大数据导论
Introduction to Big Data

第2讲：大数据的分布式存储与处理
—— 以Hadoop为例

叶允明

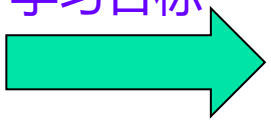
计算机科学与技术学院

哈尔滨工业大学（深圳）

大数据分布式存储与处理部分的授课安排

- Hadoop入门知识
- 深入理解HDFS/MR
- Hadoop应用案例实践

学习目标



✓ 基础概念和思想

✓ 系统架构设计能力

✓ 应用开源工具的能力

主要参考资料

- 林子雨.《大数据技术原理与应用(第2版)》. 人民邮电出版社, 2017.
- 第2、3、7、8章

（一）Hadoop入门

- 引例：Web搜索引擎
- Hadoop的基础架构
- Hadoop平台搭建示例

引例：Web搜索引擎

——理解大数据存储与处理面临的技术问题

大数据 hadoop

全部 图片 新闻 视频 地图 更多 工具

找到约 19,800,000 条结果 （用时 0.57 秒）

https://zhuanlan.zhihu.com › ...

深入浅出大数据：到底什么是Hadoop？ - 知乎专栏

2019年1月15日 — 深入浅出大数据：到底什么是Hadoop？ 2 年前· 来自专栏鲜枣课堂. 1998年9月4日，Google公司在美国硅谷成立。正如大家所知，它是一家做搜索引擎起家的 ...

https://www.zhihu.com › question

hadoop和大数据的关系？和spark的关系？ - 知乎

2015年11月25日 — Pig：是一个基于Hadoop的大规模数据分析工具，它提供的SQL-LIKE语言叫Pig Latin，该语言的编译器会把类SQL的数据分析请求转换为一系列经过优化处理的MapReduc...

34 个回答 · 最佳答案: 1998年9月4日，Google公司在美国硅谷成立。正如大家所知，它是一家...

Hadoop到底是干什么用的?	16 个回答	2019年7月9日
大数据方向除了Hadoop还有什么可学的?	18 个回答	2015年12月10日
为什么很多公司的大数 ...	57 个回答	2015年9月22日
请问大数据中Hadoop的核心技术是什么?	14 个回答	2019年10月30日

www.zhihu.com站内的其它相关信息

https://www.huaweicloud.com › articles

大数据代表技术：Hadoop、Spark、Flink、Beam - 华为云

2021年2月5日 — 大数据代表技术：Hadoop、Spark、Flink、Beam Hadoop：从2005年到2015年，说到大数据都是讲hadoop。Hadoop是一整套的技术框架，不是一个单一软件， ...

搜索引擎：网络文档集合的检索器

HTML源文件

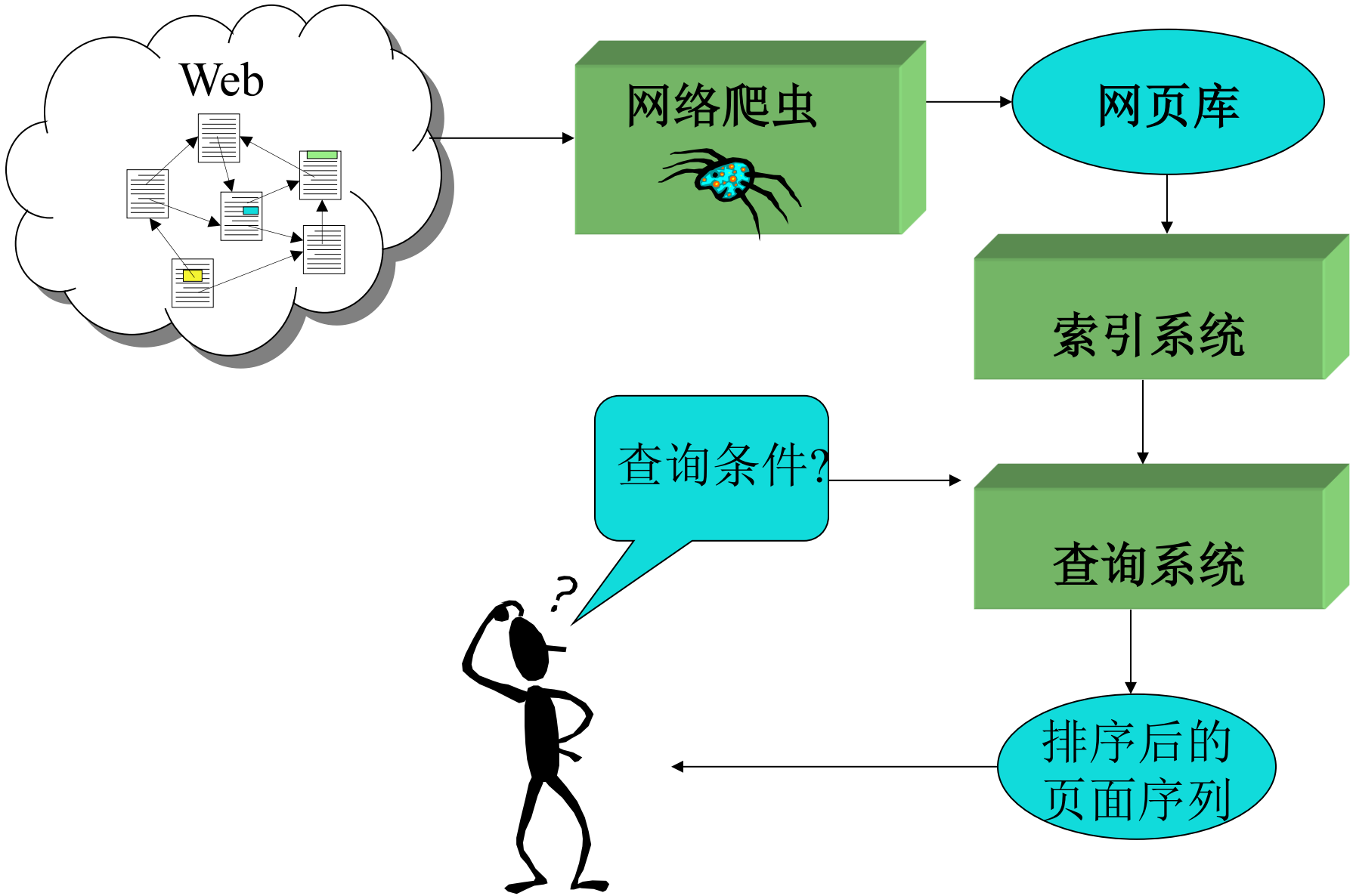
```
<nav class="navbar navbar-inverse navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed"
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <img class="navbar-logo" src="/elephant.png">
      <a class="navbar-brand" href="/"> Apache Hadoop</a>
    </div>

<div id="navbar" class="navbar-collapse collapse">
  <ul class="nav navbar-nav">
```

.....

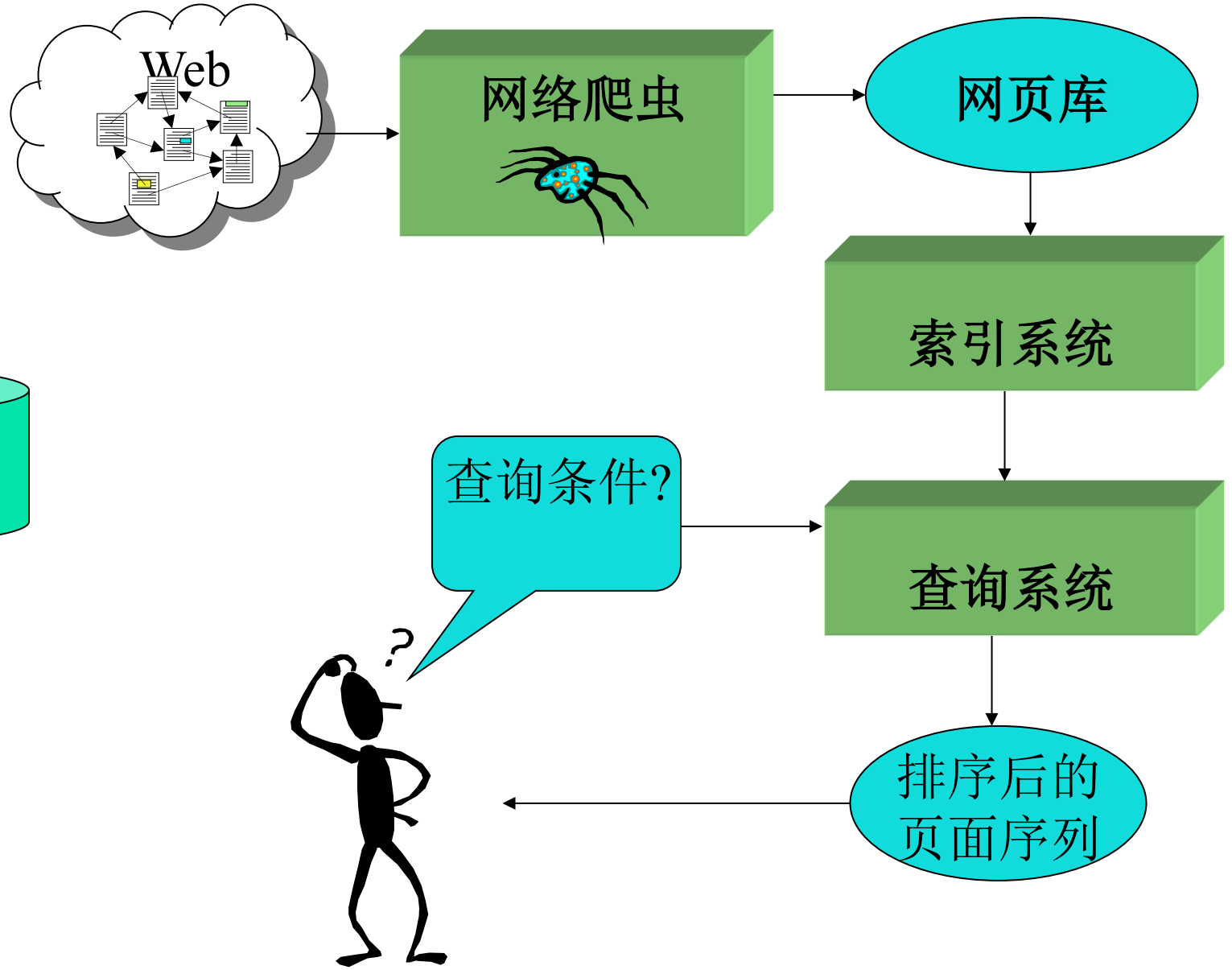

```
<!DOCTYPE html>
<html class="client-nojs" lang="zh-Hans-CN" dir="ltr"
<head>
<meta charset="UTF-8"/>
<title>Apache Hadoop - 维基百科，自由的百科全书</titl
<script>document.documentElement.className="client-js
"wgPageContentModel":"wikitext","wgRelevantPageName":
"user":"ready","user.options":"loading","ext.cite.sty
"ext.visualEditor.desktopArticleTarget.init","ext.vis
<script>(RLQ=window.RLQ||[]).push(function(){mw.loade
});});</script>
<link rel="stylesheet" href="/w/load.php?lang=zh-cn&a
<script async="" src="/w/load.php?lang=zh-cn&amp;modu
<meta name="ResourceLoaderDynamicStyles" content=""/>
<link rel="stylesheet" href="/w/load.php?lang=zh-cn&a
<link rel="stylesheet" href="/w/load.php?lang=zh-cn&a
<meta name="generator" content="MediaWiki 1.38.0-wmf.
```

Web搜索引擎的基本原理



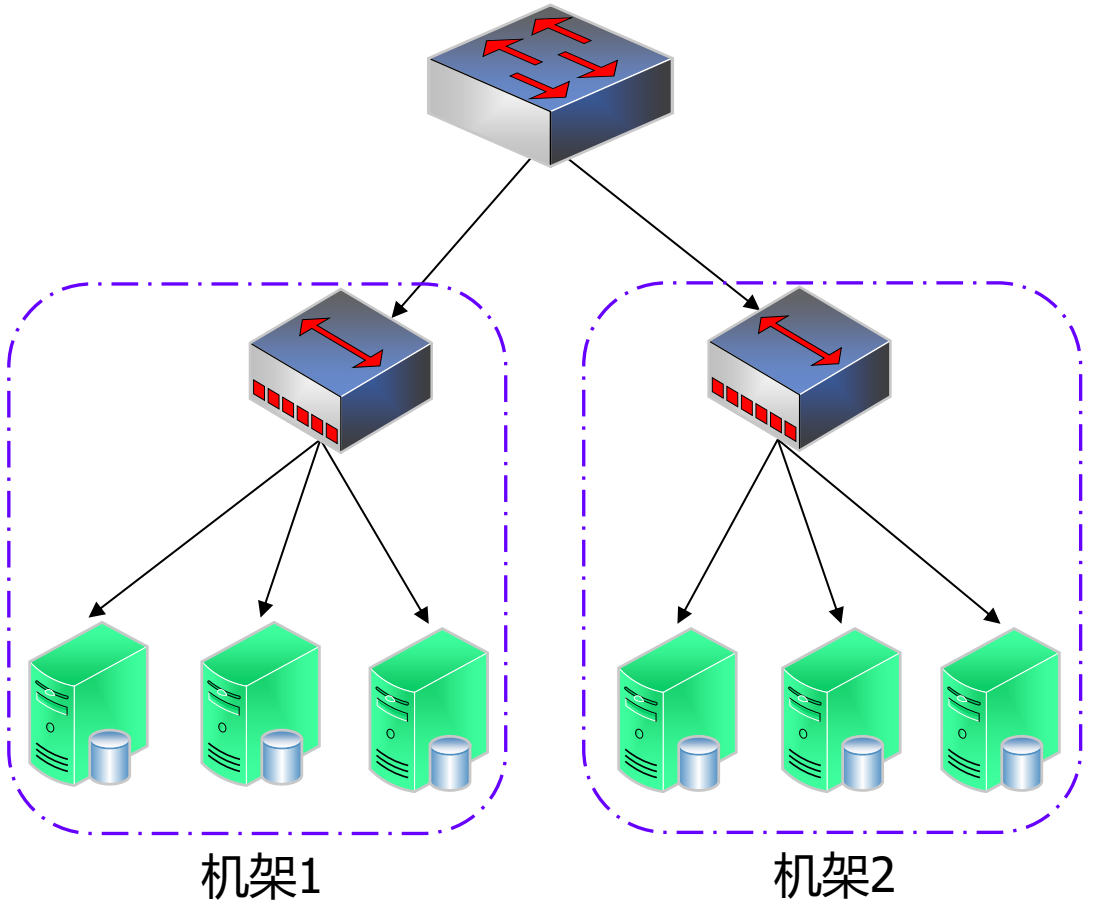
Web搜索引擎的大数据存储与处理问题

- 1万亿个网页
  - 问题1：怎么快速存储？



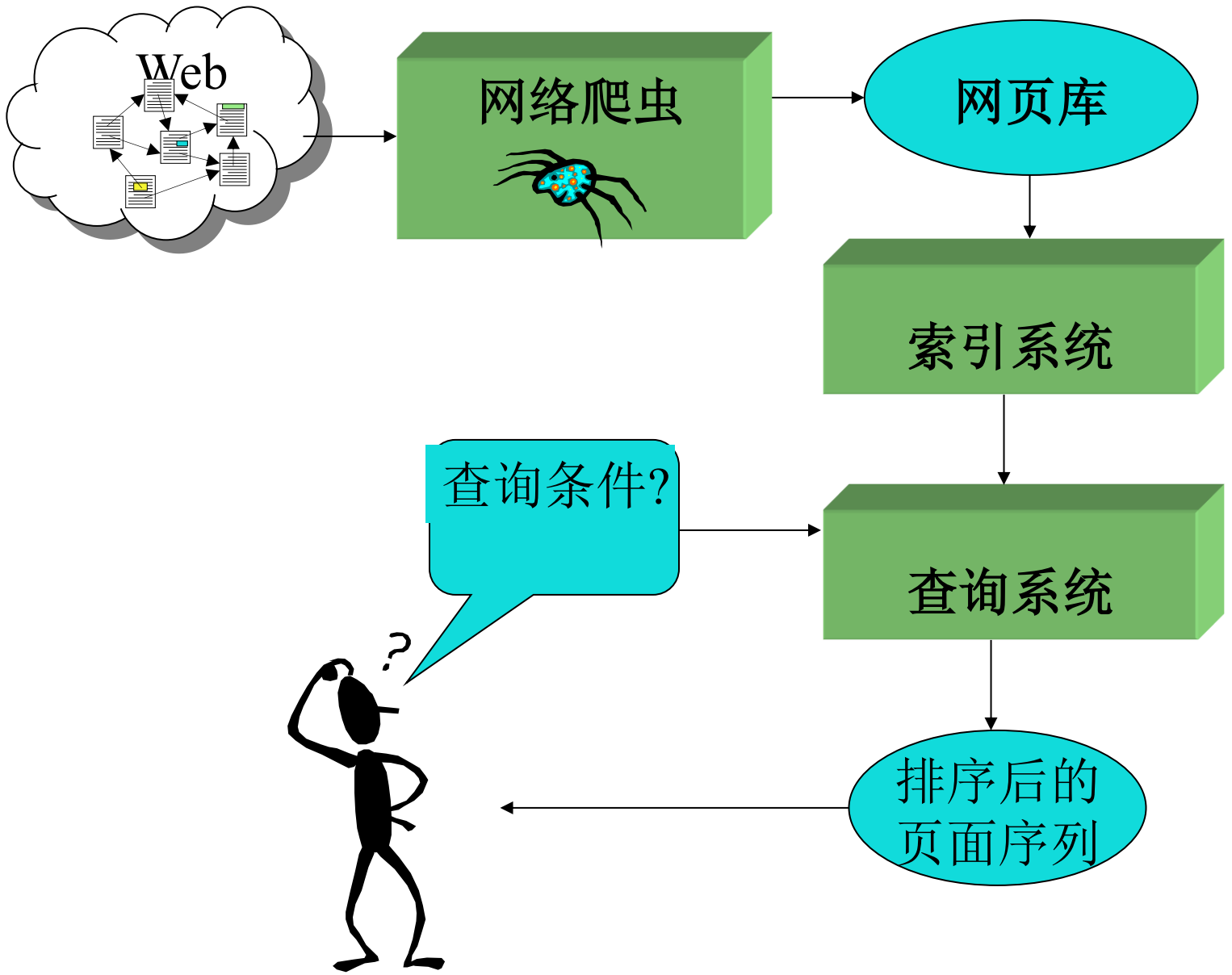
大数据分布式存储的主要技术问题分析

- 数据怎么分布?
- 存取性能如何?
  - 包括并发读、写
- 硬件故障怎么办?
  - 磁盘故障、其它硬件和网络故障
- 系统的访问接口是否简单易用?
- 硬件性能需求及成本如何?



Web搜索引擎的大数据挑战

- 1万亿个网页
  - 问题1：怎么快速存储?
  - 问题2：怎么快速检索?



问题2：怎么快速检索?

```
<nav class="navbar navbar-inverse navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed"
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <img class="navbar-logo" src="/elephant.png">
      <a class="navbar-brand" href="/"> Apache Hadoop</a>
    </div>

<div id="navbar" class="navbar-collapse collapse">
  <ul class="nav navbar-nav">
```

网页1的源代码文档

```
<!DOCTYPE html>
<html class="client-nojs" lang="zh-Hans-CN" dir="ltr"
<head>
<meta charset="UTF-8"/>
<title>Apache Hadoop - 维基百科，自由的百科全书</titl
<script>document.documentElement.className="client-js
"wgPageContentModel":"wikitext","wgRelevantPageName":
"user":"ready","user.options":"loading","ext.cite.sty
"ext.visualEditor.desktopArticleTarget.init","ext.vis
<script>(RLQ=window.RLQ||[]).push(function(){mw.loade
});});</script>
<link rel="stylesheet" href="/w/load.php?lang=zh-cn&a
<script async="" src="/w/load.php?lang=zh-cn&amp;modu
<meta name="ResourceLoaderDynamicStyles" content=""/>
<link rel="stylesheet" href="/w/load.php?lang=zh-cn&a
<link rel="stylesheet" href="/w/load.php?lang=zh-cn&a
<meta name="generator" content="MediaWiki 1.38.0-wmf.
```

网页2的源代码文档

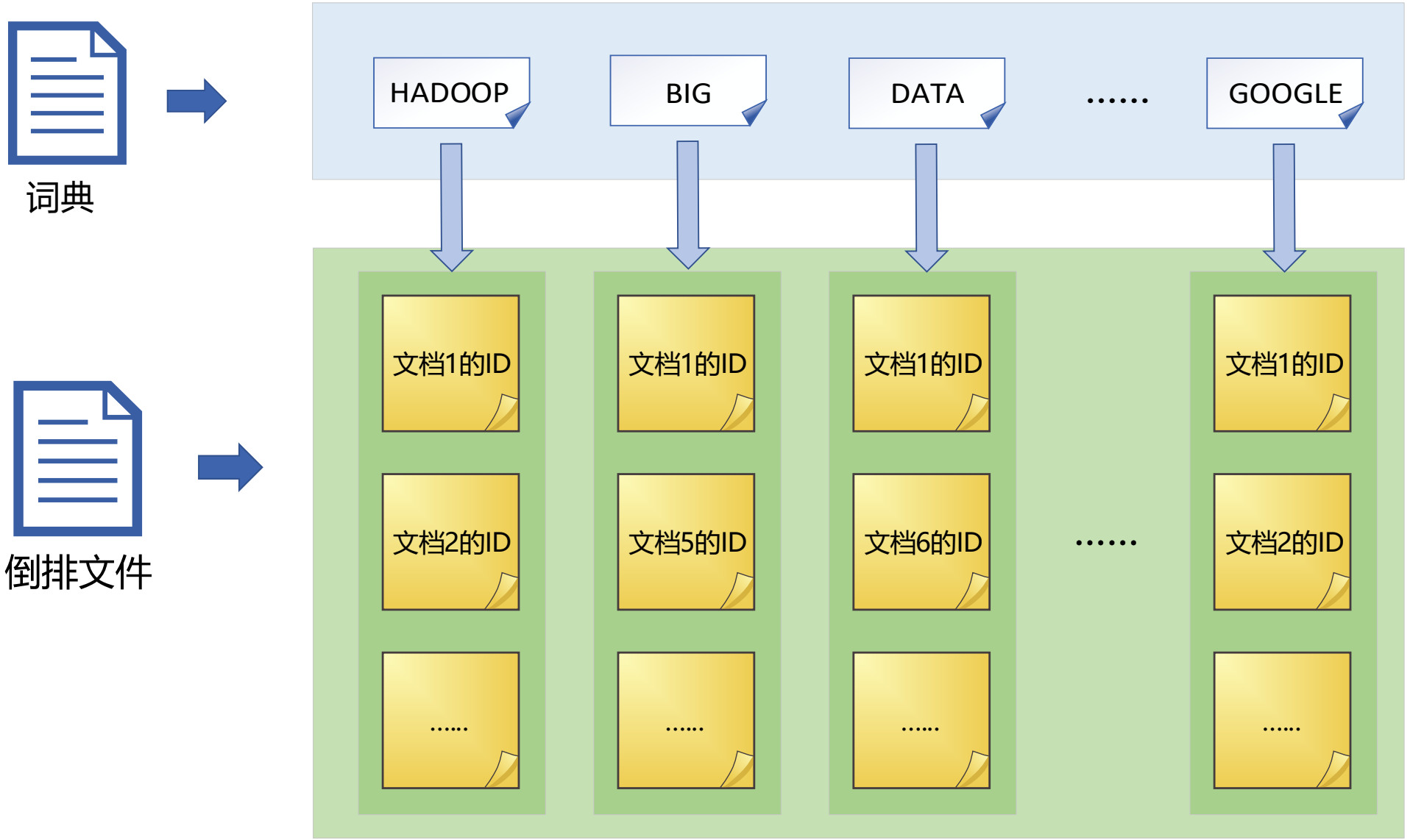
快速检索模型：基于倒排索引（Inverted Index）

文档1：
Hadoop is open-source.

文档2：
HDFS is a distributed file system.

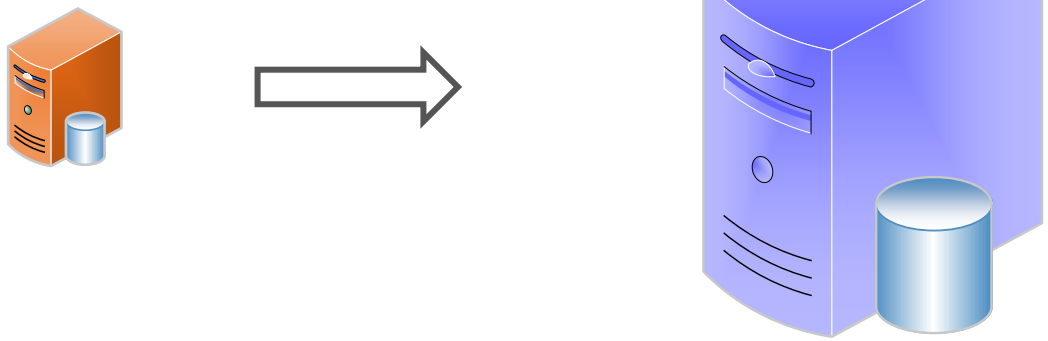
文档3：
A Hadoop system can deliver high availability.

倒排索引（Inverted Index）

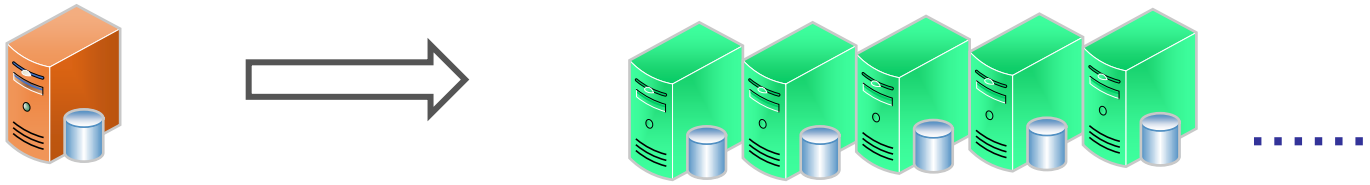


怎么快速构建倒排索引?

"垂直扩展"

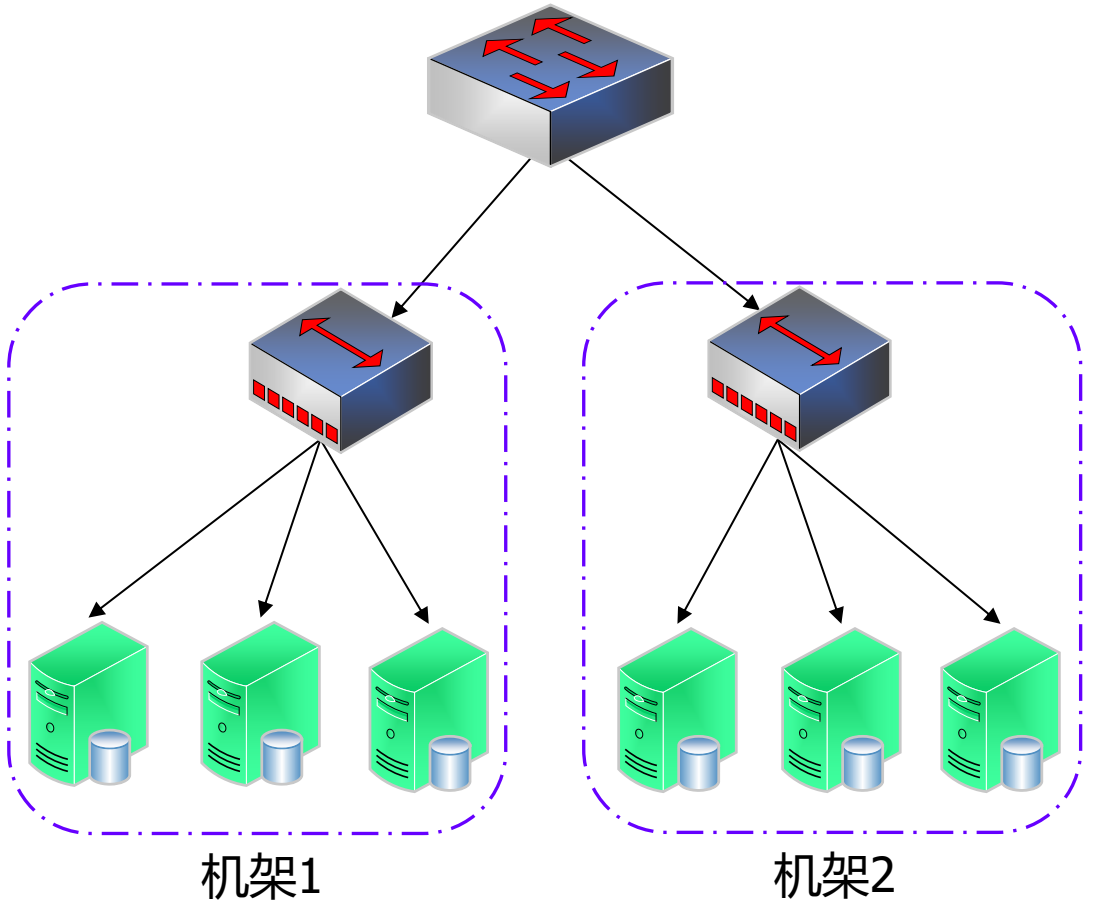


"水平扩展": 分布式处理!



大数据分布式处理的主要技术问题分析

- 计算任务（负载）如何分配?
- 分布式通信带来的额外开销问题?
  - 数据分发、处理结果收集
  - 大数据分布处理可带来的数据传输问题
- 硬件故障怎么办?
  - 节点故障、网络故障
- 系统的访问接口是否简单易用?
- 硬件性能需求及成本如何?



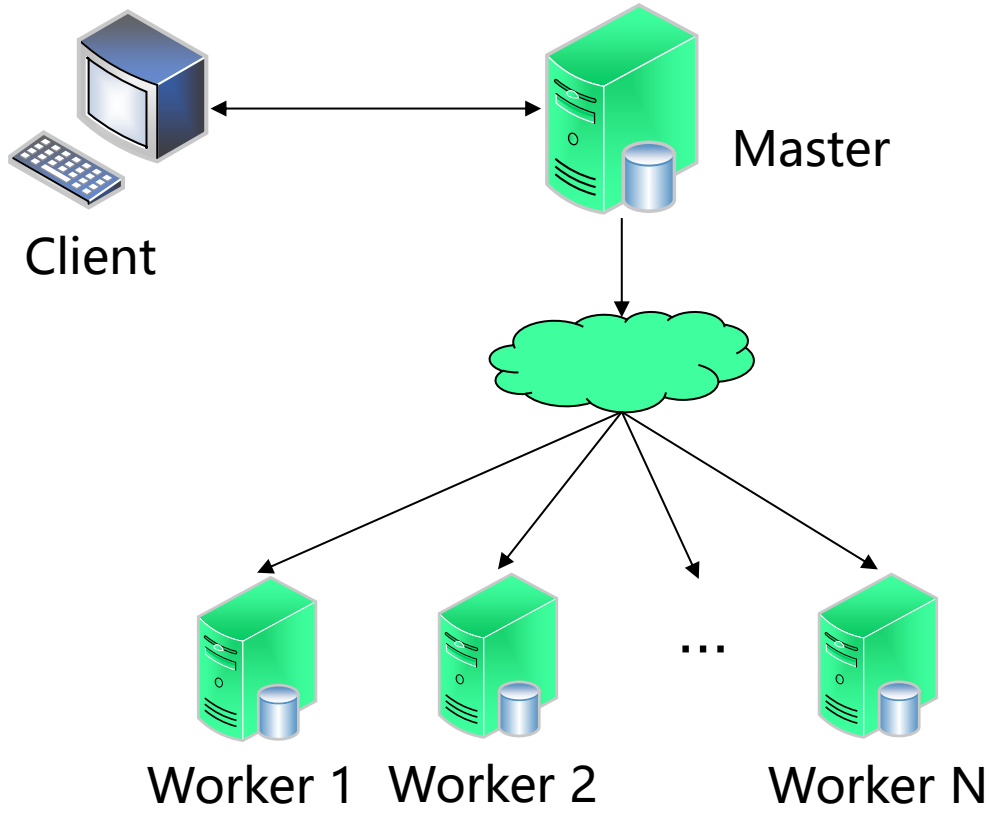
Hadoop基础架构概览

Hadoop的诞生

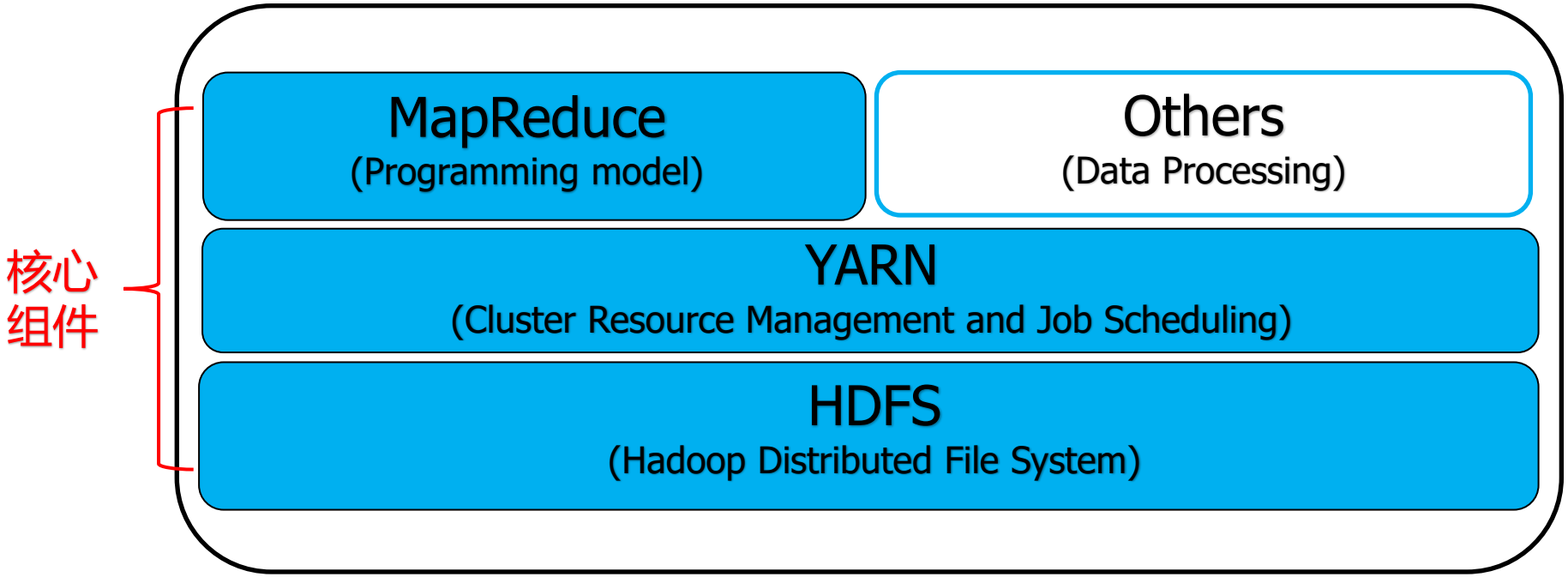
- 产生背景：为解决互联网时代的大数据存储与计算架构问题
  - 硬件故障问题、存储成本问题、快速计算问题……
- Hadoop：实现高效数据存储、处理的一种分布式框架
  - 谷歌的GFS和MapReduce的开源实现版本
- Hadoop可以解决PB级别的数据存储与计算问题
- Hadoop基于Java语言开发：具有很好的跨平台性
  - Hadoop 上的应用程序也可用其它语言编写，如C/C++

Hadoop分布式框架的基本思想

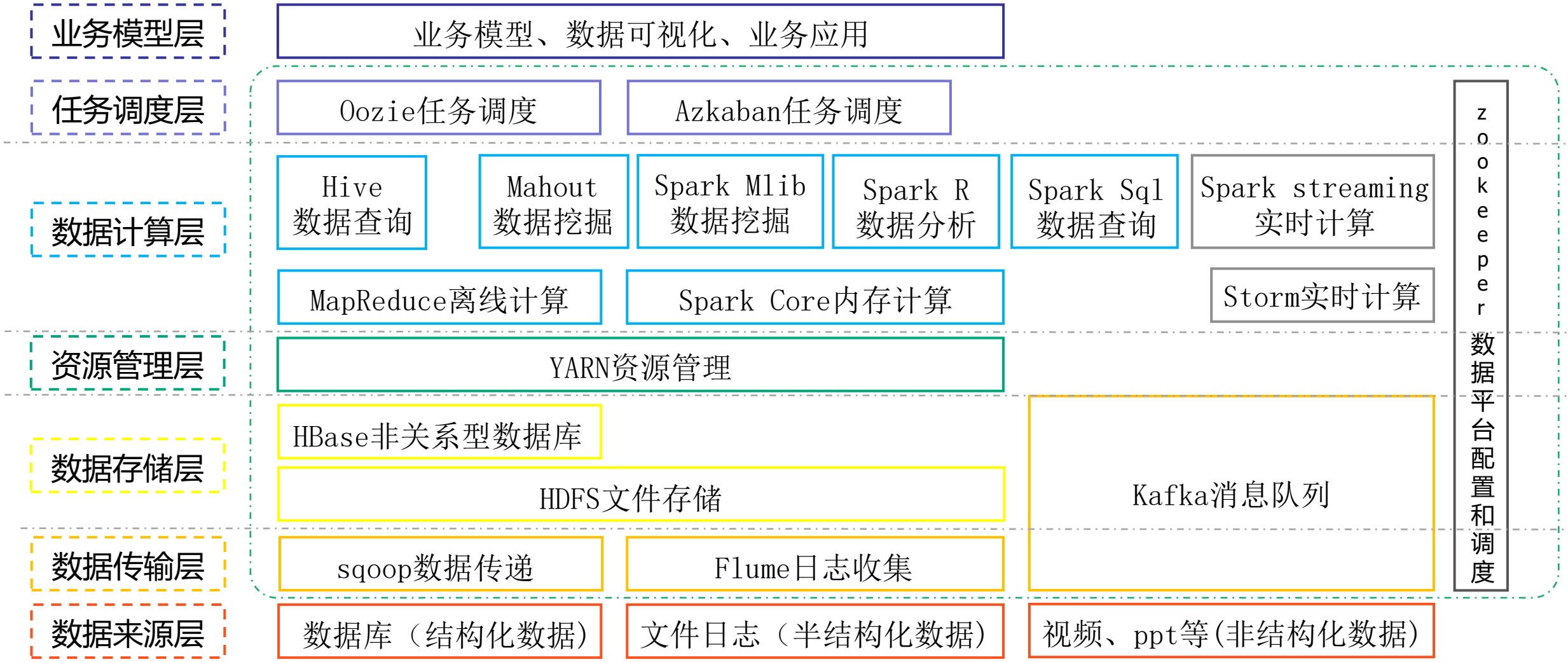
- Master-slave架构
- 分布式存储：HDFS
- 分布式计算：Mapreduce
- 存储与处理的一体化!



Hadoop系统的核心组件



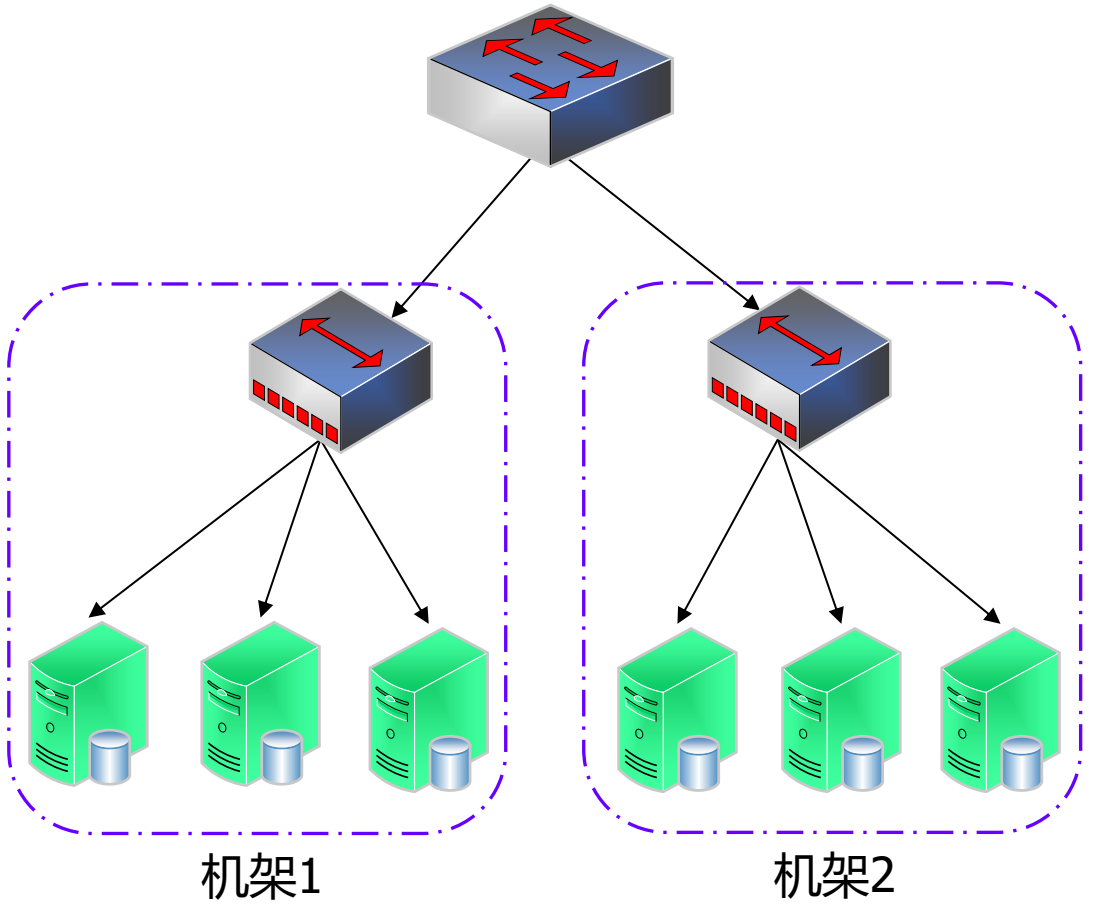
Hadoop生态


业务模型层
业务模型、数据可视化、业务应用
任务调度层
Oozie任务调度
Azkaban任务调度
数据计算层
Hive
数据查询
Mahout
数据挖掘
Spark Mlib
数据挖掘
Spark R
数据分析
Spark Sql
数据查询
Spark streaming
实时计算
MapReduce离线计算
Spark Core内存计算
Storm实时计算
资源管理层
YARN资源管理
数据存储层
HBase非关系型数据库
HDFS文件存储
Kafka消息队列
数据传输层
sqoop数据传递
Flume日志收集
数据来源层
数据库（结构化数据）
文件日志（半结构化数据）
视频、ppt等(非结构化数据)
zookeeper
数据平台配置和调度

HDFS：Hadoop Distributed File System

回顾：大数据分布式存储的主要技术问题分析

- 数据怎么分布?
- 存取性能如何?
  - 包括并发读、写
- 硬件故障怎么办?
  - 磁盘故障、其它硬件和网络故障
- 系统的访问接口是否简单易用?
- 硬件性能需求及成本如何?

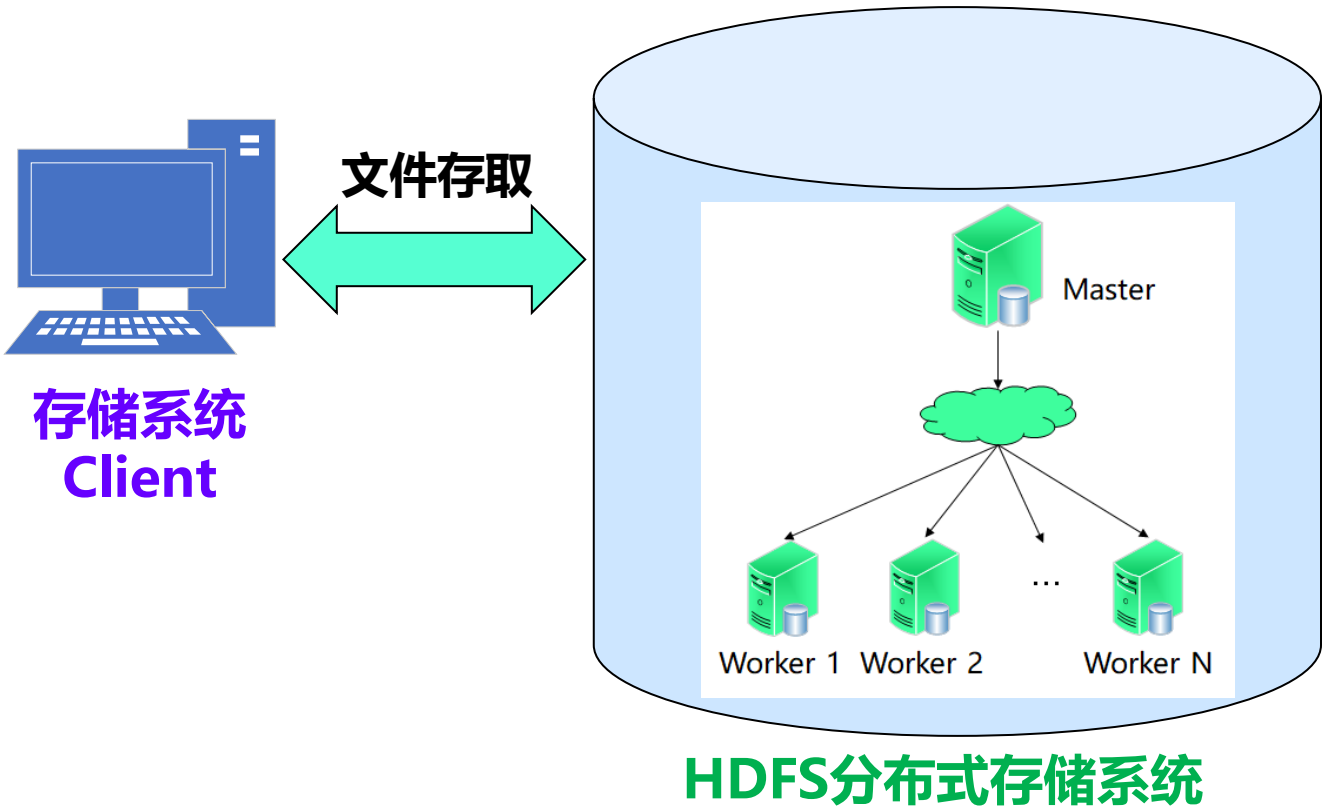


HDFS简介

- “高容错、低成本的分布式大磁盘”，设计需求：
  - 简单的文件访问模型：类似于linux的文件系统！
  - PB级数据的可靠存储
  - 流数据读写：高吞吐率
  - 支持上万台服务器集群
  - 对硬件设备性能要求低
  - 集群规模具有可扩展性

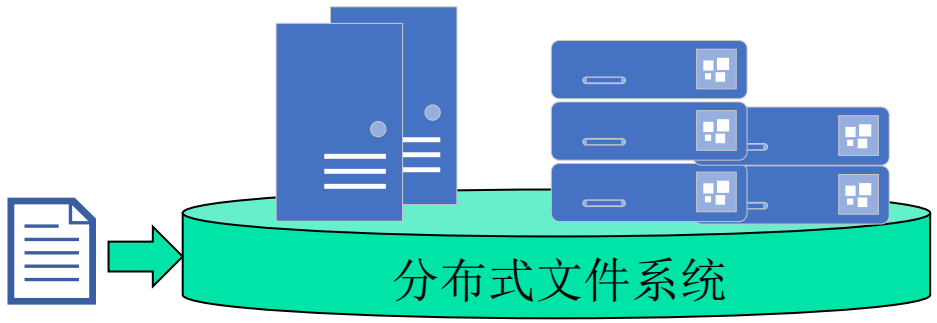
√

  - 兼容性好，支持跨平台 √



基于分布式文件系统的存储方案优势

- 分布式文件系统改变了数据存储和管理方式，相对于本地文件系统具有很多优势：
  - 低成本
  - 易扩展
  - 强可靠
  - 高可用



- 用户无需关心数据是存储在哪个节点上，可以如同使用本地文件系统一样存储和管理分布式文件系统里的数据。

HDFS的文件系统

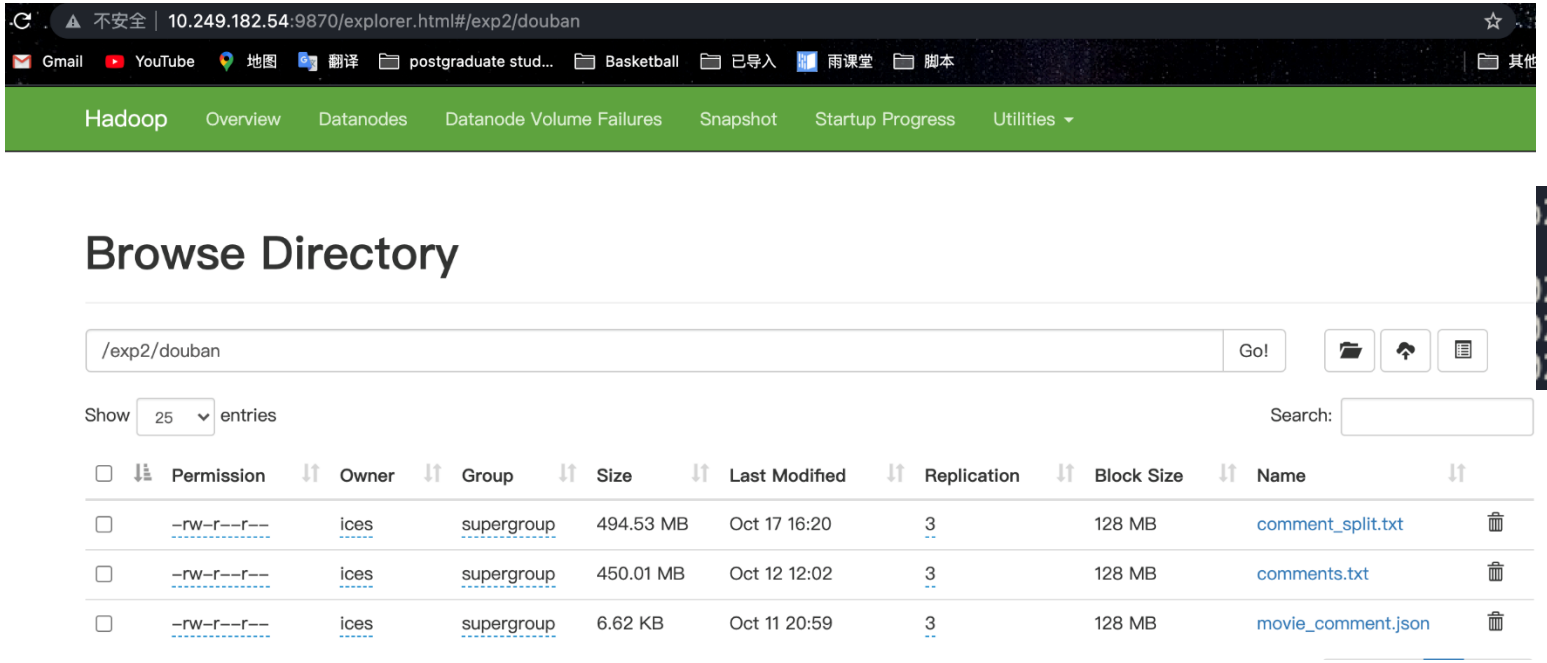
```
(base) ices@ices-master:~$ hdfs dfs -ls /exp2/douban
Found 3 items
-rw-r--r--   3 ices supergroup  518555983 2021-10-17 16:20 /exp2/douban/comment_split.txt
-rw-r--r--   3 ices supergroup  471869510 2021-10-12 12:02 /exp2/douban/comments.txt
-rw-r--r--   3 ices supergroup       6783 2021-10-11 20:59 /exp2/douban/movie_comment.json
```

- **文件URL定位：hdfs://10.28.36.101:8020/exp2/douban/commens.txt**

- 简单的文件访问模型：类似于linux的文件系统！ √

HDFS的逻辑存储模型：文件系统

- 以“文件”为基本的逻辑存储单位，形成文件系统
- 分级的文件系统：其命名空间包含目录、文件
  - 用户可象使用普通文件系统一样创建、删除目录和文件，在目录间转移文件、重命名文件



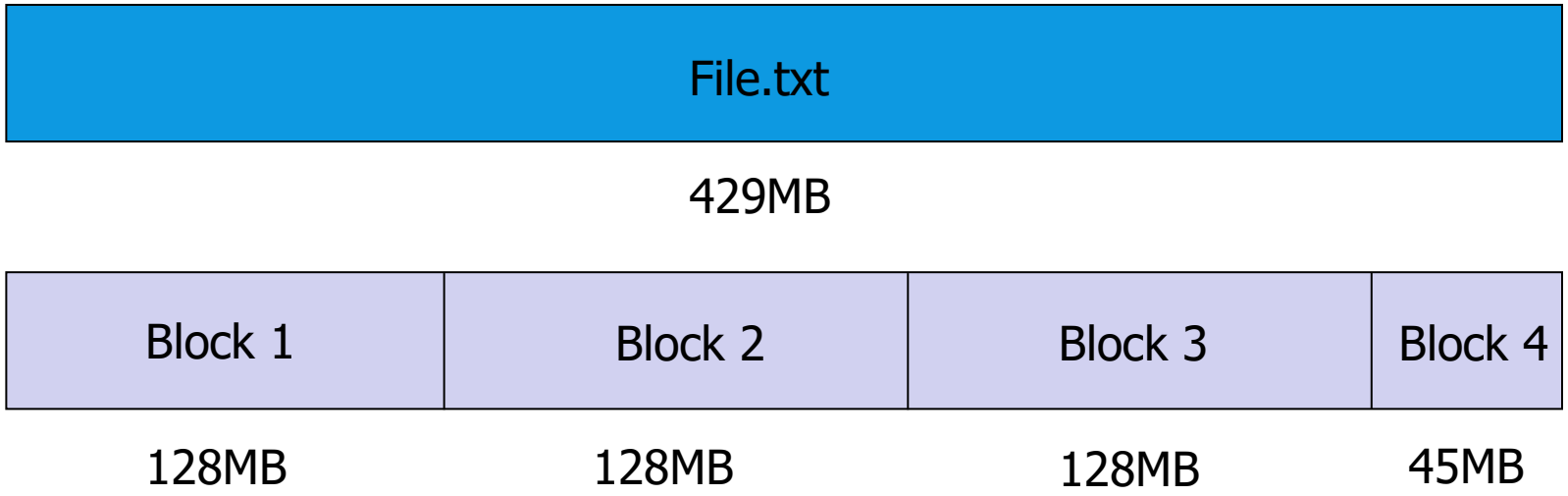

```
2/douban

21-10-17 16:20 /exp2/douban/comment_split.txt
21-10-12 12:02 /exp2/douban/comments.txt
21-10-11 20:59 /exp2/douban/movie_comment.json
```

- 整个HDFS集群中只有一个统一的命名空间（由一个名称节点管理）

HDFS的逻辑存储模型：数据块

- 每个HDFS文件以一个或多个数据块进行存储（每个块64MB/128MB）

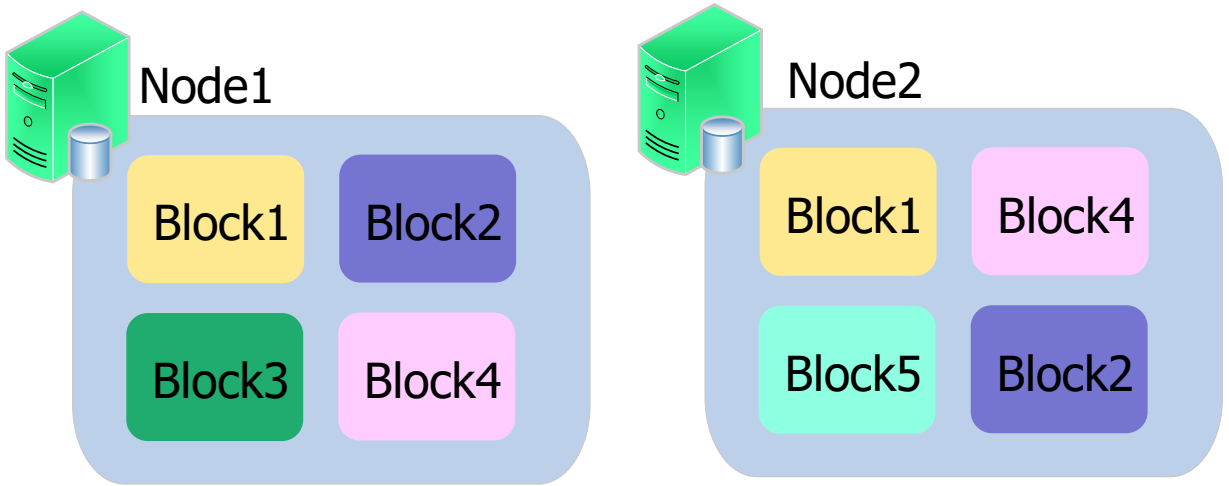


- 块的大小远远大于普通文件系统，可以最小化寻址开销

HDFS文件的“分**块**”存储思想

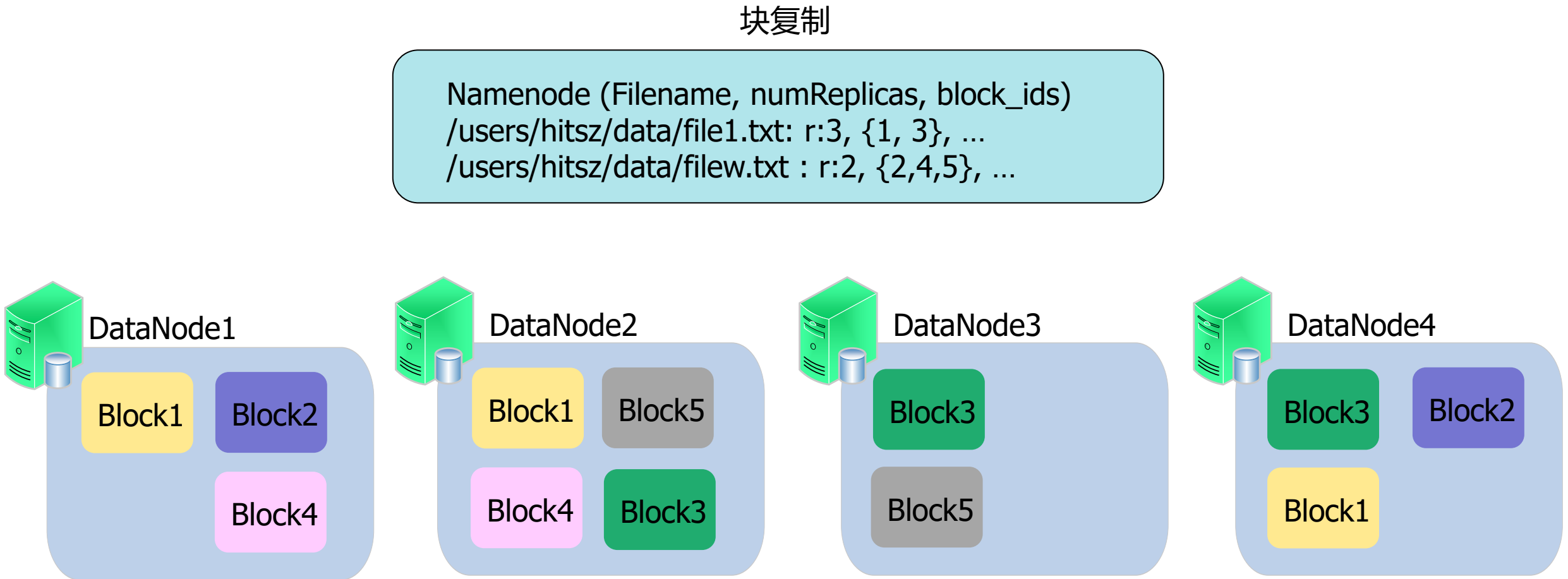
- 一个大规模文件被切分成不同的块，**每个块尽可能地存储于不同的数据节点中**
  - 块的大小远远大于普通文件系统，可以最小化寻址开销
  - 支持大规模文件存储、简化系统设计、适合数据备份

- PB级数据的可靠存储 √
- 流数据读写：高吞吐率 √



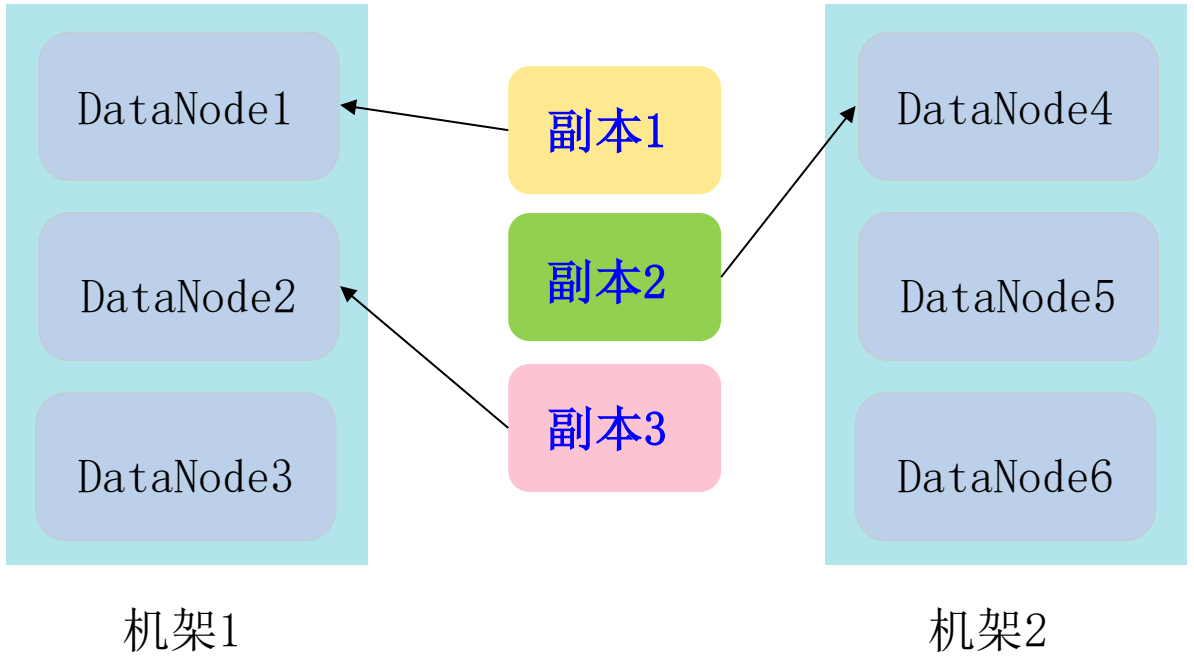
数据块的物理存储模型

- 分布式多副本冗余存储



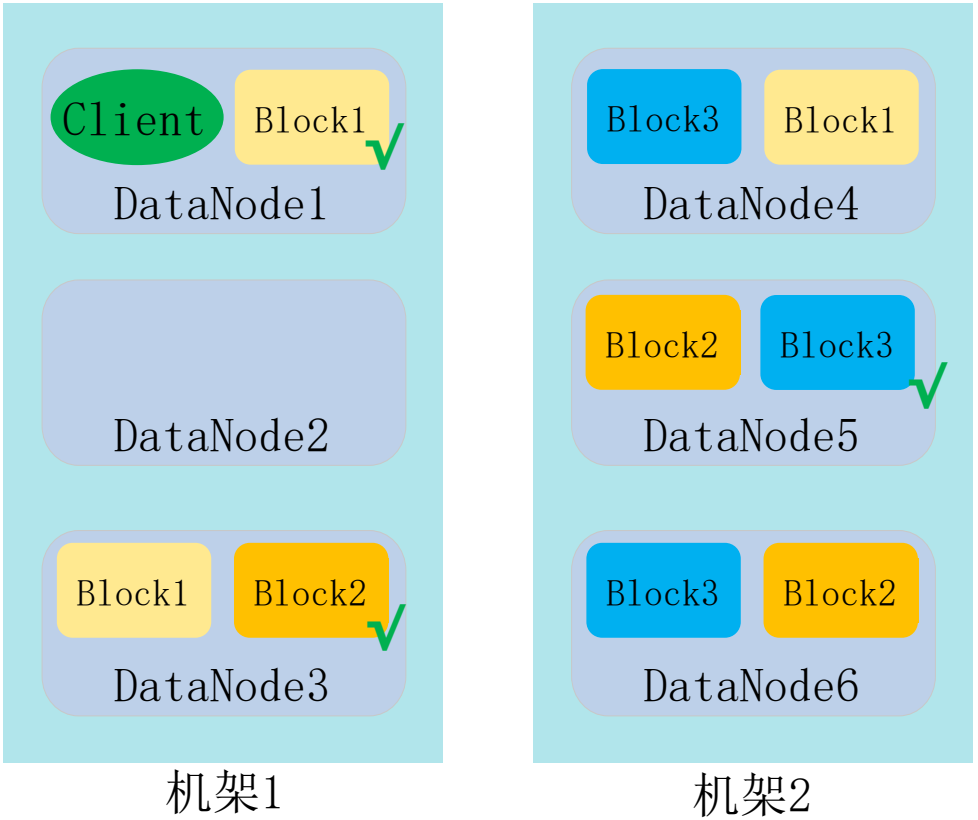
数据存放策略

- **第一个副本：**放置在上传文件的数据节点；如果是集群外提交，则随机挑选一台磁盘不太满、CPU不太忙的节点
- **第二个副本：**放置在与第一个副本不同的机架的节点上
- **第三个副本：**与第一个副本相同机架的其他节点上
- 更多副本：随机节点

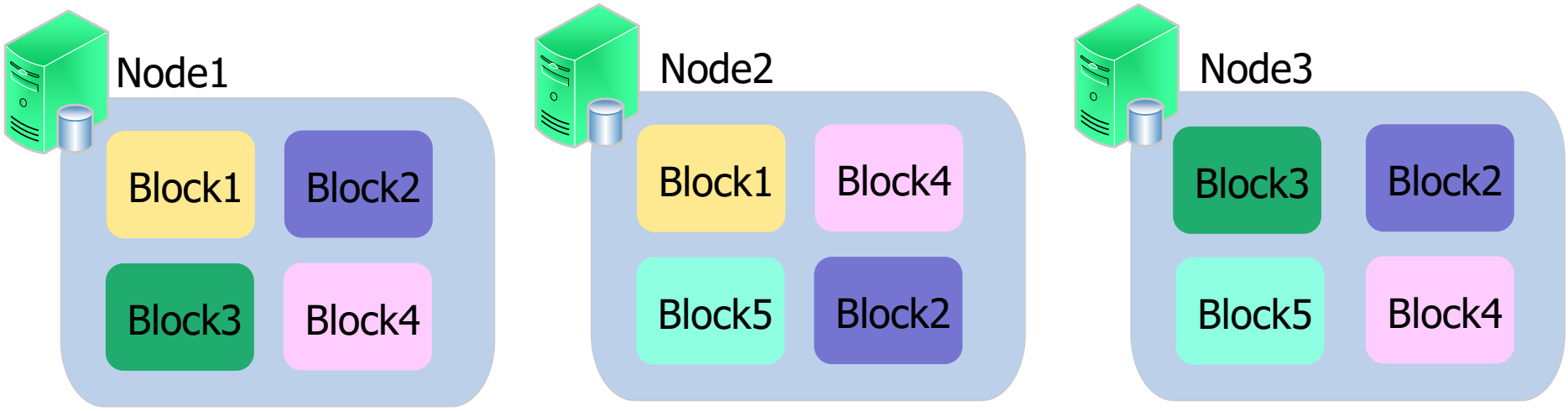


数据读取策略

- 机架ID获取：HDFS提供了一个API可以确定一个数据节点所属的机架ID，客户端也可以调用API获取自己所属的机架ID
- 副本读取策略：（1）优先选择同一节点；（2）次选同一机架；（3）随机读取

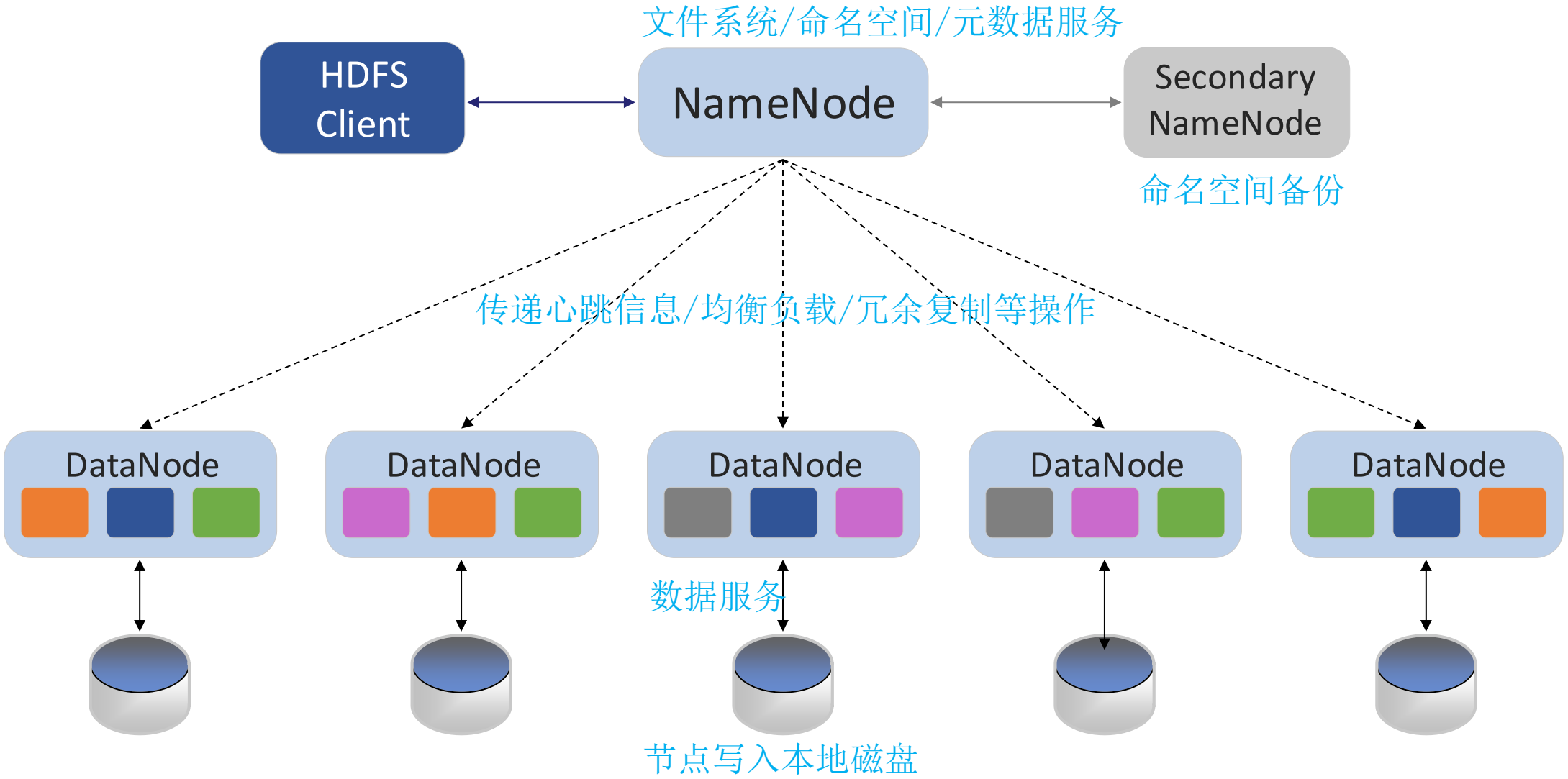


分布式文件块存储的优点



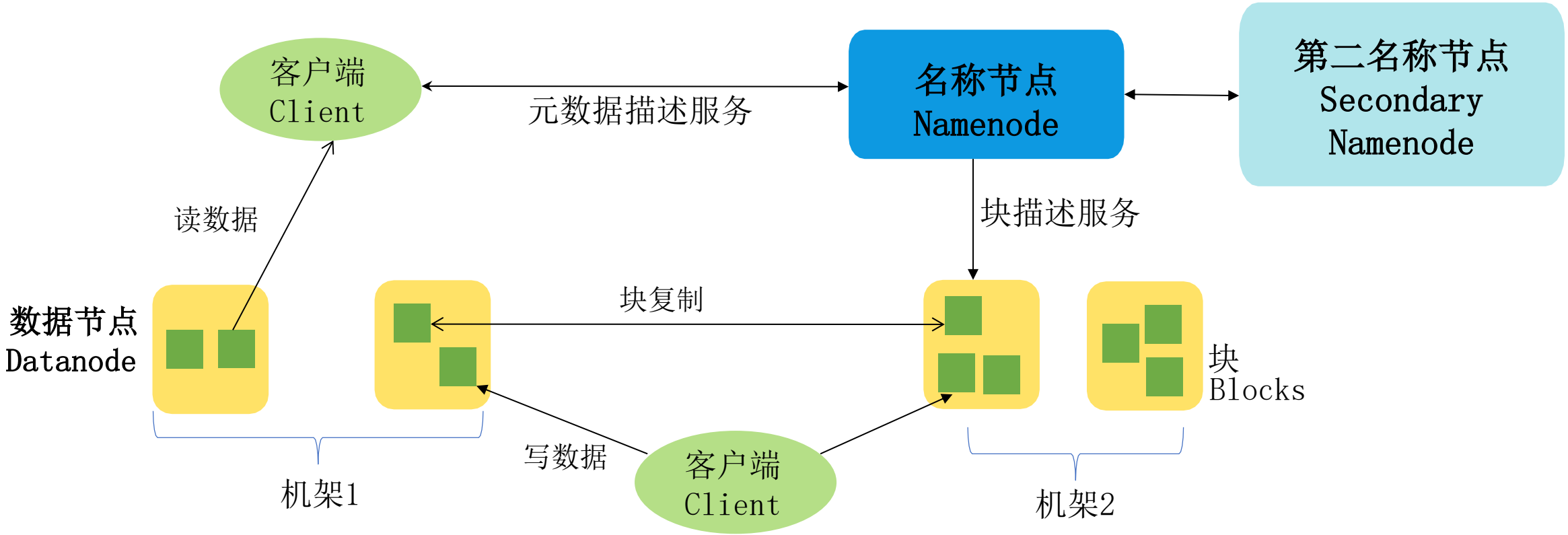
- 支持大文件存储
- 保证数据可靠性
- 加快数据存取速度：
  - ✓ 单个文件的分布式块存取
  - ✓ 多个文件分布式并行读取
- 简化了存储系统设计

HDFS的分布式架构


文件系统/命名空间/元数据服务
HDFS Client
NameNode
Secondary NameNode
命名空间备份
传递心跳信息/均衡负载/冗余复制等操作
DataNode
DataNode
DataNode
DataNode
DataNode
数据服务
节点写入本地磁盘

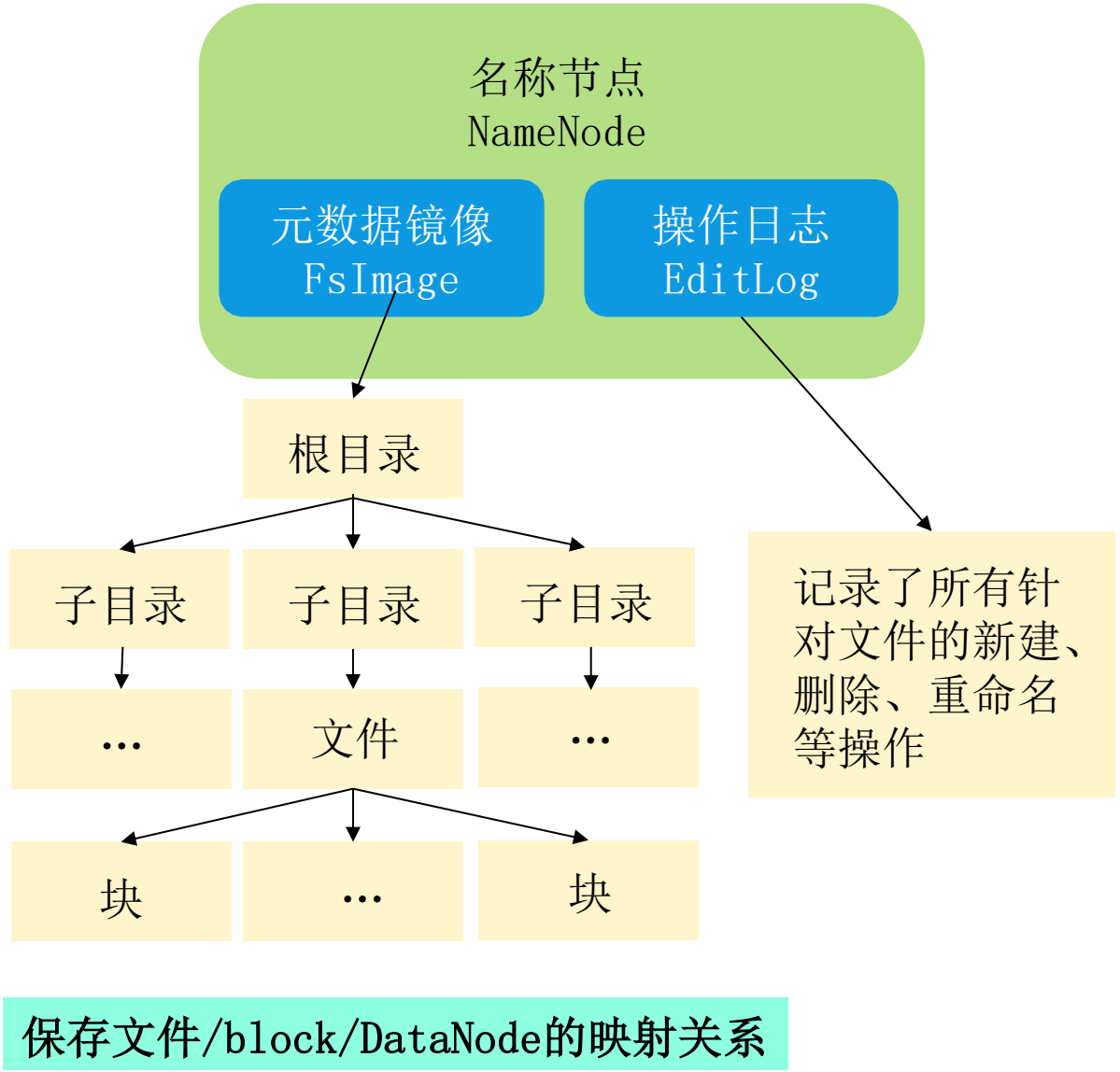
HDFS的基本组成

- HDFS采用了主从（Master/Slave）体系结构，易于实现和管理



名称节点（NameNode）

- 名称节点是分布式文件系统中的管理者
- 命名空间和元数据管理
- 核心数据结构FsImage和EditLog
  - ✓ FsImage：维护文件系统树及元数据
  - ✓ EditLog：操作日志文件记录



第二名称节点（Secondary NameNode）

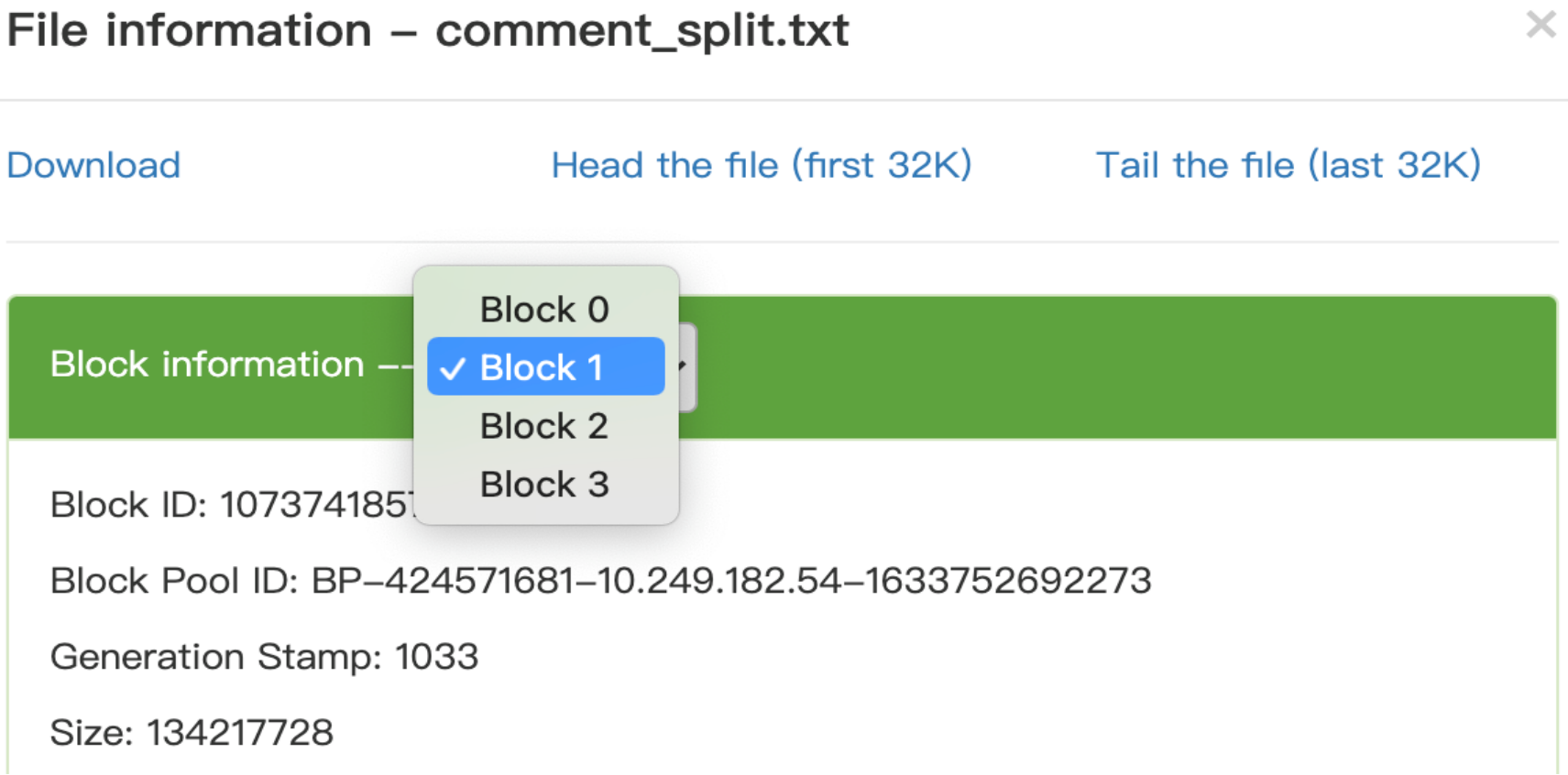
- 用来保存名称节点中HDFS元数据的备份，并减少名称节点重启时间
- 一般是单独运行在一台机器上。

数据节点（Data Node）

- 数据节点负责数据的存储和读取
- “主动汇报”：定期向名称节点发送自己所存储的块的列表。
- 数据节点中的数据块会被保存在各自节点的本地Linux文件系统中

数据存储

数据读取



节点信息

据
ta

HDFS客户端

- HDFS客户端是用户操作HDFS最常用的方式，HDFS在部署时都提供了客户端
- HDFS客户端是一个库，暴露了HDFS文件系统接口，这些接口隐藏了HDFS实现中的大部分复杂性
- 客户端提供一个类似于POSIX（可移植操作系统界面）的文件系统接口，因此用户在编程时无需知道Namenode和Datanode也可实现其功能。

HDFS读写接口（Java）

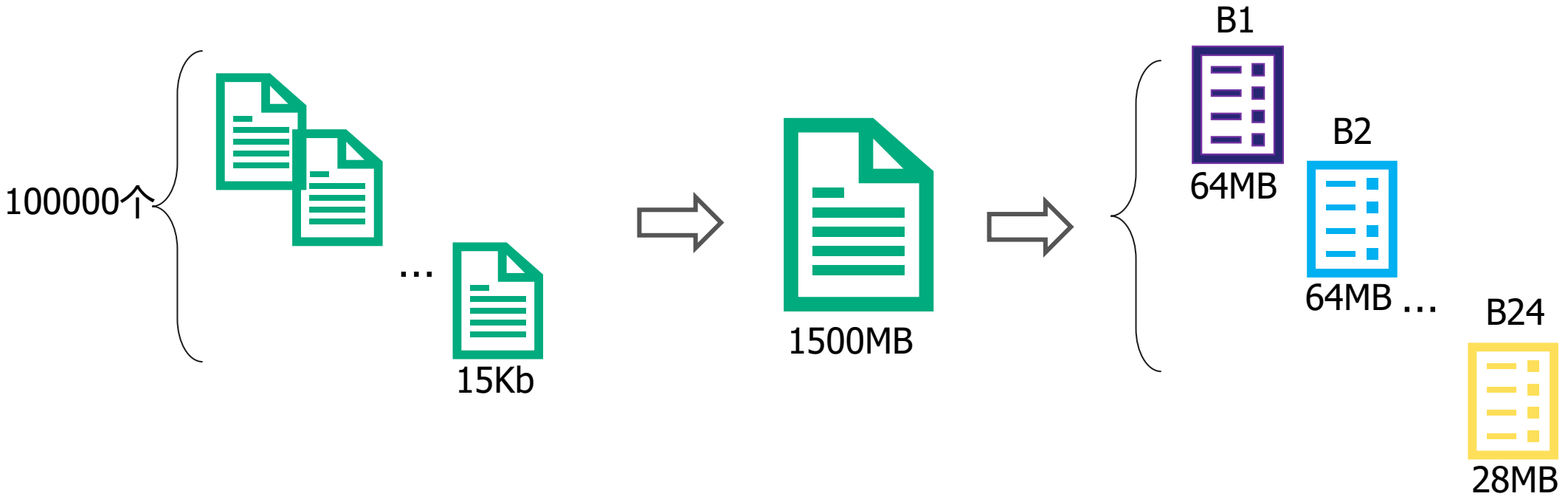
- FileSystem是一个通用文件系统的抽象基类，可以被分布式文件系统继承，所有可能使用Hadoop文件系统的代码，都要使用这个类
- Hadoop为FileSystem这个抽象类提供了多种具体实现，DistributedFileSystem就是FileSystem在HDFS文件系统中的具体实现
- FileSystem的open()方法返回的是一个输入流FSDataInputStream对象，在HDFS文件系统中，具体的输入流就是DFSInputStream；FileSystem中的create()方法返回的是一个输出流FSDataOutputStream对象，在HDFS文件系统中，具体的输出流就是DFSOutputStream。

```
Configuration conf = new Configuration();
FileSystem fs = FileSystem.get(conf);
FSDataInputStream in = fs.open(new Path(uri));
FSDataOutputStream out = fs.create(new Path(uri));
```

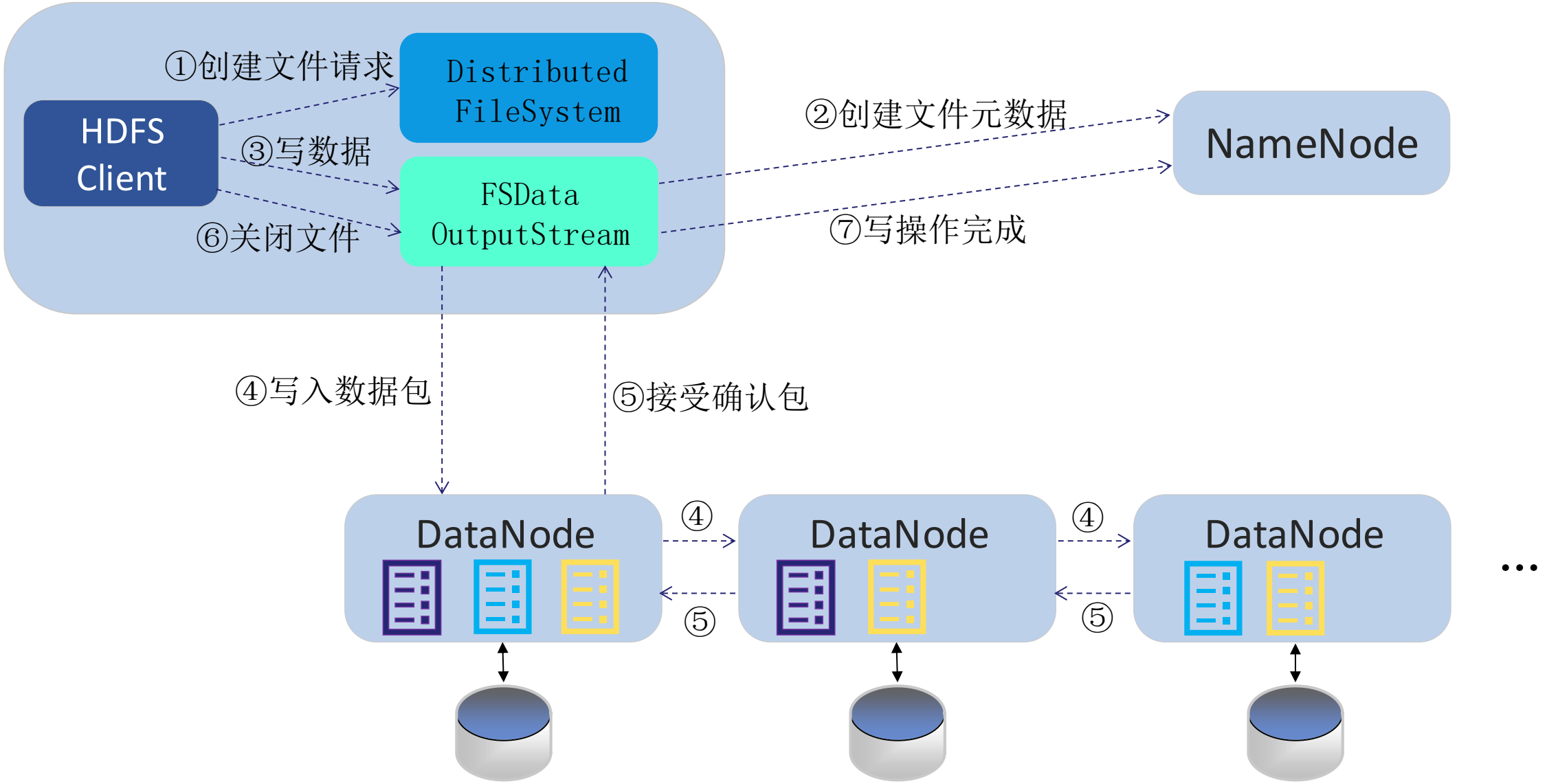
HDFS的读写流程与容错恢复机制

网页存储应用示例（问题）

- 用户需求：将10万个网页存储到HDFS系统中：



网页存储应用示例（流程）

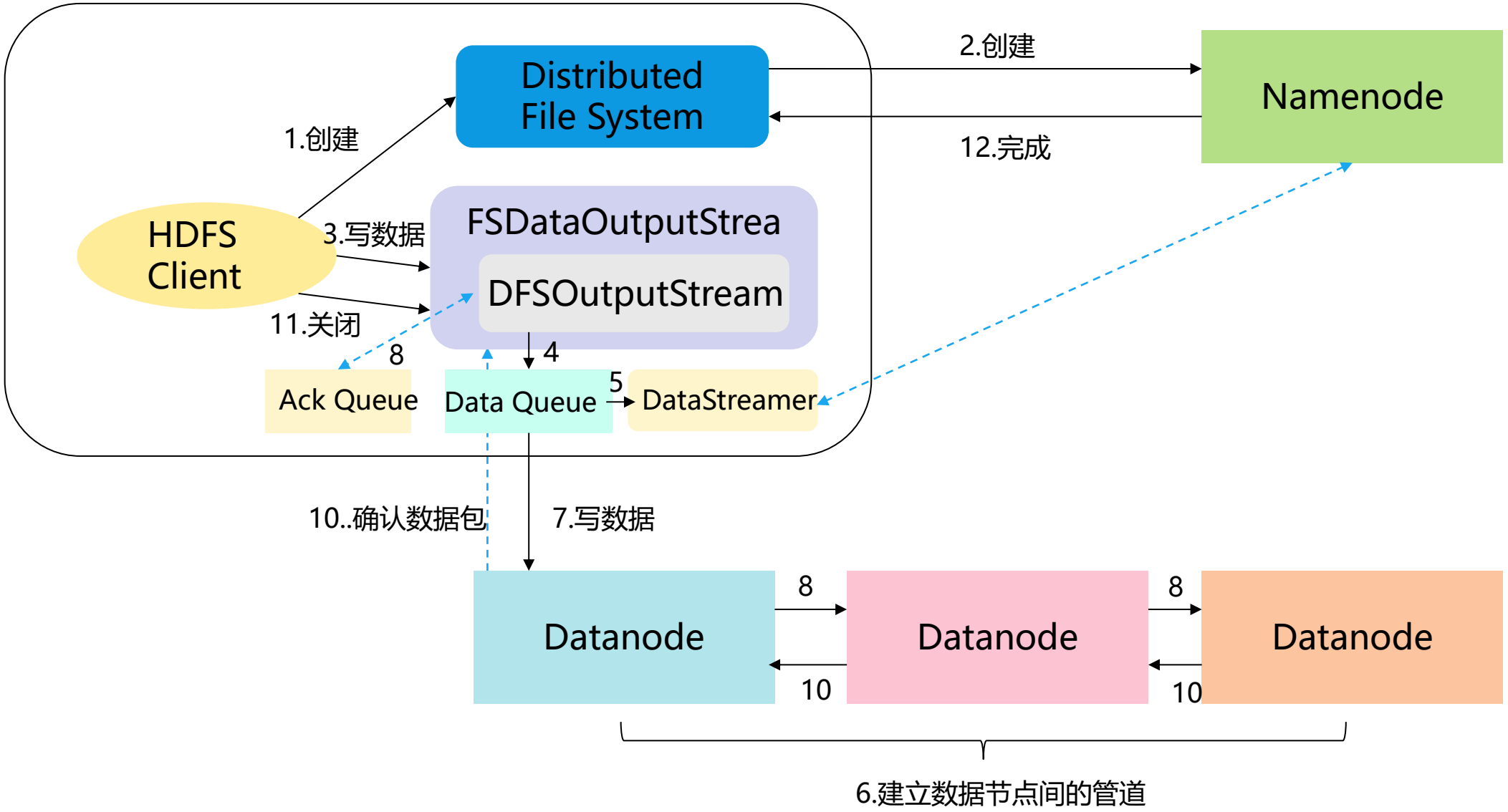

①创建文件请求
Distributed FileSystem
HDFS Client
③写数据
FSData OutputStream
⑥关闭文件
②创建文件元数据
NameNode
⑦写操作完成
④写入数据包
⑤接受确认包
DataNode
④
⑤
DataNode
④
⑤
DataNode
...

HDFS数据读写过程——写数据请求

- 首先，client请求NameNode，要将文件（可能需分为多个块）写入到HDFS。
- NameNode会给client赋予写权限，并为client提供可以写入数据的DataNode的IP地址。NameNode在选择可写入数据的DN的规则是：结合了DN的健康状态、复制因子、机架感知等因素
- 假如复制因子是3(默认值)，那么会为每个block返回三个IP地址。

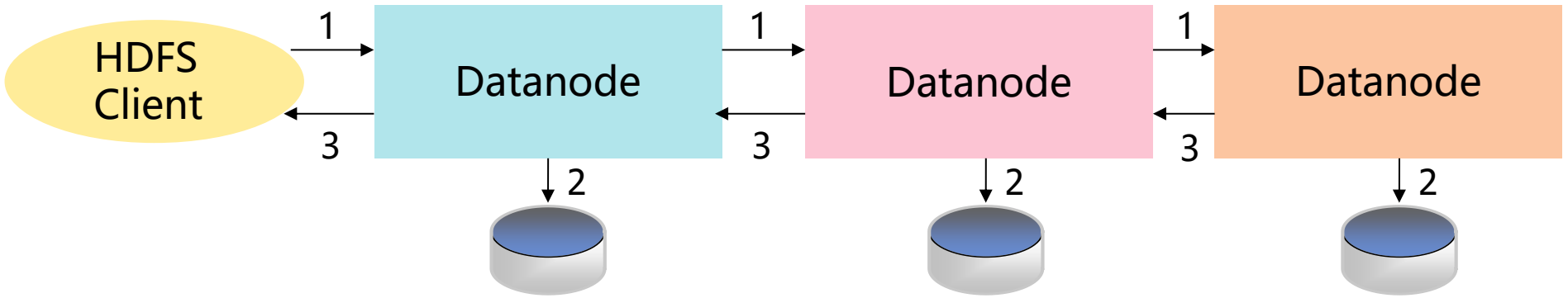
- 整体的数据复制流程分三个阶段:1.流水线建立；2.复制数据；3.关闭流水线

HDFS数据读写过程——写数据



写数据——流水线(pipeline)机制

- 创建流水线：Client通过连接各个块的ip列表来为每个块创建流水线
- 块写入：Client向流水线中写入块数据
- 关闭流水线：当数据块复制到所有DN后，按ip地址列表相反方向，依次反馈写入成功信息至Client，最终Client再反馈给NameNode以更新元数据信息



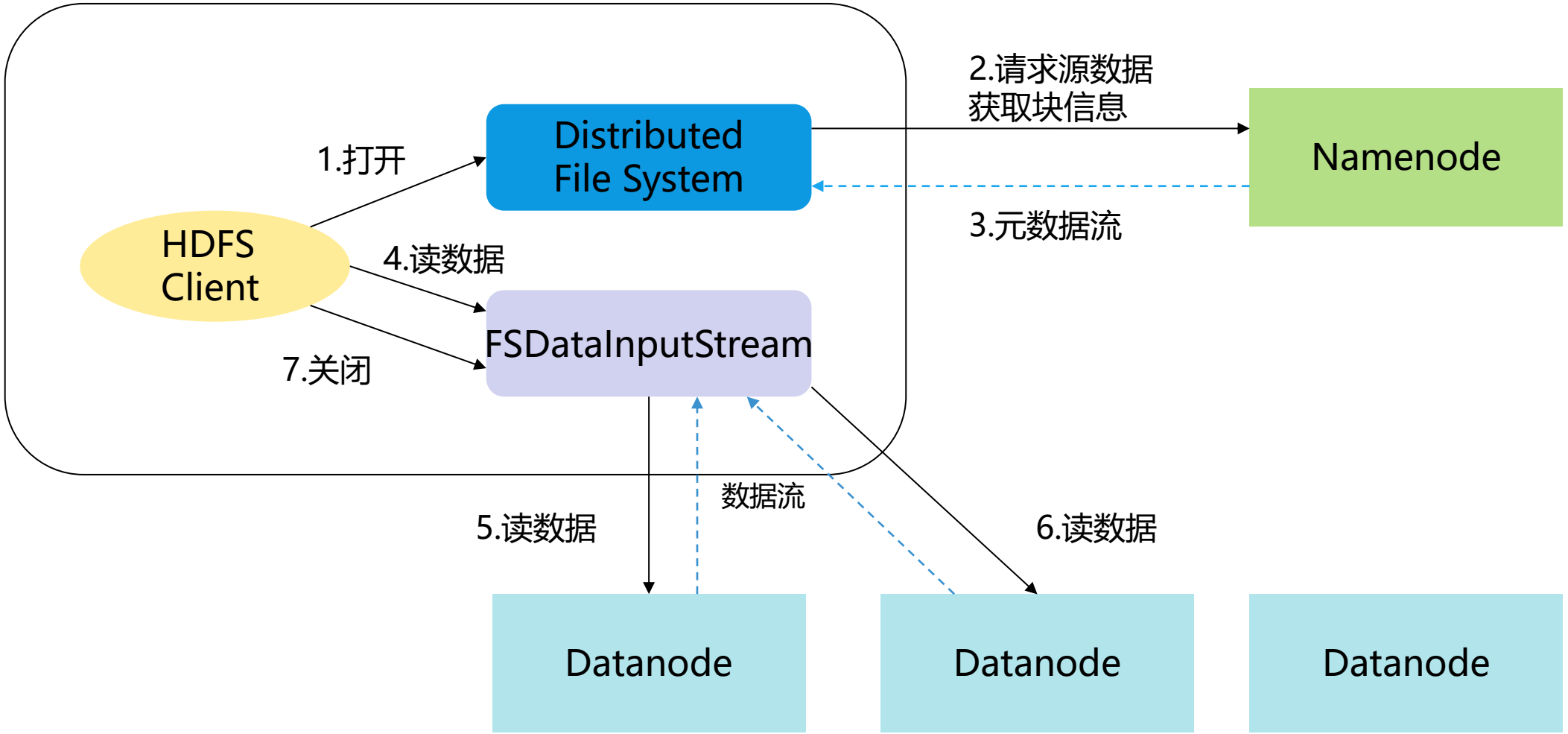
HDFS数据读写过程——写数据

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.Path;
public class Chapter3 {
        public static void main(String[] args) {
                try {
                        Configuration conf = new Configuration();
                        conf.set("fs.defaultFS","hdfs://localhost:9000");
                        conf.set("fs.hdfs.impl",
                                "org.apache.hadoop.hdfs.
                                 DistributedFileSystem");
                        FileSystem fs = FileSystem.get(conf);
                        byte[] buff = "Hello world".getBytes(); // 要写入的内容
                        String filename = "test"; //要写入的文件名
                        FSDataOutputStream os = fs.create(new Path(filename));
                        os.write(buff,0,buff.length);
                        System.out.println("Create:"+ filename);
                        os.close();
                        fs.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

HDFS数据读写过程——读数据请求

- Client请求NameNode要读取文件。
- NameNode根据自己的元数据信息，反馈给client一个DataNode的列表(包含所有的块)。
- Client连接DataNode，读取块数据，合并成文件

HDFS数据读写过程——读数据



HDFS数据读写过程——读数据

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.fs.FSDataInputStream;

public class Chapter3 {
        public static void main(String[] args) {
                try {
                        Configuration conf = new Configuration();
                        conf.set("fs.defaultFS","hdfs://localhost:9000");
                conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");
                        FileSystem fs = FileSystem.get(conf);
                        Path file = new Path("test");
                        FSDataInputStream getIt = fs.open(file);
                        BufferedReader d = new BufferedReader(new InputStreamReader(getIt));
                        String content = d.readLine(); //读取文件一行
                        System.out.println(content);
                        d.close(); //关闭文件
                        fs.close(); //关闭hdfs
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
}
```

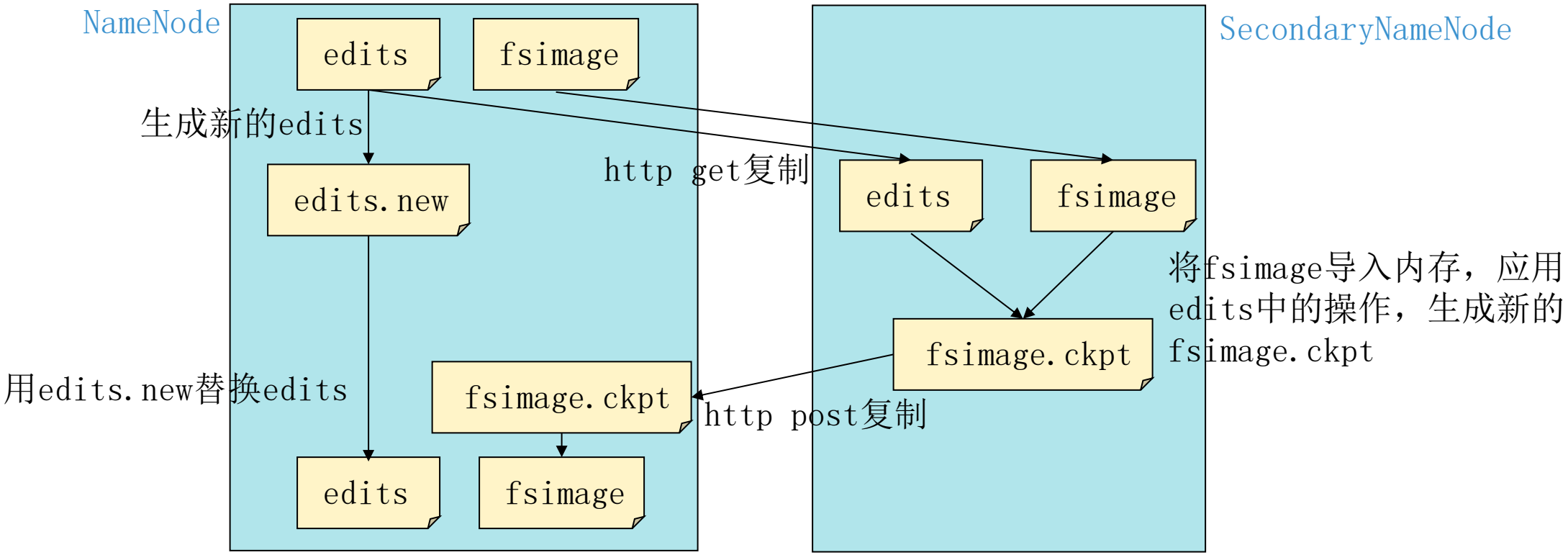
HDFS通信协议

- HDFS是一个部署在集群上的分布式文件系统，因此很多数据需通过网络进行传输
- 所有的HDFS通信协议都是构建在TCP/IP协议基础之上的
- 客户端通过一个可配置的端口向名称节点主动发起TCP连接，并使用客户端协议与名称节点进行交互
- 名称节点和数据节点之间则使用数据节点协议进行交互
- 客户端与数据节点的交互是通过RPC（Remote Procedure Call）来实现的。在设计上，名称节点不会主动发起RPC，而是响应来自客户端和数据节点的RPC请求。

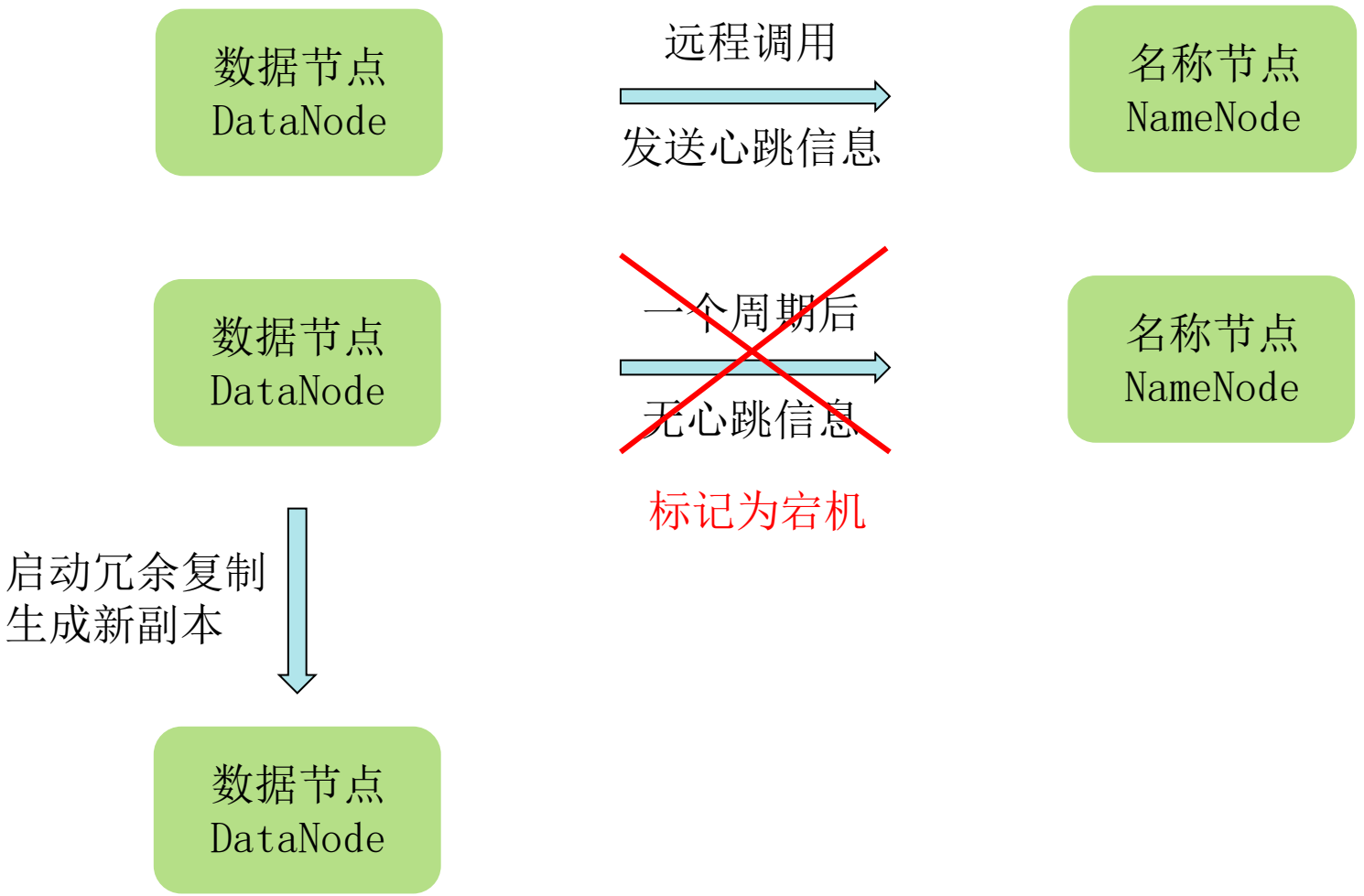
HDFS的容错与恢复机制

- HDFS的容错假设：“**硬件出错看作一种常态，而不是异常**”
- 以“**检测节点和数据错误并进行自动恢复**”为主要目标
- 主要包括以下几种情形：
  - 名称节点出错
  - 数据节点出错
  - 数据出错

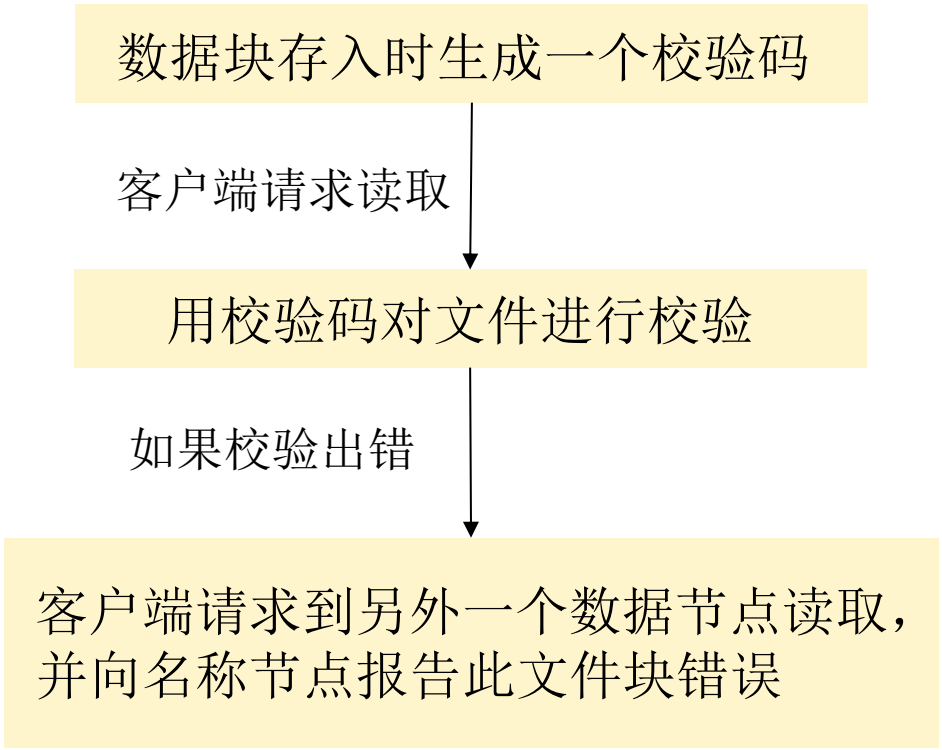
名称节点错误与恢复方法



数据节点错误与恢复方法



数据校验错误与恢复方法



HDFS体系结构的局限性

- HDFS只设置唯一一个名称节点，这样做虽然大大简化了系统设计，但也带来了一些明显的局限性，具体如下：
  - 命名空间的限制：名称节点是保存在内存中的，因此，名称节点能够容纳的对象（文件、块）的个数会受到内存空间大小的限制。
  - 性能的瓶颈：整个分布式文件系统的吞吐量，受限于单个名称节点的吞吐量。
  - 隔离问题：由于集群中只有一个名称节点，只有一个命名空间，因此，无法对不同应用程序进行隔离。
  - 集群的可用性：一旦这个唯一的名称节点发生故障，会导致整个集群变得不可用

其它常用的分布式存储系统

- GFS (Google File System): Google公司为了满足本公司需求而开发的基于Linux的专有分布式文件系统
- Lustre:大规模的、安全可靠的，具备高可用性的集群文件系统，它是由SUN公司开发和维护的
- TFS (Taobao File System):面向互联网服务的分布式文件系统，主要针对海量的非结构化数据，为淘宝提供海量小文件存储
- Ceph、MooseFS、GlusterFS、GridFS等
- Google Colossus FS/Facebook Tectonics FS/SeaweedFS

致谢

- 一小部分图表、文字参考了教材、互联网上的开放资料等，本文件仅供公益性的学习参考，在此表示感谢！如有版权要求请联系：yym@hit.edu.cn，谢谢！