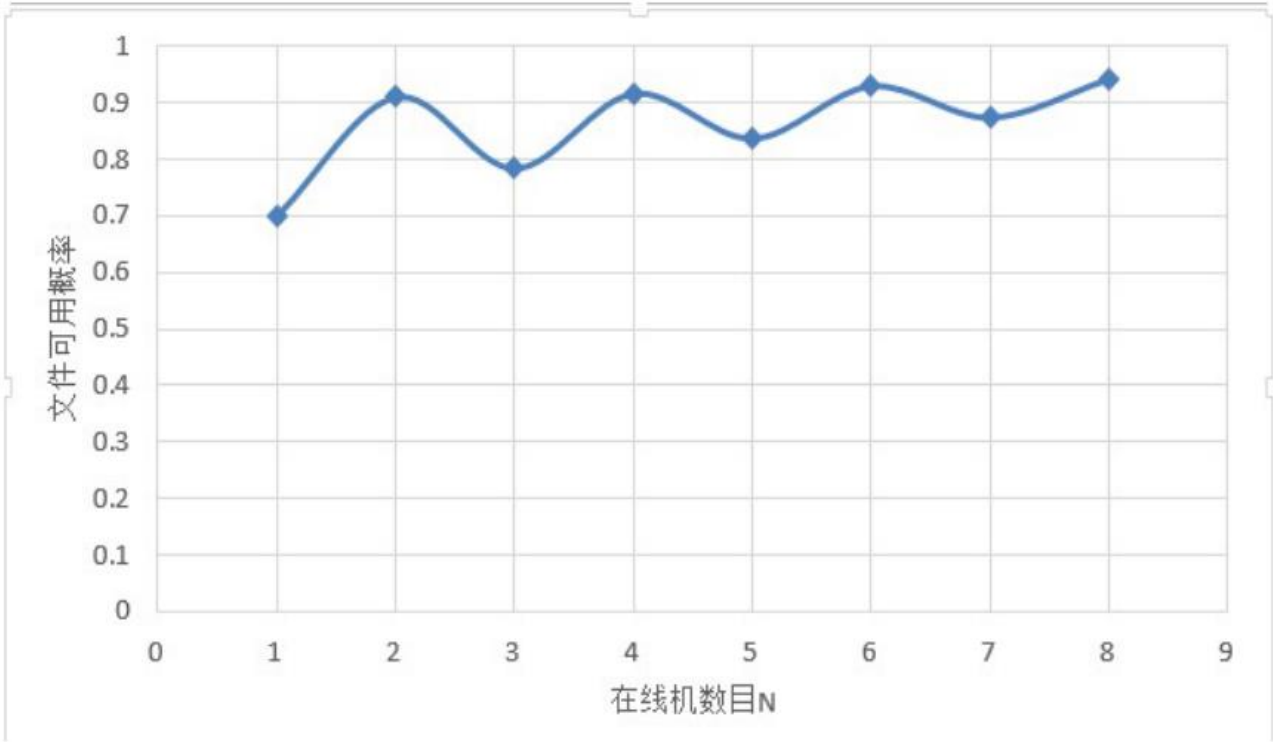
HANDY & COMPATIBILITY

USABILITY

- 个人设备非 24h 在线
- 碎片分配策略
- 在线时间重合度

如果上传文件时在线客户机数目为n，文件可用的概率即n个客户机中至少n/2个客户机在线的概率，为$\sum_{i=n/2}^{n} C_n^i p^i (1-p)^{n-i}$。

下为p=70%时文件可用概率与n的关系图：



TECHNIQUE

相关技术

WEB ASSEMBLY

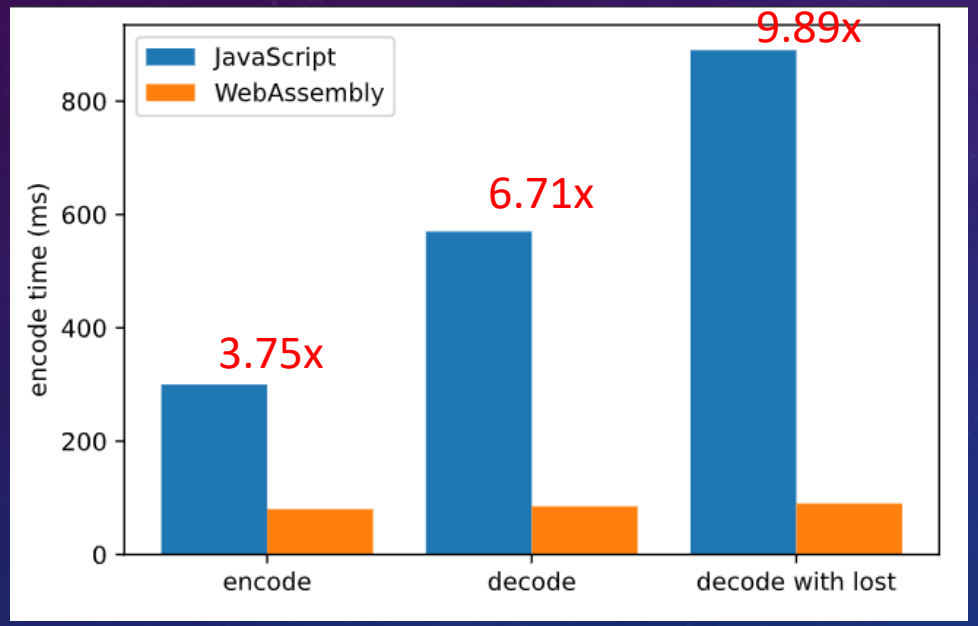
- JavaScript: https://github.com/ianopolous/ErasureCodes

Go -> WebAssembly: https://github.com/klauspost/reedsolomon

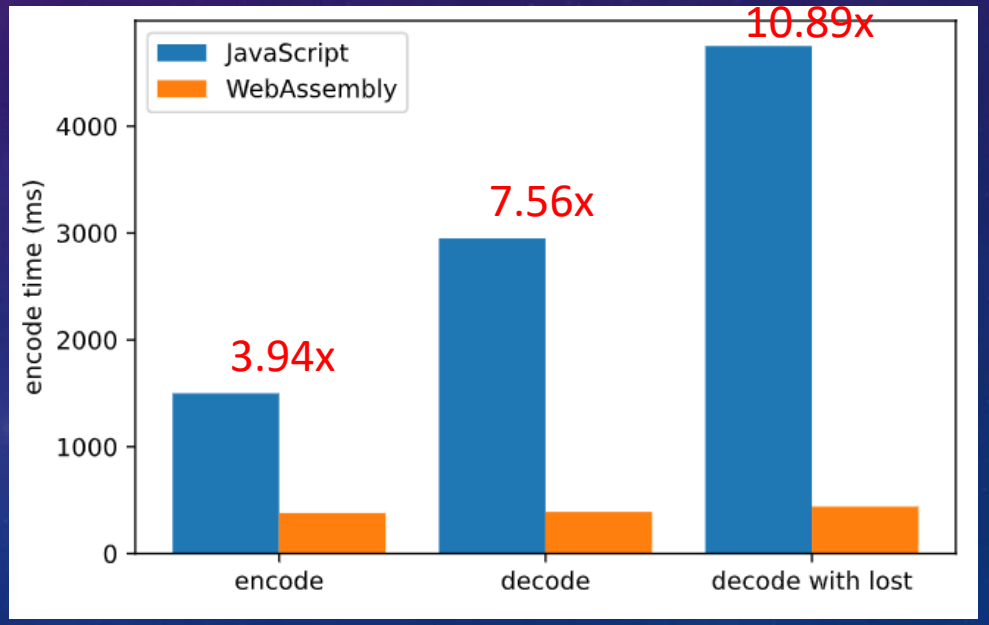
VS

WEB ASSEMBLY

File size: 1MB, division: 40+20

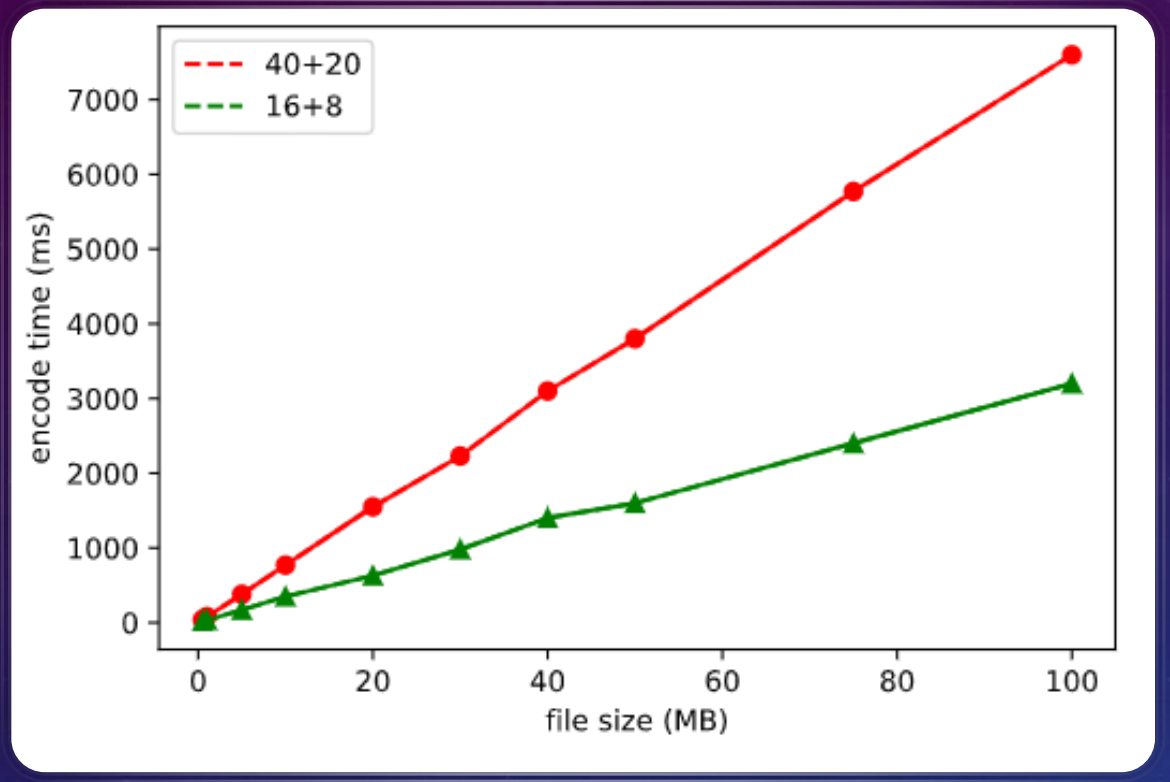


File size: 5MB, division: 40+20




FINISH
JS
WA

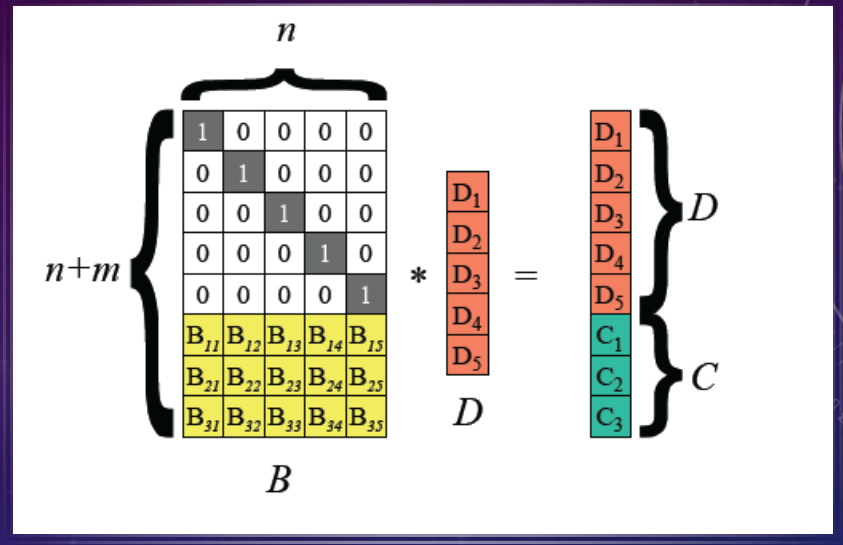
WEB ASSEMBLY

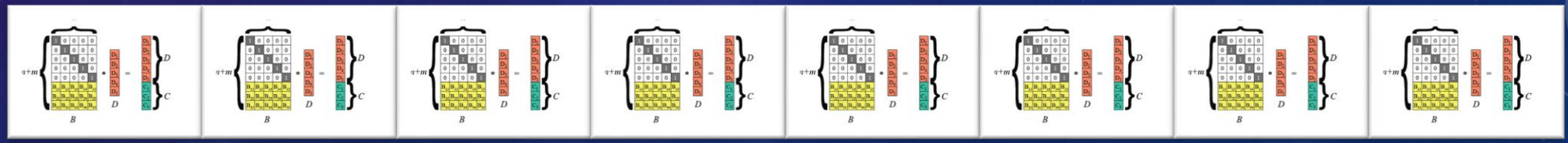


- 40+20: ~76ms/MB or ~13MB/s
- 16+8: ~33ms/MB or ~30MB/s

WEB ASSEMBLY

- 线性关系：编码时间 & 文件大小
- 并行编码
- 流水作业



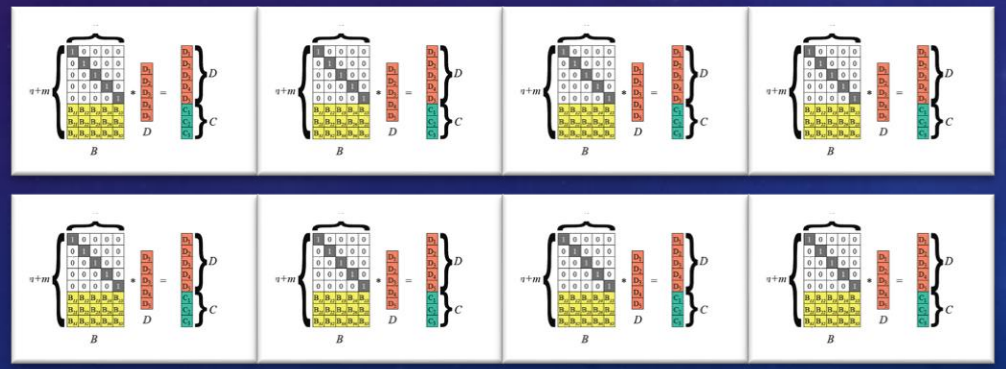


WEB ASSEMBLY

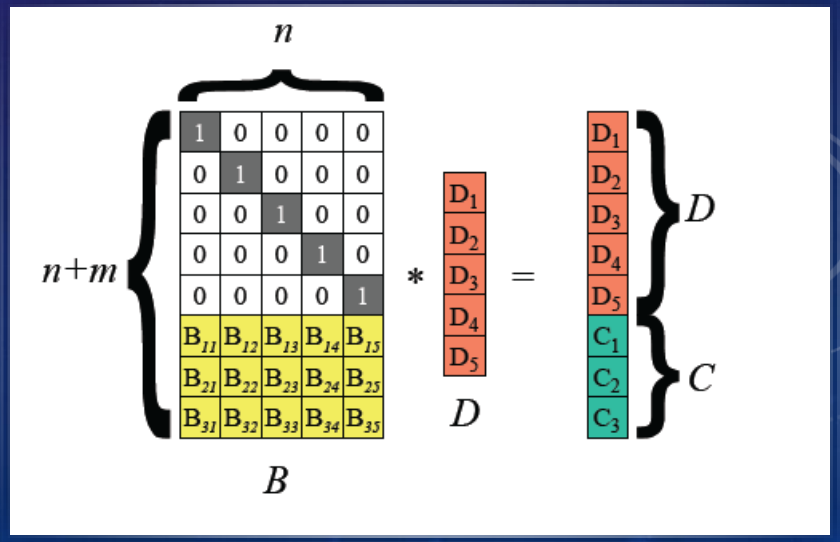
- 线性关系：编码时间 & 文件大小
- 并行编码
- 流水作业

Worker 1:

Worker 2:

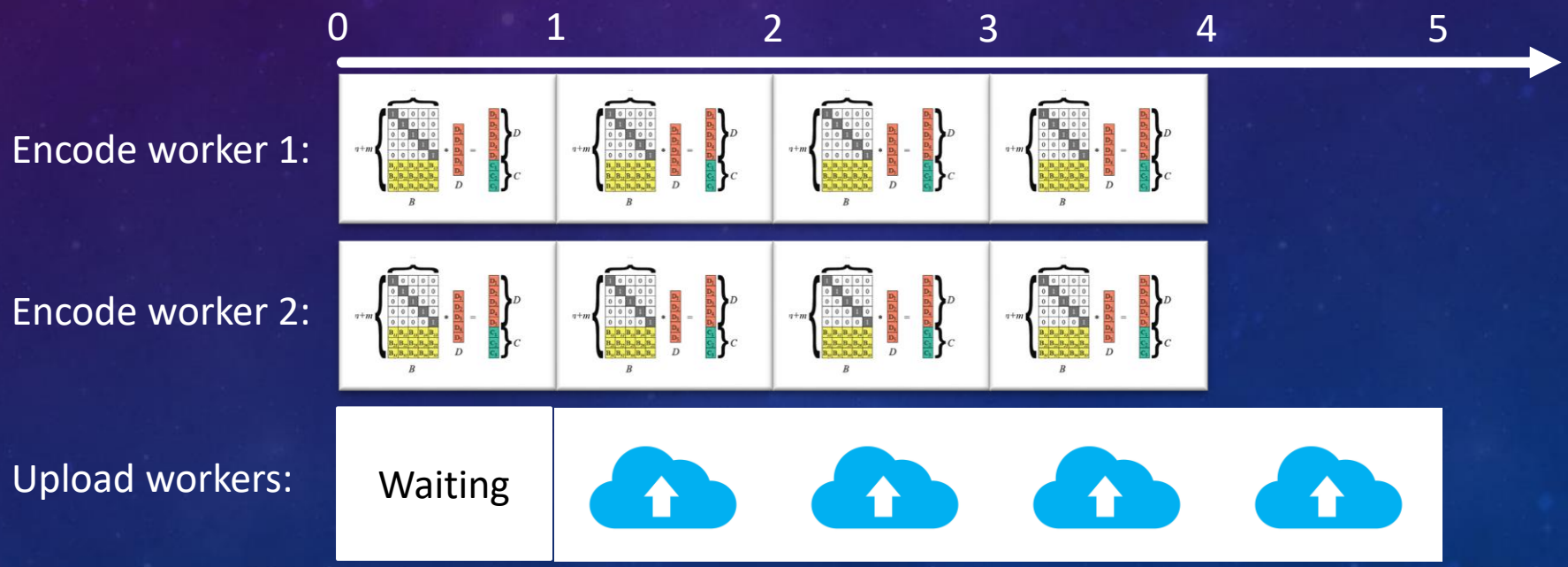


$$= \frac{1}{2} \times$$



流水作业

- 40+20: ~13MB/s
- 16+8: ~30MB/s
- （降低分块参数，吞吐还能更高）
- 千兆网最大速率：1024 Mbit/s = 128 MB/s



传统部署方案

服务器配置

可选使用/不使用 docker。

tomcat+mysql+server.jar（不推荐）

以下安装以 ubuntu 为例

tomcat

用途：web 后端

安装 tomcat：

- 给 tomcat 一个工作（如 "/opt/to
- 从 tomca
- 创建 tom（[illegible]了 tomcat 用）[illegible] tomcat:tomcat 工作

配置 tomcat（参考原项目配置文

使用以下命令查 va 版本

```
sudo update-java-alternatives -l
```

输出可能是这样的：

```
java-1.8.0-[illegible]/lib/jvm/java-[illegible]openjdk-amd64
```

你的 JAVA_HOME [illegible]加这条

JAVA_HOME=[illegible]openjdk-

利用以上信息[illegible]

```
$sudo vim /et[illegible]cat.se
```

复制以下信息到[illegible]Java 位[illegible]间，记得[illegible]JAVA_HO

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target
[Service]
Type=forking
Environment=JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64/jre
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'
Environment='JAVA_OPTS=-Djava.awt.headless=true -
Djava.security.egd=file:/dev/./urandom'
ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh
User=tomcat
Group=tomcat
UMask=0007
RestartSec=10
Restart=always
[Install]
WantedBy=multi-user.target
```

重新载入配置：

```
$sudo systemctl daemon-reload
```

启动 tomcat：

```
$sudo systemctl start tomcat
```

确认没有错误发生：

```
$sudo systemctl status tomcat
```

此时浏览器打开 localhost:8080 即可看到 tomcat 默认页面。

放通 8080 端口，具体操作视机器防火墙决定。此时远程访问服务器的 8080 端口，即可看到 tomcat，默认界面。

设置开机启动 tomcat：

```
sudo systemctl enable tomcat
```

mysql

用途：数据库

安装 mysql：`sudo apt install mysql-server`

导入数据表：`mysql -u root -p <mysql.sql`

（密码暂时写死为 201314）

server.jar

用途：接受 client 的控制连接，维护数据库中的节点状态

先安装 JDK。安装配置 openjdk-8 的方法如下（参考自原项目配置文档）：

```
安装 openjdk-8-jdk:
$ sudo apt-get install openjdk-8-jdk

查看 java 版本:
$ java -version

编辑/etc/profile:
$ sudo vim /etc/profile

在文件尾添加 java 环境变量:
export JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64/jre/bin"
```

运行：

```
java -jar server.jar
```

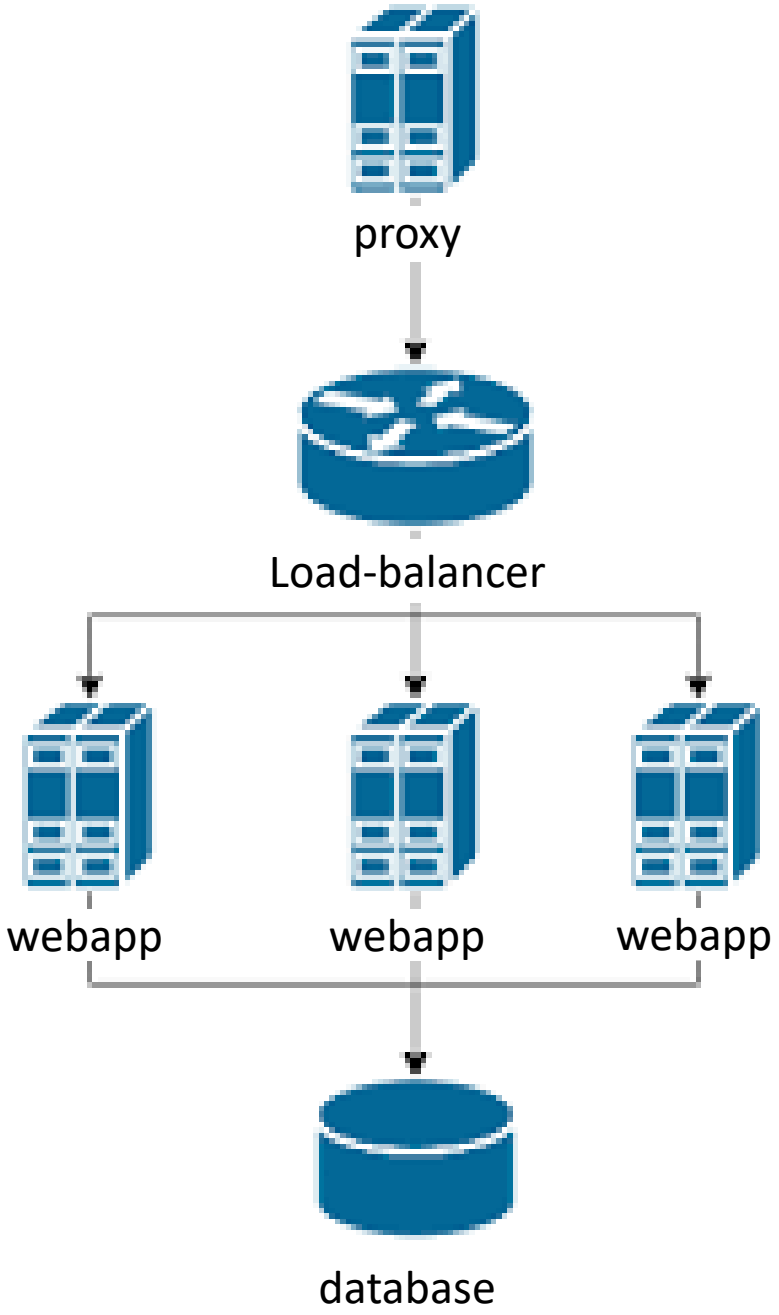
VS

Docker-compose方案

docker-compose （推荐）

以下使用以 Arch 为例

- 安装 docker 包或是 AUR 中的 docker-git 包及 docker-compose 包
- （推荐）添加自己的用户到 docker user group。
- （推荐）在 `/etc/docker/daemon.json` 加入一些 docker hub 镜像。
- （推荐）复制 ./src/docker 到一个新的位置，假设文件名为 panic。要确保服务所需要的网络端口已开启。
- 在主目录（也就是 docker-compose.yml 文件存在的目录）下输入 `docker-compose build`，即可开始构建服务。
- 使用 `docker-compose up` 即可启动服务，若不想看到服务的命令输出部分，可以加上 `-d` 参数即 `docker-compose up -d` 来分离日志输出。
- 若要停止服务，在终端里键入 `docker-compose down` 即可停止并删除容器，容器的数据（MySql 的数据表和 TOMCAT 的 webapps）均已进行数据持久化，不会因为删除容器而丢失所有数据。

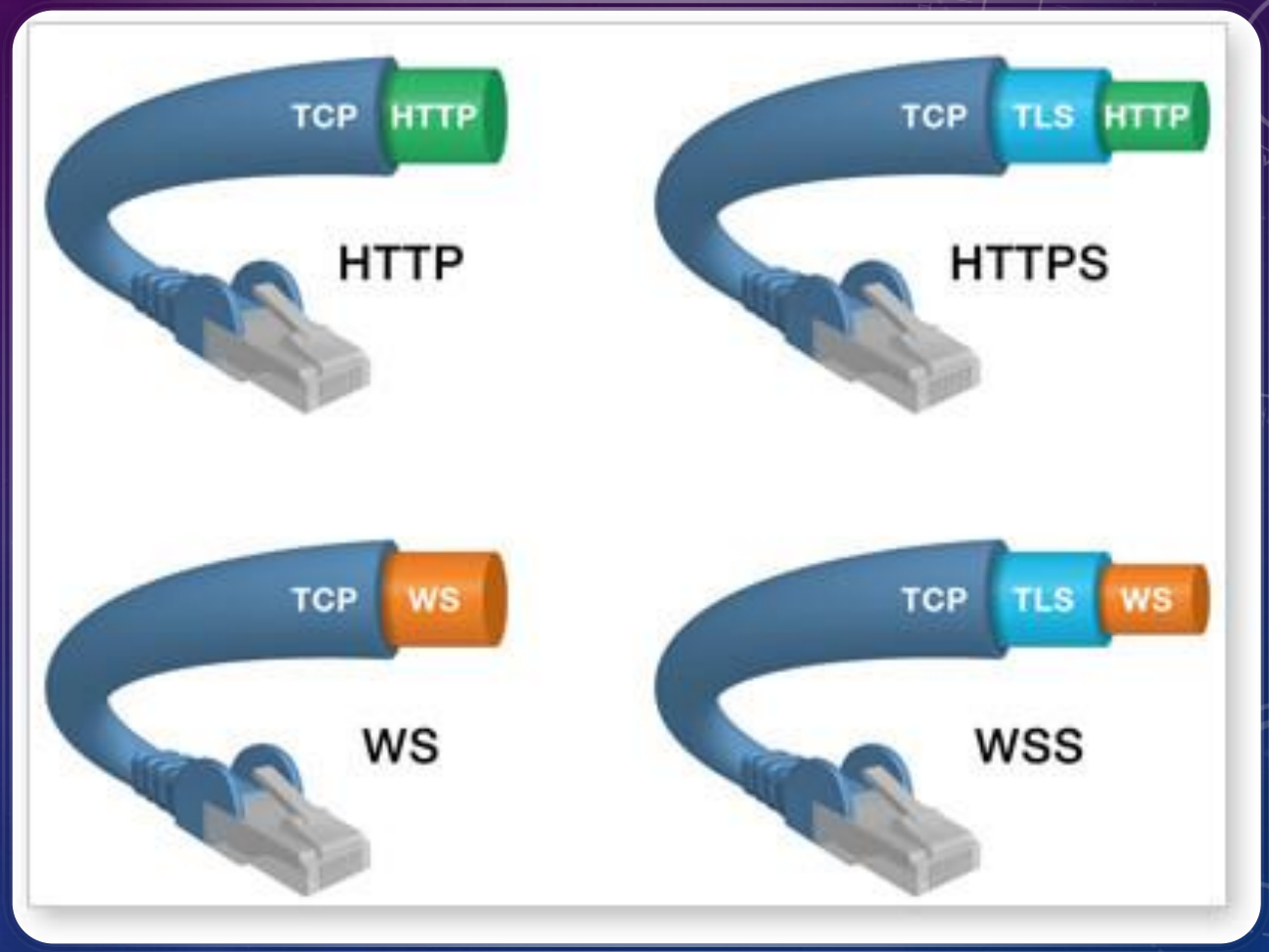


DOCKER-COMPOSE

- 并发
- 扩容
- 负载均衡

WEBSOCKET

- 建立在 TCP 上
- 兼容 HTTP
- 通信高效



WEBSOCKET

- 建立在 TCP 上
- 兼容 HTTP
- 通信高效

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

PROTOTYPE

我们的实现

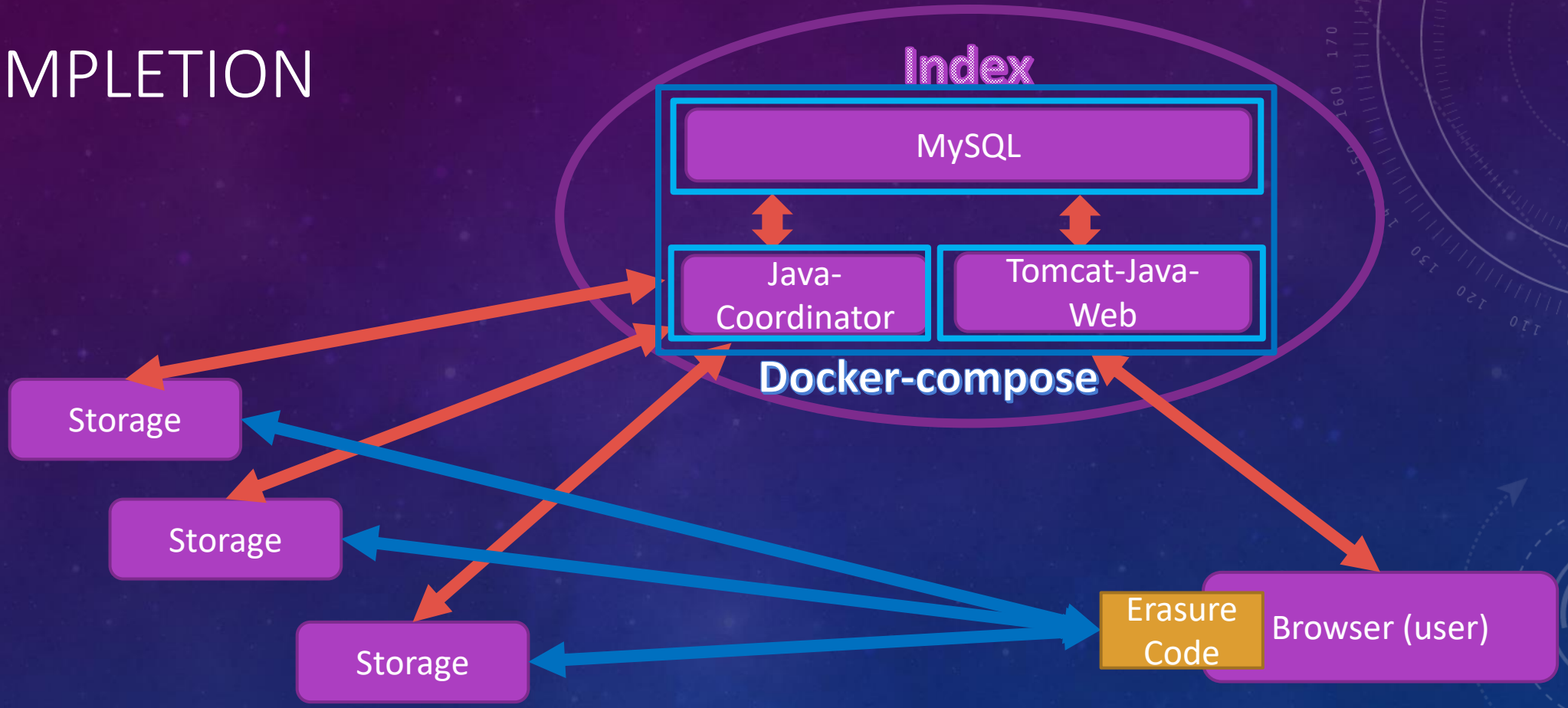
ALLOCATION

[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]

[0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]

- 长度为 24 的 0/1 向量
- 覆盖上传者在线时间段 x% 以上
- 两个向量做 and 操作，得到的向量中 1 的个数占上传者向量中 1 的个数的 x%
- X=70, n=4, m=4, 成功率=94.2%

COMPLETION



Repo：https://github.com/OSH-2020/x-dontpanic

COMPLETION

Repo：https://github.com/OSH-2020/x-dontpanic

DISCUSSION

PROSPECT

- 激励机制
  - 更多用户/更大空间
  - 良性循环
- 工作目录
  - 统一访问方式
  - 实时同步备份

DEBATE

- 目录节点重新承担中转工作？
  - 预约离线下载
  - 代为执行编解码

1+1+1+1=?

彩蛋：烤鸽子

制定合理惩罚措施进行监督

组员	工作
袁一玮	将目录节点需要运行的服务 docker 化，修改 web 前端 UI，制作和演示 demo
罗丽薇	实现浏览器 WebSocket 直连客户端，使用 js worker 调用纠删码模块避免阻塞
余致远	分别研究如何用 js/webassembly 在浏览器实现纠删码模块，给出性能 benchmark
邱子悦	隔离不同用户的目录，cookie 验证，尝试 CI 自动部署，碎片分配策略

彩蛋：烤鸽子

制定合理惩罚措施进行监督

组员	工作
袁一玮	将目录节点需要云[illegible]服务 docker 化，修改 web 前端 UI，制作和演示 demo
罗丽薇	实现浏览器 WebSocket 连客户端，使用 [illegible]ar 调用纠删码模块避免阻塞
余致远	分别研究如何用 js/webassembly 在浏览器[illegible]现纠删码模块[illegible]出性[illegible] benchmark
邱子悦	隔离不同用户的目录，cookie 验证，尝试 CI 自动部署，碎片分配策略

还有谁敢鸽？

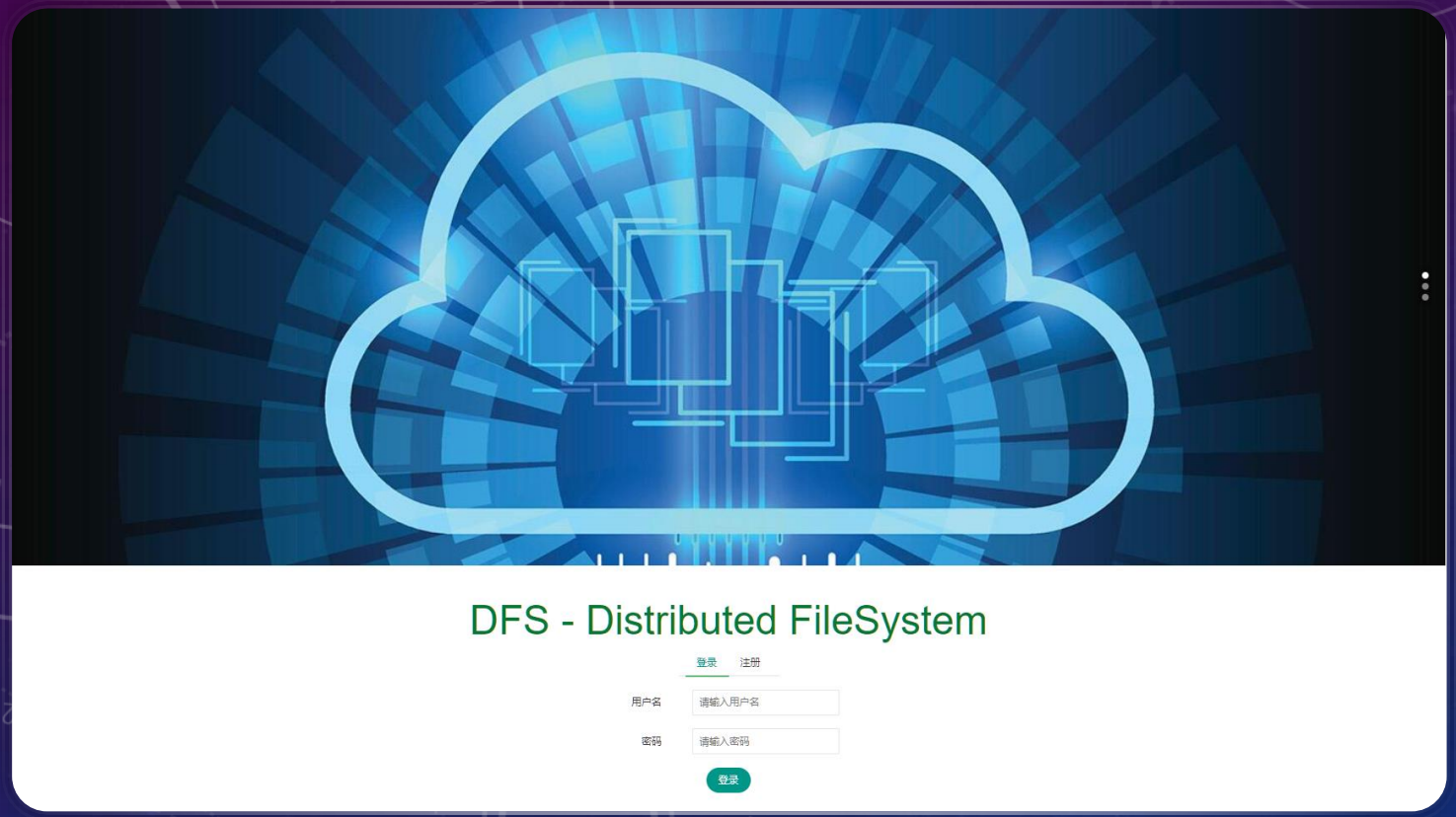
彩蛋：烤鸽子

制定合理惩罚措施进行监督

WHAT WE GAIN?

- 选题
- 设计
- 分锅

WHAT WE GAIN



DEMO

Q&A