

The background features a dark blue gradient with faint, light blue circular patterns. On the left side, there are several concentric circles with degree markings ranging from 40 to 260. Some of these circles have arrows indicating a clockwise direction. The overall aesthetic is technical and modern.

# 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

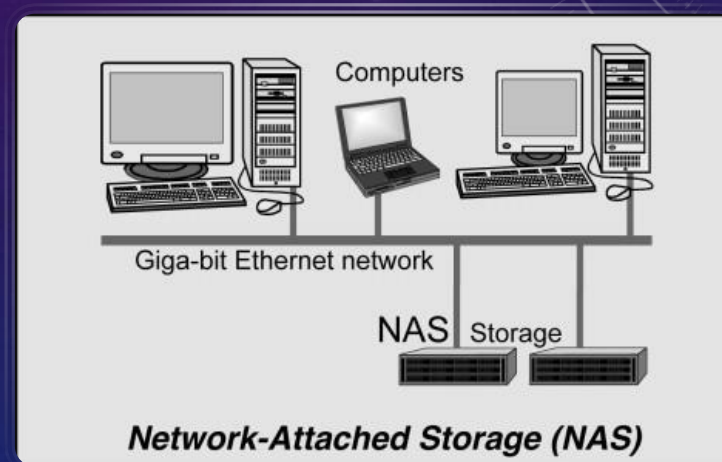
- 分布式存储
- 基于区块链
- P2P
- 哈希码寻址
- Filecoin 激励

A large, dark, translucent cube is centered in the background. The letters 'IPFS' are embossed on the front face of the cube in a large, bold, sans-serif font. The letters have a textured, almost crystalline appearance. The background is dark with faint, glowing geometric patterns, including concentric circles and lines, suggesting a technical or digital theme.

IPFS

# 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	专用设备	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

DFS-私有网盘 优秀的分布式文件管理系统



当前访问位置:

ROOT

文件名	读写权限	修改时间
<input type="checkbox"/> 文件夹 ..		
<input type="checkbox"/> 文件夹 233	rw	20200324
<input type="checkbox"/> 文件夹 TIM	rw	20200322
<input type="checkbox"/> 文件夹 TIM!!	rw	20200325

当前任务进度:

预下载

删除

上传

重命名

提示:

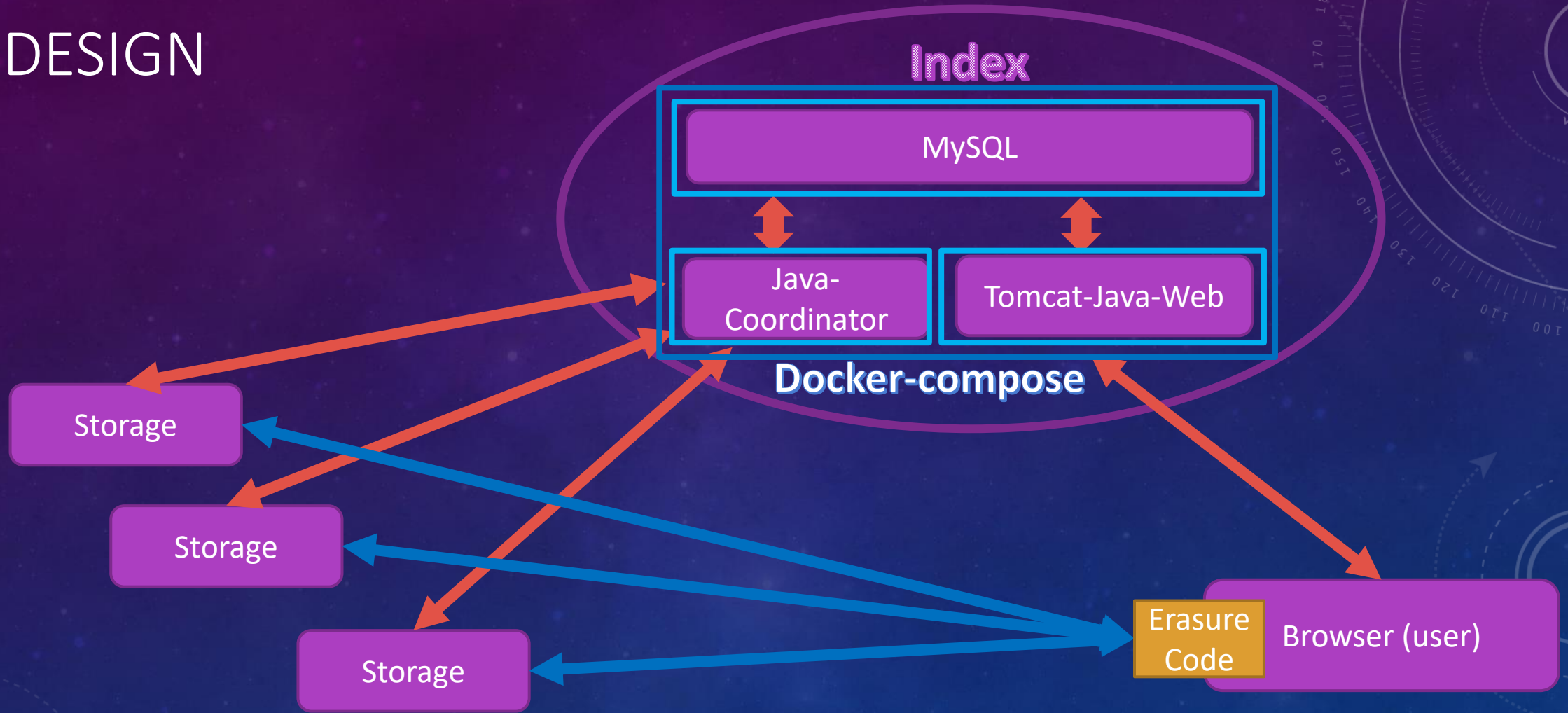
欢迎使用本系统 ~

Repo: [https://github.com/IngramWang/DFS\\_OSH2017\\_USTC](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

$$\begin{array}{c}
 \overbrace{\begin{array}{|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 \\ \hline B_{11} & B_{12} & B_{13} & B_{14} & B_{15} \\ \hline B_{21} & B_{22} & B_{23} & B_{24} & B_{25} \\ \hline B_{31} & B_{32} & B_{33} & B_{34} & B_{35} \\ \hline \end{array}}^n \\
 \left. \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \right\} n+m \\
 B
 \end{array}
 *
 \begin{array}{|c|} \hline D_1 \\ \hline D_2 \\ \hline D_3 \\ \hline D_4 \\ \hline D_5 \\ \hline D \\ \hline \end{array}
 =
 \begin{array}{|c|c|} \hline D_1 \\ \hline D_2 \\ \hline D_3 \\ \hline D_4 \\ \hline D_5 \\ \hline C_1 \\ \hline C_2 \\ \hline C_3 \\ \hline C \\ \hline \end{array}$$



# 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:

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

$p = 1\%, n = 5, m = 5$ :

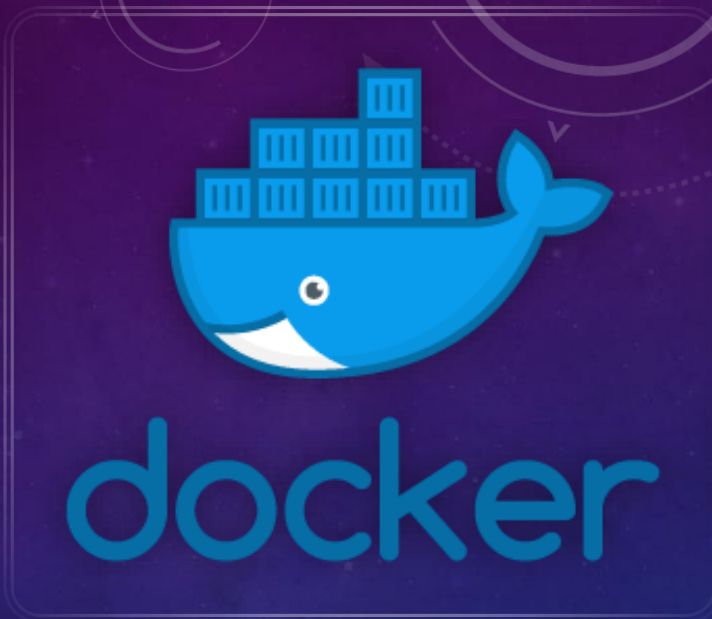
纠删码冗余存储丢失概率:  $2.03e-10$

达到同样概率所需备份数量: 5

纠删码节约空间:  $\text{file size} \times 3$



Java



Docker



Web

HANDY & COMPATIBILITY

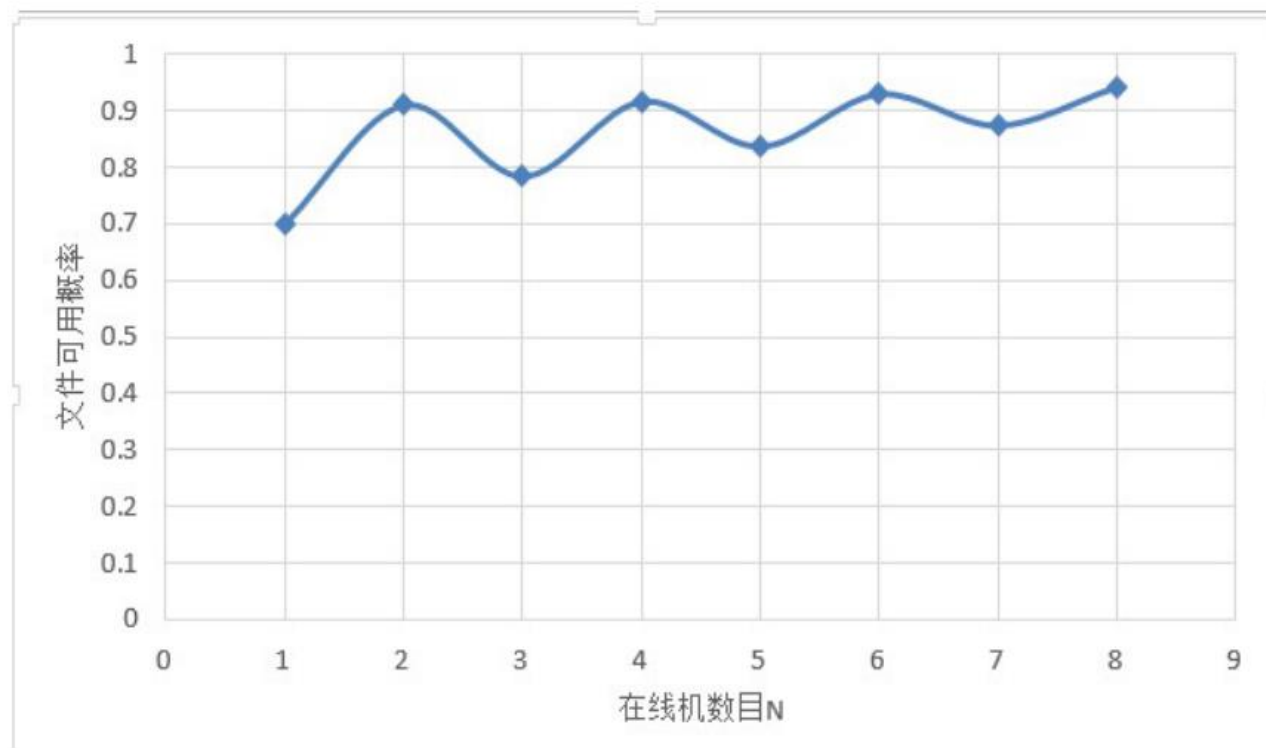


# USABILITY

- 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/ianopolo  
us/ErasureCodes](https://github.com/ianopolo/us/ErasureCodes)



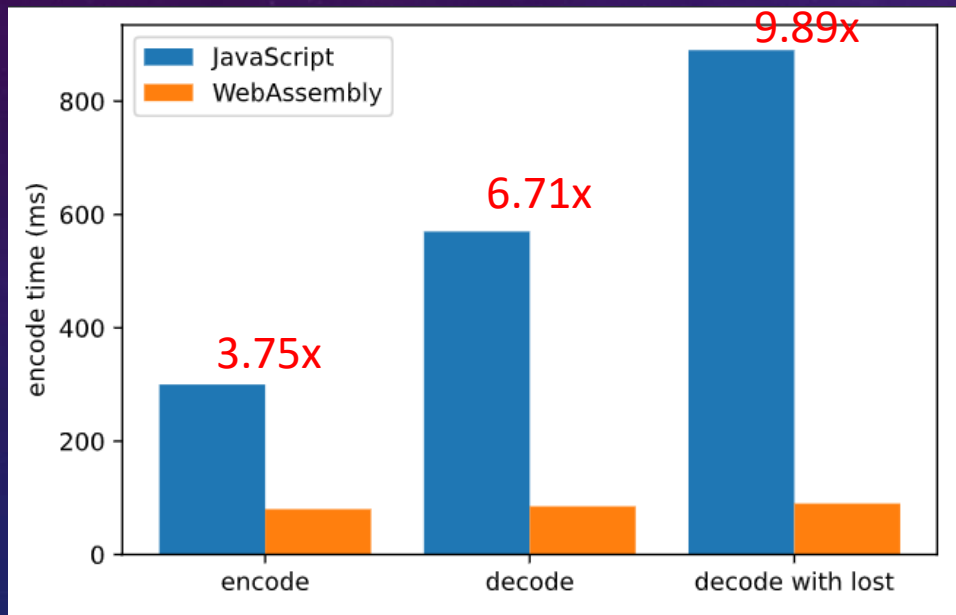
VS

Go -> WebAssembly:  
[https://github.com/klauspost/r  
eedsolomon](https://github.com/klauspost/reedsolomon)

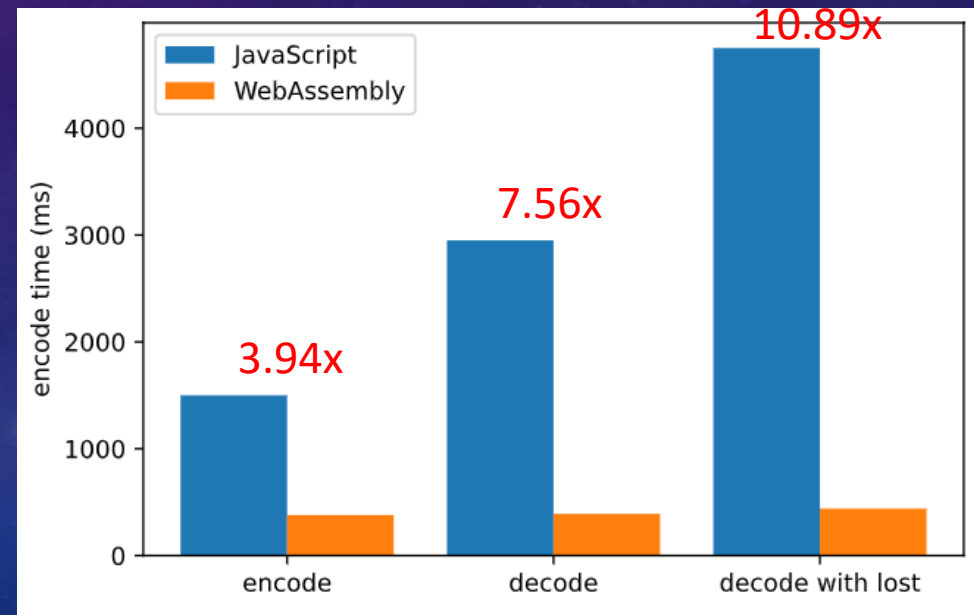


# WEB ASSEMBLY

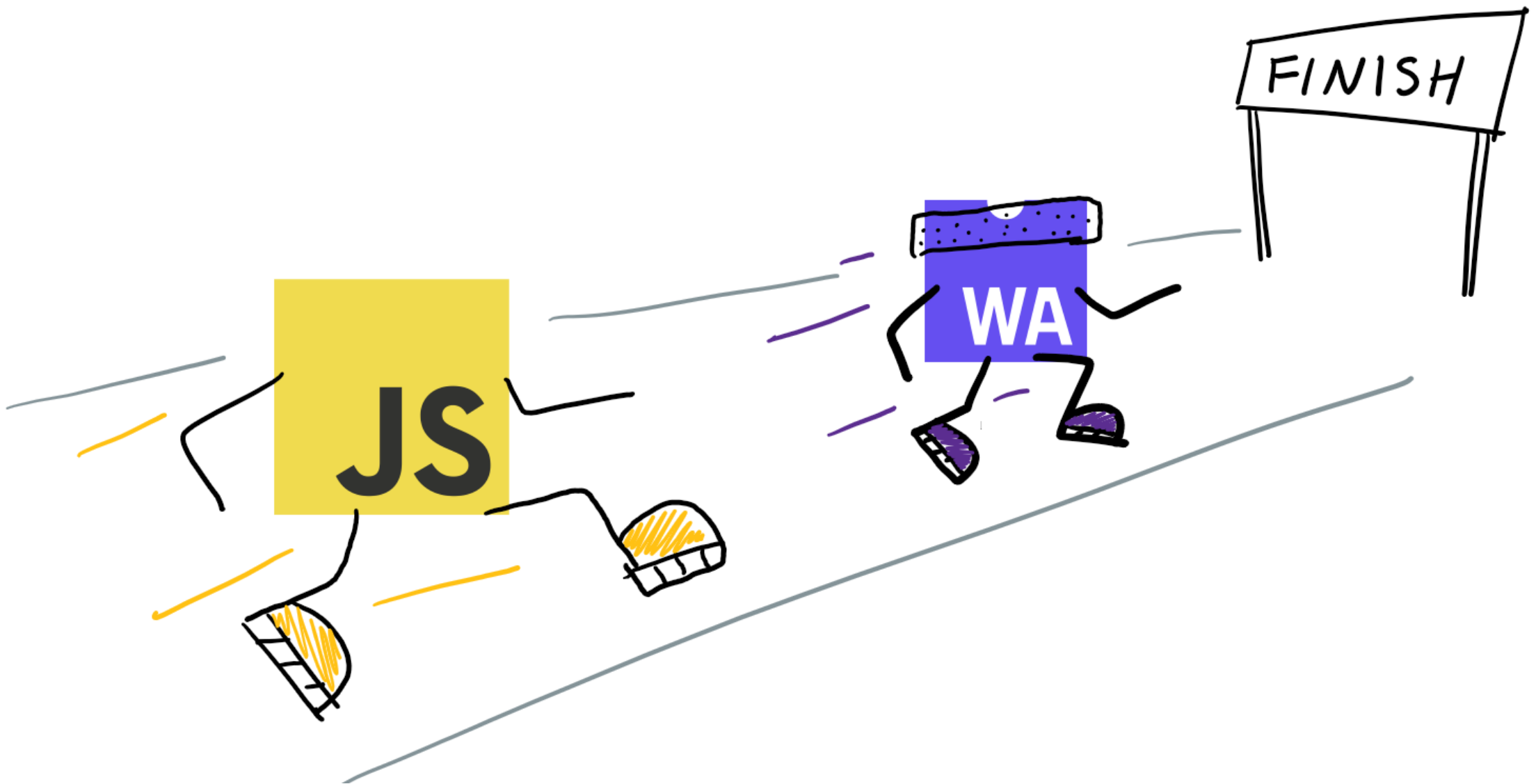
File size: 1MB, division: 40+20



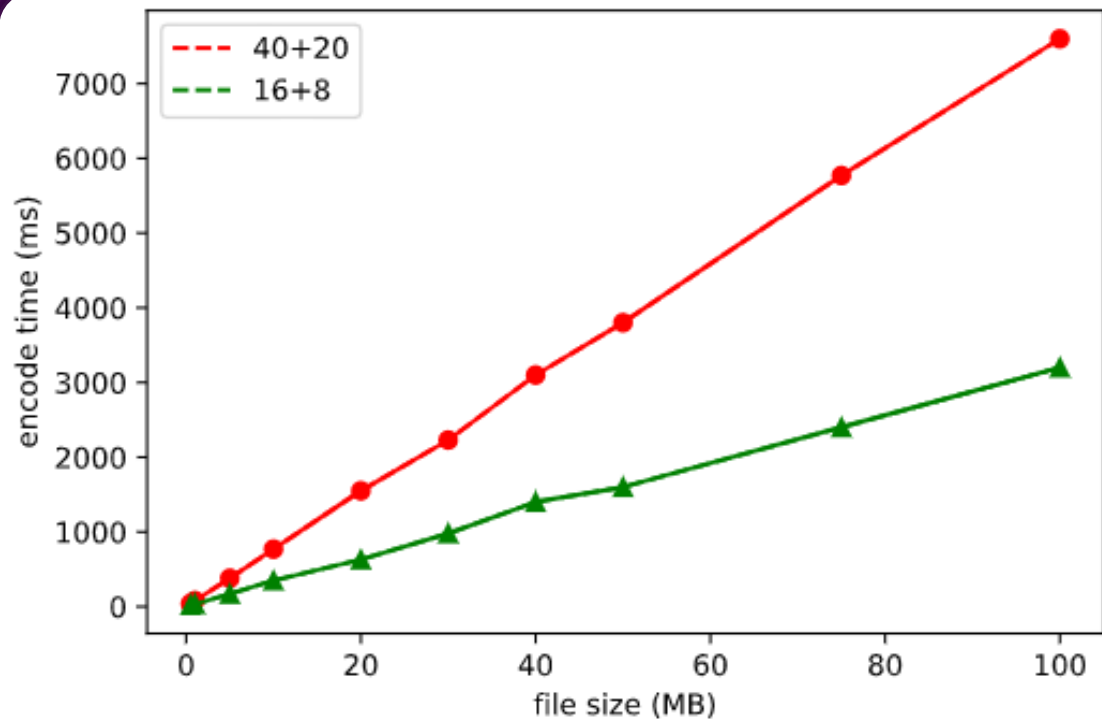
File size: 5MB, division: 40+20







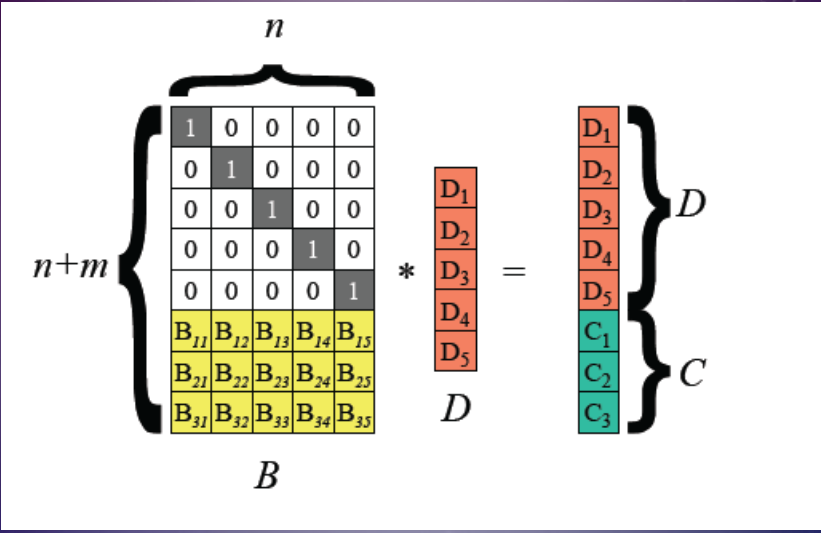
# WEB ASSEMBLY



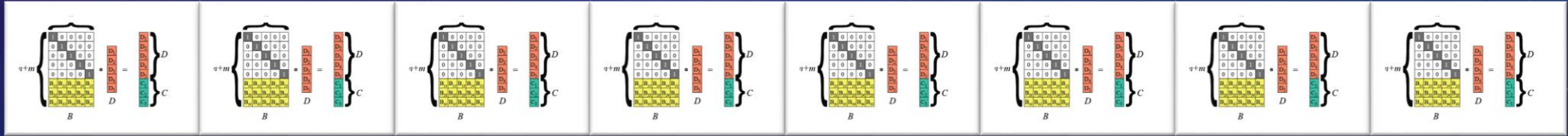
- 40+20: ~76ms/MB or ~13MB/s
- 16+8: ~33ms/MB or ~30MB/s

# WEB ASSEMBLY

- 线性关系：编码时间 & 文件大小
- 并行编码
- 流水作业



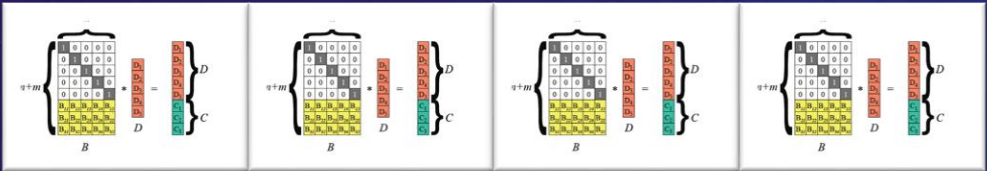
||



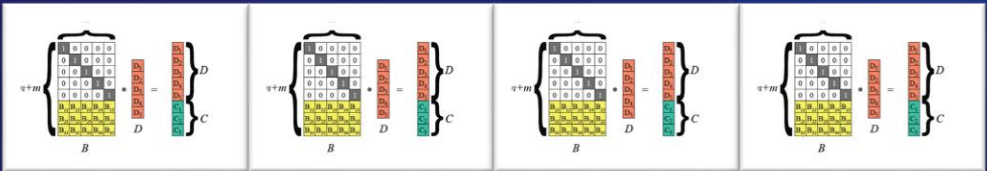
# WEB ASSEMBLY

- 线性关系：编码时间 & 文件大小
- 并行编码
- 流水作业

Worker 1:



Worker 2:



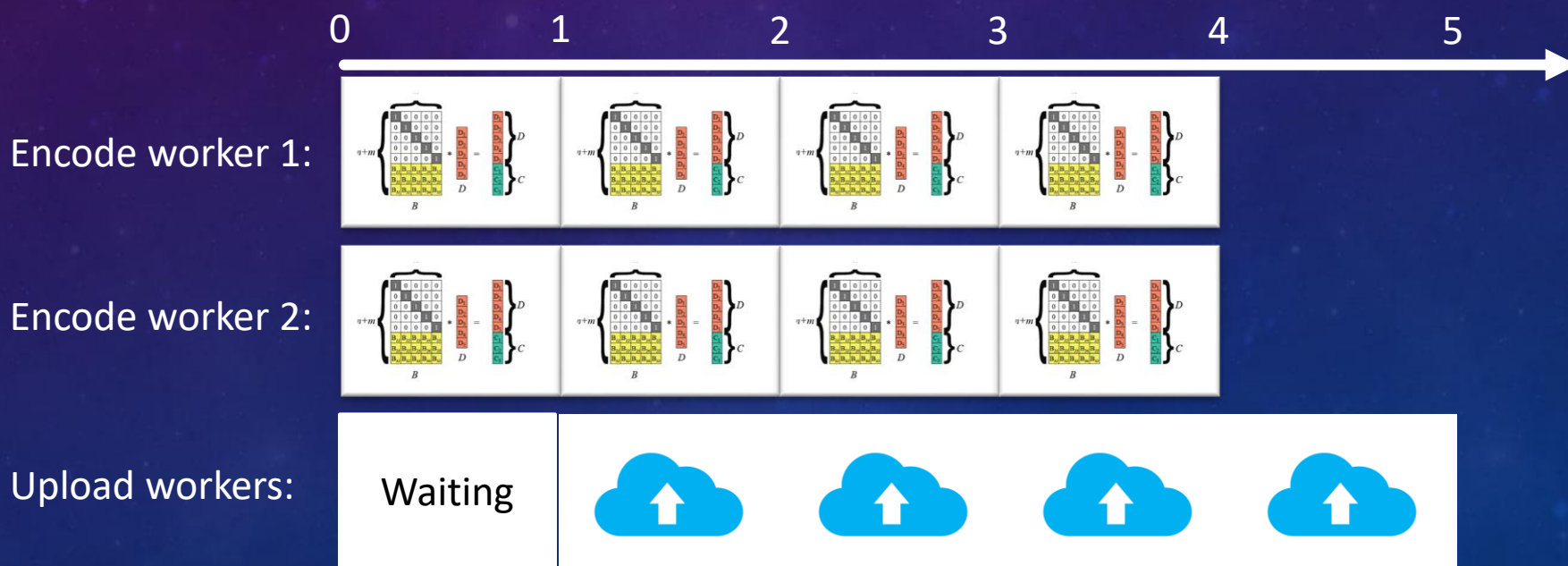
$$= \frac{1}{2} \times$$

$$\begin{array}{c}
 \begin{array}{c} n \\ \hline \begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ B_{11} & B_{12} & B_{13} & B_{14} & B_{15} \\ B_{21} & B_{22} & B_{23} & B_{24} & B_{25} \\ B_{31} & B_{32} & B_{33} & B_{34} & B_{35} \end{array} \\ n+m \\ \hline B \end{array} \\
 \end{array}
 \begin{array}{c}
 \begin{array}{c} D_1 \\ D_2 \\ D_3 \\ D_4 \\ D_5 \end{array} \\
 \hline D
 \end{array}
 =
 \begin{array}{c}
 \begin{array}{c} D_1 \\ D_2 \\ D_3 \\ D_4 \\ D_5 \\ C_1 \\ C_2 \\ C_3 \end{array} \\
 \hline \begin{array}{c} D \\ C \end{array}
 \end{array}$$



# 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+serverjar (不推荐)

以下安装以 ubuntu 为例

tomcat

用途: web 后端

安装 tomcat:

- 给 tomcat 设置环境变量
- 从 tomcat 官网下载 tomcat 压缩包
- 创建 tomcat 用户并设置权限

配置 tomcat (参考 tomcat 官方文档)

使用以下命令安装 java 版本 8

sudo update-alternatives --install

输出可能像这样的:

java-1.8.0-openjdk-amd64

你的 JAVA\_HOME 是:

JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre

利用以上信息设置

\$sudo vim /etc/profile

复制以下信息到文件末尾 (记得在文件末尾添加 JAVA\_HOME)

```
[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="-Xms128M -Xmx1024M -server -XX:+UseParallelGC"
Environment="JAVA_OPTS=-Djava.net.preferIPv4Stack=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。

开放 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。

设置开机自动 tomcat:

sudo systemctl enable tomcat

mysql

用途: 数据库

安装 mysql: sudo apt install mysql-server

导入数据库: mysql -u root -p mysql.sql

(密码暂时写死为 201314)

serverjar

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

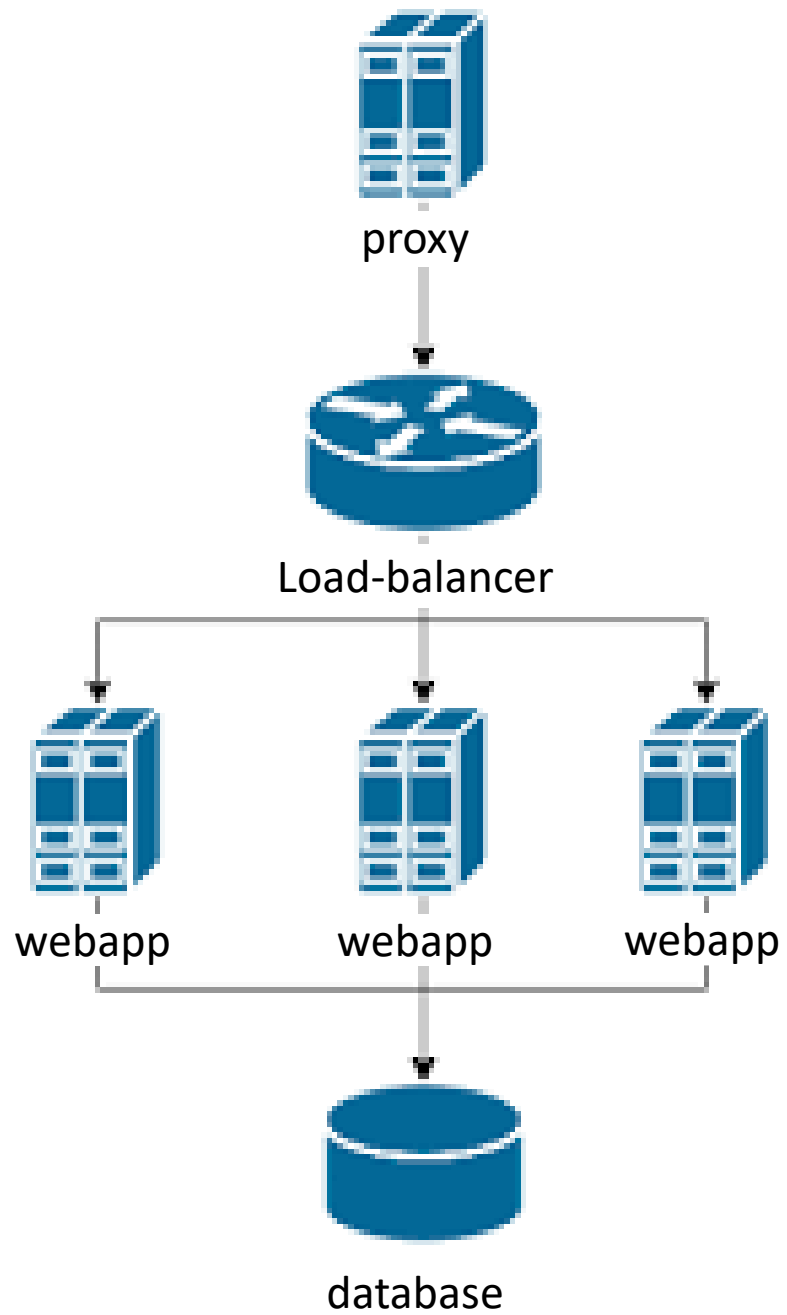
VS

## docker-compose (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 方案

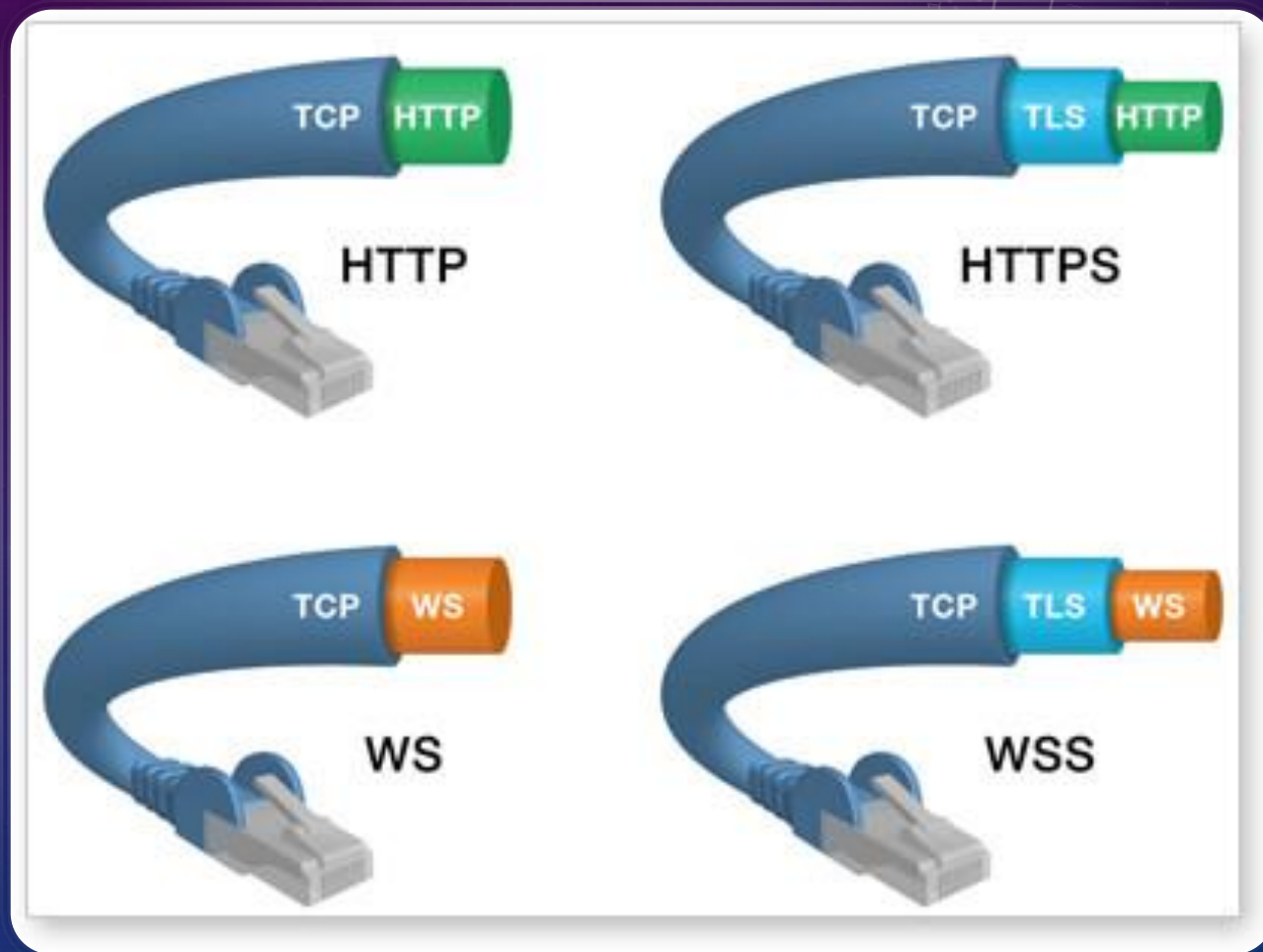


# DOCKER-COMPOSE

- 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

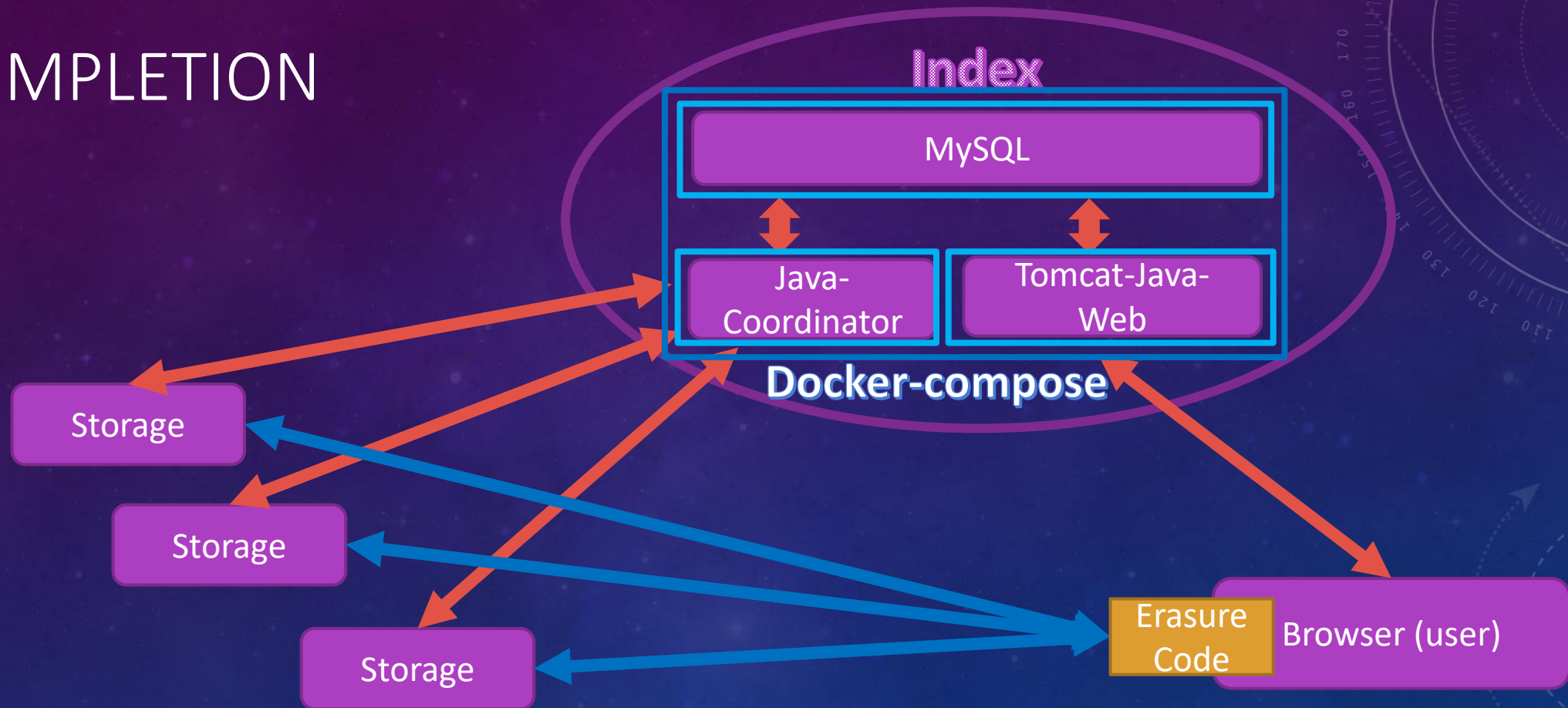
# ALLOCATION

[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]

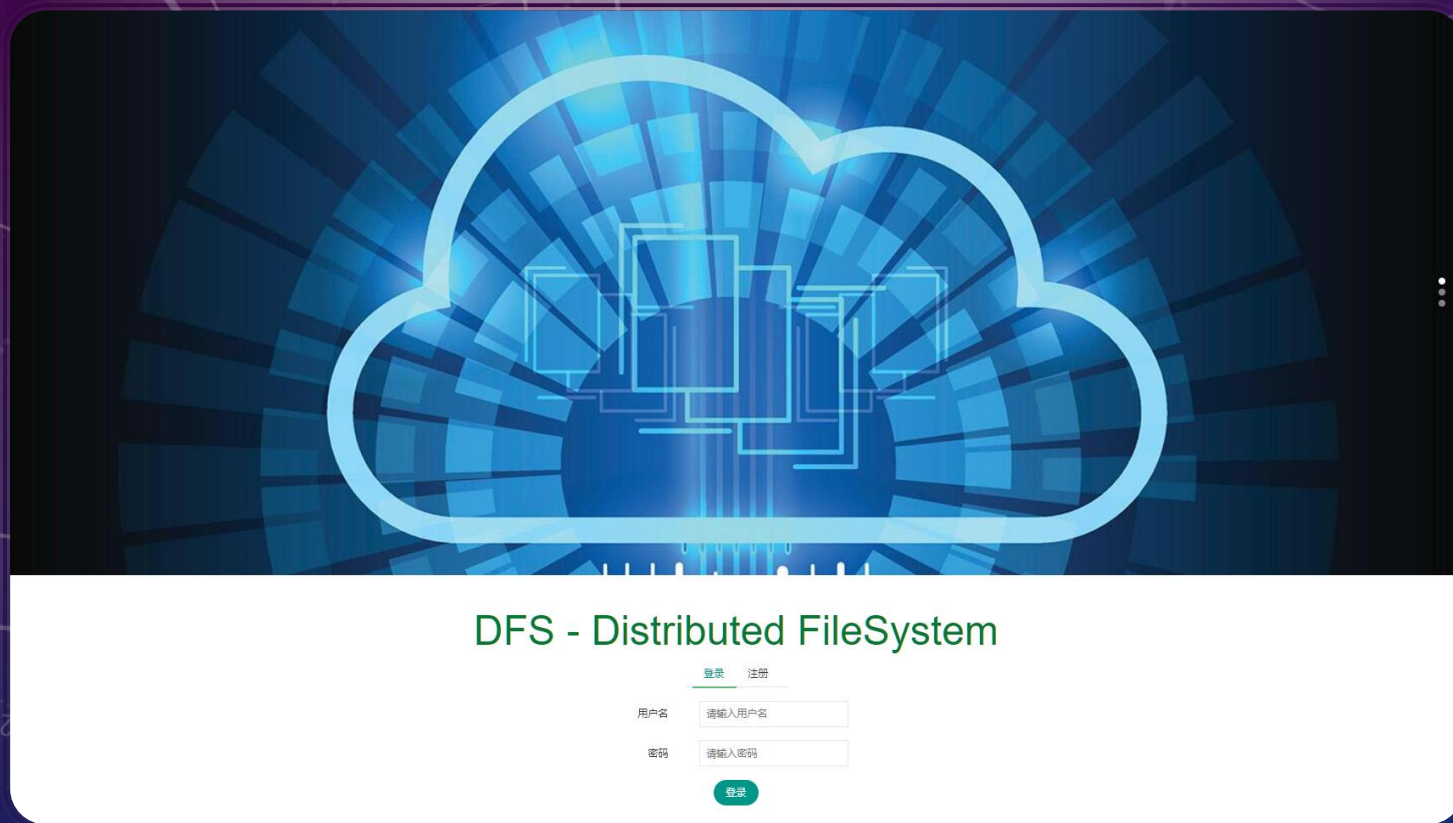
- 长度为 24 的 0/1 向量
- 覆盖上传者在线时间段  $x\%$  以上
- 两个向量做 **and** 操作，得到的向量中 1 的个数占上传者向量中 1 的个数的  $x\%$
- $X=70, n=4, m=4$ , 成功率=94.2%

# 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



The background is a solid dark blue color. It is decorated with several faint, light blue circular patterns. These include concentric circles, arcs, and dashed lines. Some of these patterns have small arrows indicating a direction of rotation. Additionally, there are degree markings (numbers) along some of the arcs, ranging from 140 to 260 in increments of 10. The overall aesthetic is technical and geometric.
$$1+1+1+1=?$$

# 彩蛋：烤鸽子

制定合理惩罚措施进行监督



## 肌本挑战

A= 30个开合跳	N= 25个Burpees
B= 20个卷腹	O= 40个开合跳
C= 30个深蹲	P= 20个俯身登山
D= 15个俯卧撑	Q= 30个卷腹
E= 1分钟靠墙半蹲	R= 15个俯卧撑
F= 10个Burpees	S= 30个Burpees
G= 1分钟平板支撑	T= 15个深蹲
H= 20个深蹲	U= 1分钟平板支撑
I= 50个开合跳	V= 2分钟靠墙半蹲
J= 15个卷腹	W= 20个Burpees
K= 10个俯卧撑	X= 60个开合跳
L= 15个俯身登山	Y= 10个卷腹
M= 20个Burpees	Z= 20个俯卧撑





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



# 还有谁敢鸽?

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 连接客户端, 使用 server 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, 输出 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

# WHAT WE GAIN?

- 选题
- 设计
- 分锅



WHAT WE GAIN





## DFS - Distributed FileSystem

[登录](#) [注册](#)

用户名

密码

[登录](#)

# DEMO



Q&A

---