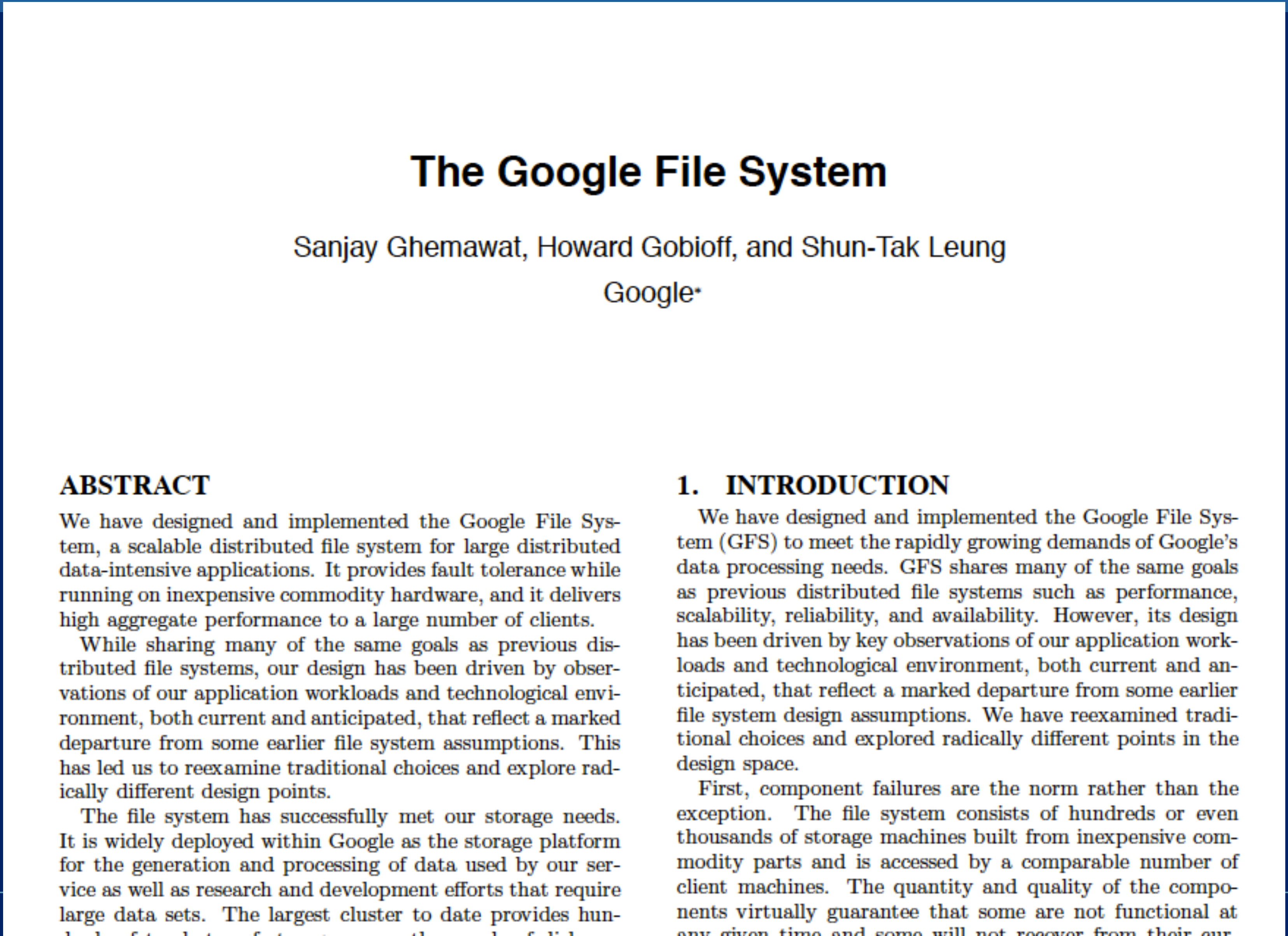
# GFS

(Google File System)

# GFS

*ACM SIGOPS Operating Systems Review.* 37(5) 2003



## The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google*

### ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform for the generation and processing of data used by our service as well as research and development efforts that require large data sets. The largest cluster to date provides hun-

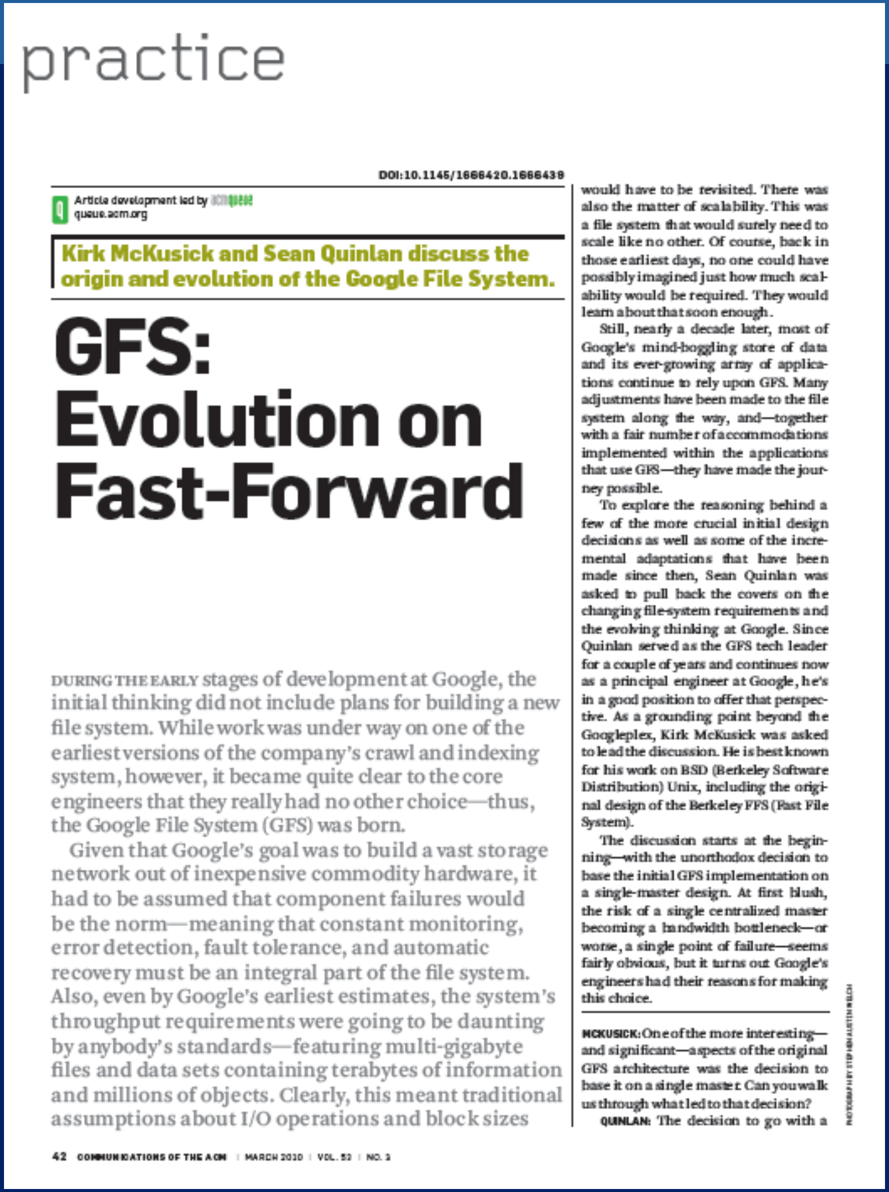
### 1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

First, component failures are the norm rather than the exception. The file system consists of hundreds or even thousands of storage machines built from inexpensive commodity parts and is accessed by a comparable number of client machines. The quantity and quality of the components virtually guarantee that some are not functional at any given time and some will not recover from their cur-

## GFS

*Comm. ACM*, 53(3): 42-49, March 2010.



practice

DOI:10.1145/1666420.1666439

Article development led by acmqueue
queue.acm.org

**Kirk McKusick and Sean Quinlan discuss the origin and evolution of the Google File System.**

# GFS: Evolution on Fast-Forward

DURING THE EARLY stages of development at Google, the initial thinking did not include plans for building a new file system. While work was under way on one of the earliest versions of the company's crawl and indexing system, however, it became quite clear to the core engineers that they really had no other choice—thus, the Google File System (GFS) was born.

Given that Google's goal was to build a vast storage network out of inexpensive commodity hardware, it had to be assumed that component failures would be the norm—meaning that constant monitoring, error detection, fault tolerance, and automatic recovery must be an integral part of the file system. Also, even by Google's earliest estimates, the system's throughput requirements were going to be daunting by anybody's standards—featuring multi-gigabyte files and data sets containing terabytes of information and millions of objects. Clearly, this meant traditional assumptions about I/O operations and block sizes would have to be revisited. There was also the matter of scalability. This was a file system that would surely need to scale like no other. Of course, back in those earliest days, no one could have possibly imagined just how much scalability would be required. They would learn about that soon enough.

Still, nearly a decade later, most of Google's mind-boggling store of data and its ever-growing array of applications continue to rely upon GFS. Many adjustments have been made to the file system along the way, and—together with a fair number of accommodations implemented within the applications that use GFS—they have made the journey possible.

To explore the reasoning behind a few of the more crucial initial design decisions as well as some of the incremental adaptations that have been made since then, Sean Quinlan was asked to pull back the covers on the changing file-system requirements and the evolving thinking at Google. Since Quinlan served as the GFS tech leader for a couple of years and continues now as a principal engineer at Google, he's in a good position to offer that perspective. As a grounding point beyond the Googleplex, Kirk McKusick was asked to lead the discussion. He is best known for his work on BSD (Berkeley Software Distribution) Unix, including the original design of the Berkeley FFS (Fast File System).

The discussion starts at the beginning—with the unorthodox decision to base the initial GFS implementation on a single-master design. At first blush, the risk of a single centralized master becoming a bandwidth bottleneck—or worse, a single point of failure—seems fairly obvious, but it turns out Google's engineers had their reasons for making this choice.

**MCKUSICK:** One of the more interesting—and significant—aspects of the original GFS architecture was the decision to base it on a single master. Can you walk us through what led to that decision?

**QUINLAN:** The decision to go with a

PHOTOGRAPH BY STEPHEN AUSTIN WELCH

## GFS

When Google started, it committed to building a huge scalable distributed storage capability using cheap commodity components (PCs/servers).

Huge = file-sizes of many gigabytes; data-set sizes of many terabytes

Cheap = unreliable

"Normal Failure" became something that had to be dealt with by the file storage system…

- Constant monitoring
- Error detection
- Fault-tolerance
- Automatic recovery

GFS delivers all this

## GFS Architecture

Files in GFS are divided into fixed-sized 64Mb blocks called chunks.

Each chunk is assigned a unique i.d. called a chunk handle.

Chunks are stored on chunkservers.

Fault-tolerance provided by replicating each chunk across multiple servers.

- Typically three replicas, but user-configurable

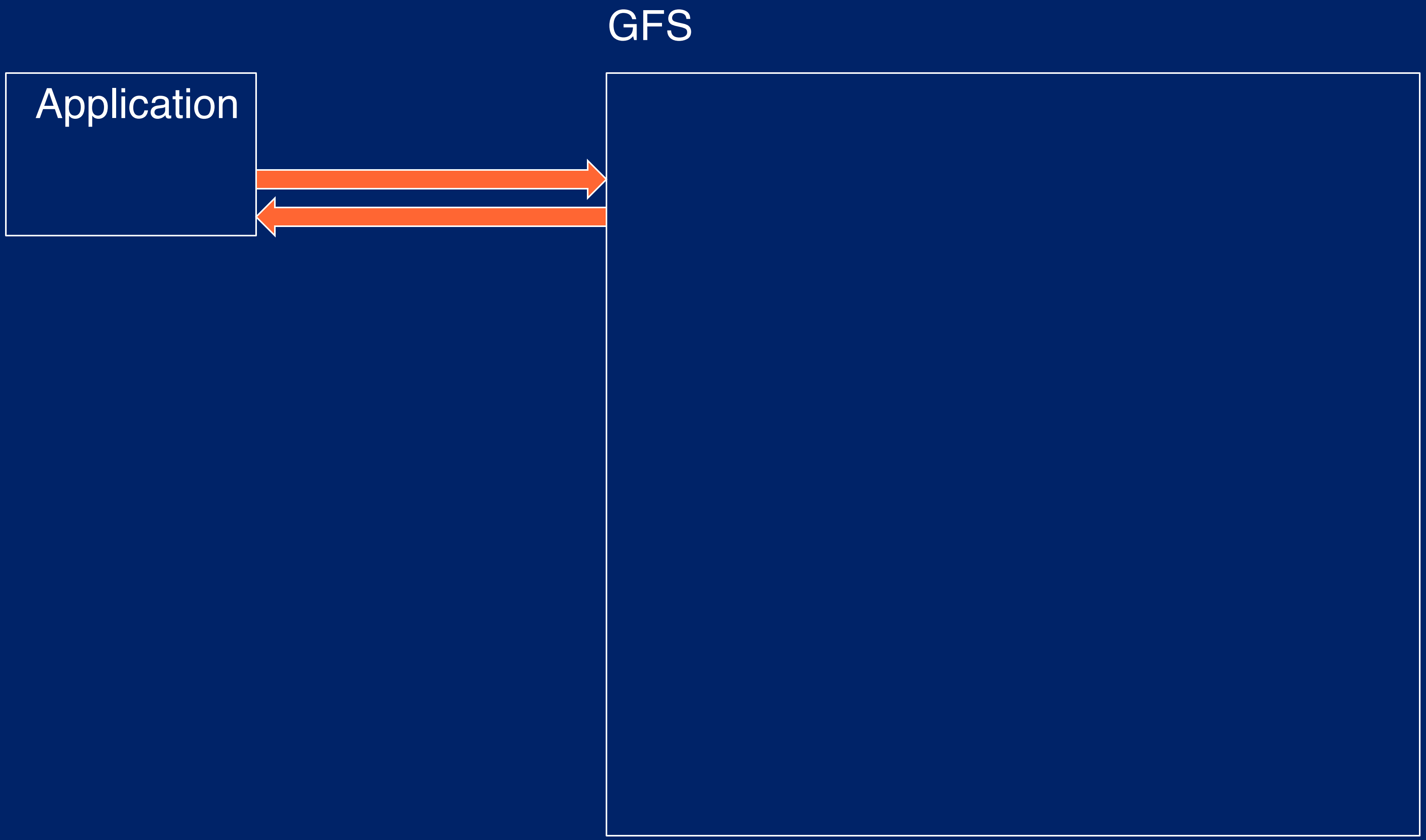
Meta-data needs to be recorded so GFS knows what chunk is where.

GFS consists of three main component types:

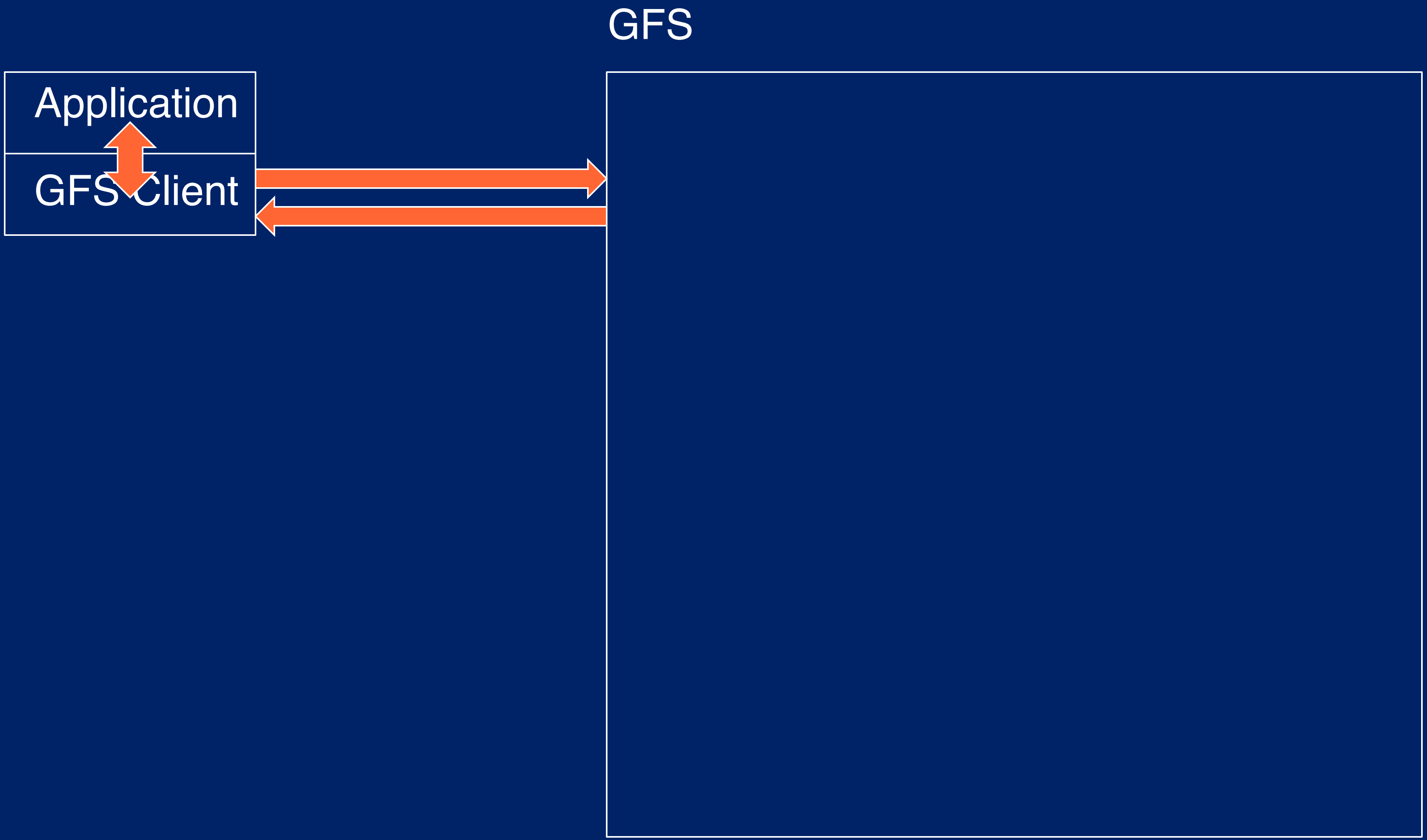
- GFS Master Server (single) – maintains all file system metadata
- GFS Chunkserver (multiple)
- GFS Client (multiple)

## GFS Architecture

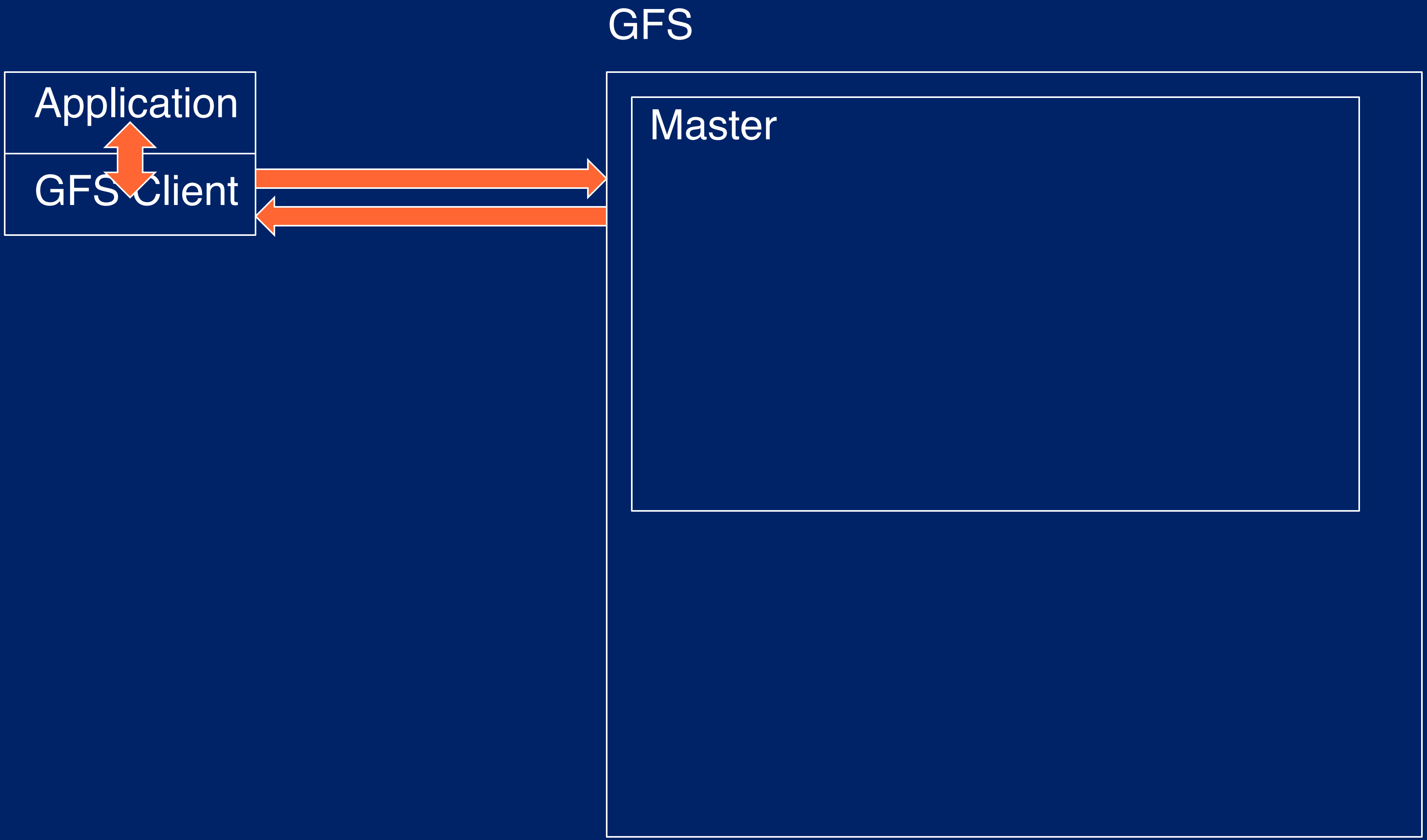
(from Fig.1 of Ghemawat, Gobioff, and Leung)



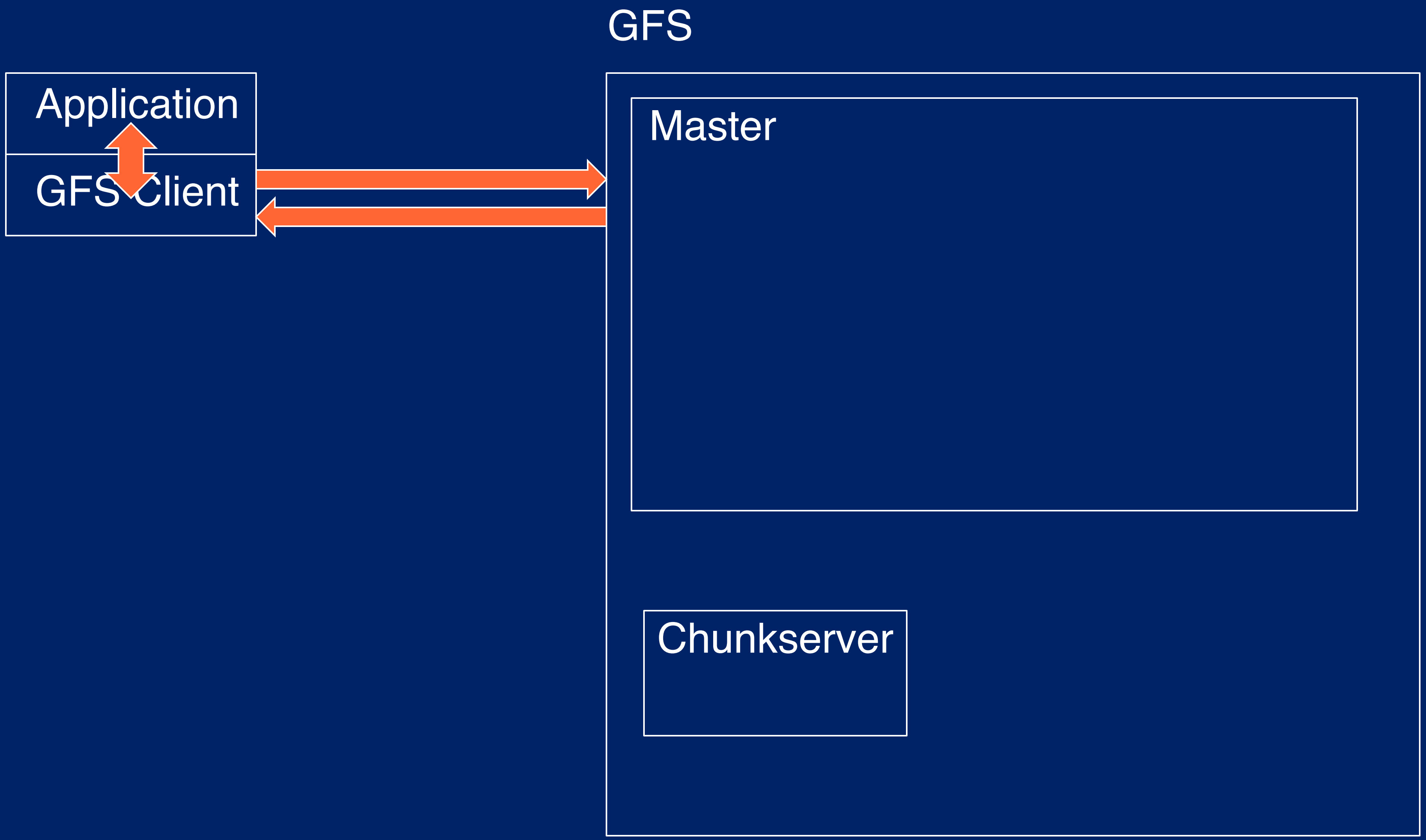
## GFS Architecture (from Fig.1 of Ghemawat, Gobioff, and Leung)



## GFS Architecture (from Fig.1 of Ghemawat, Gobioff, and Leung)

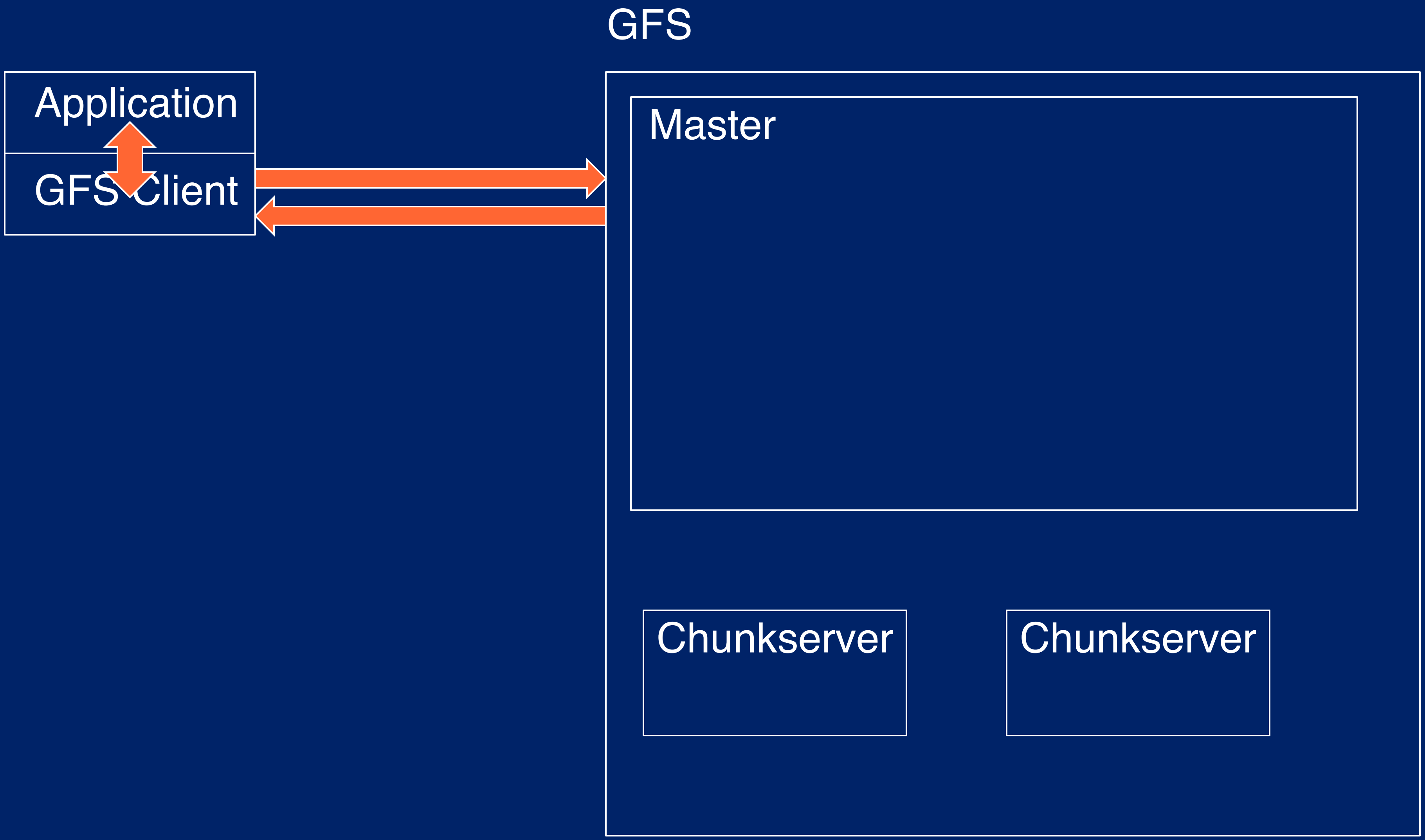


## GFS Architecture (from Fig.1 of Ghemawat, Gobioff, and Leung)



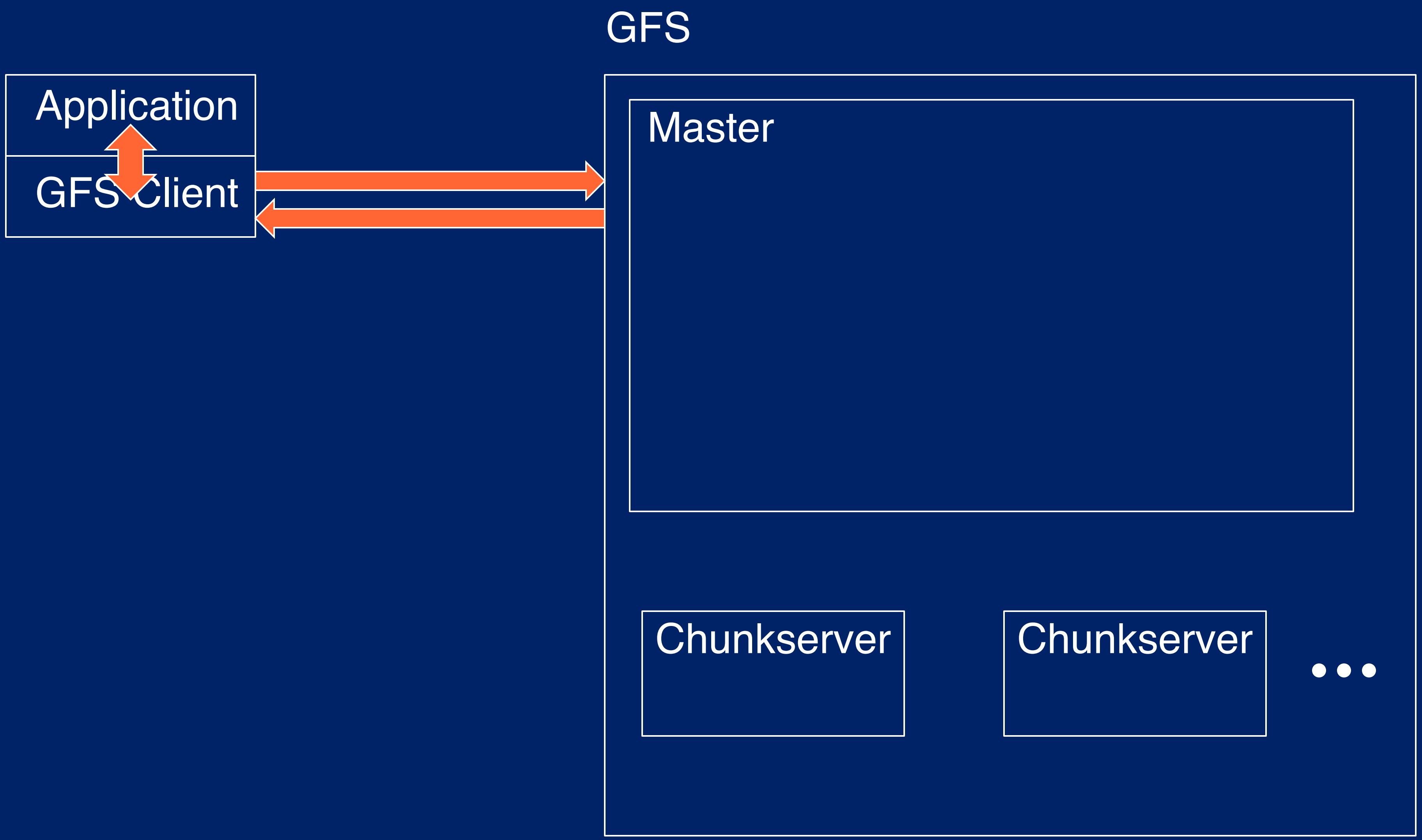
## GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)

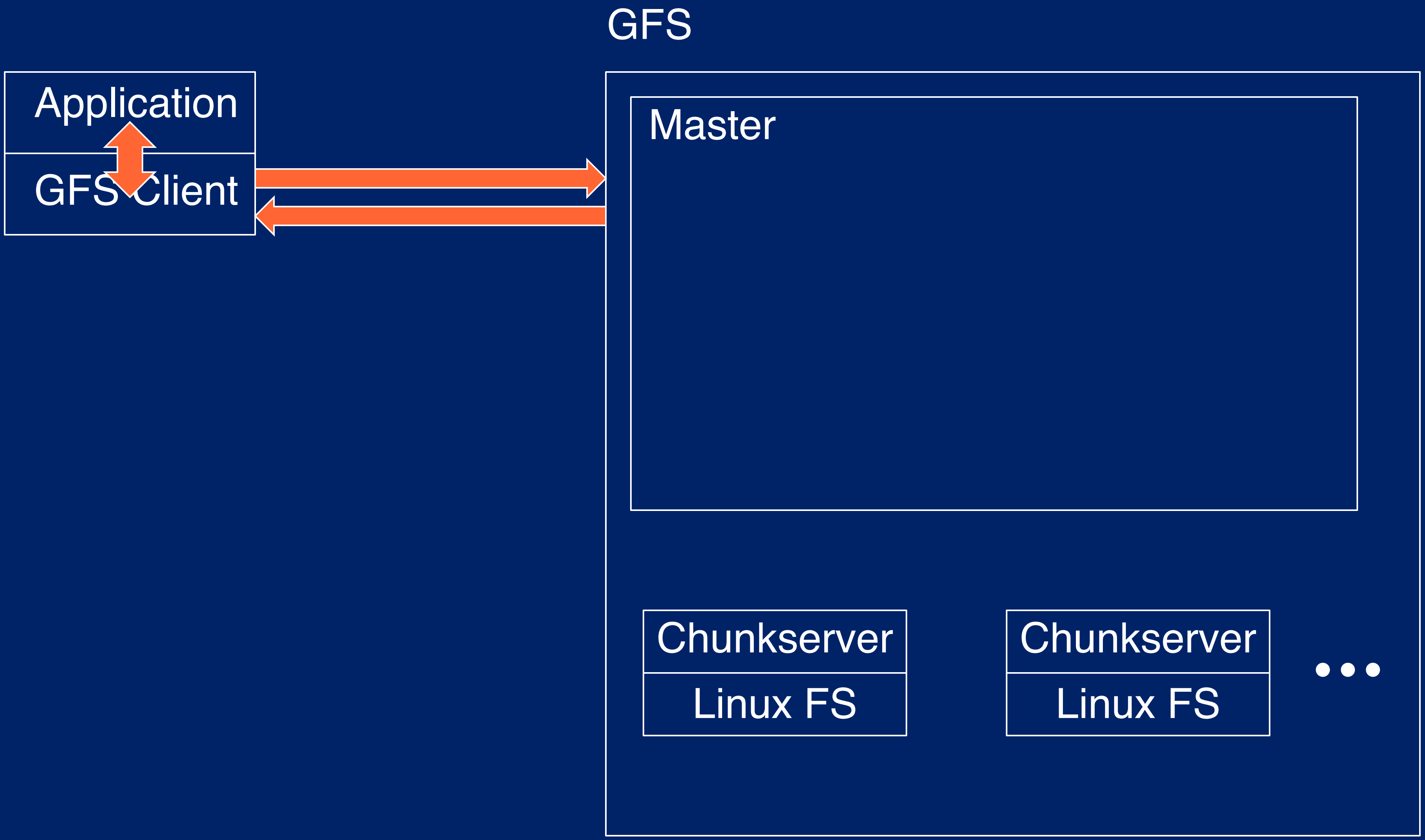


# GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)

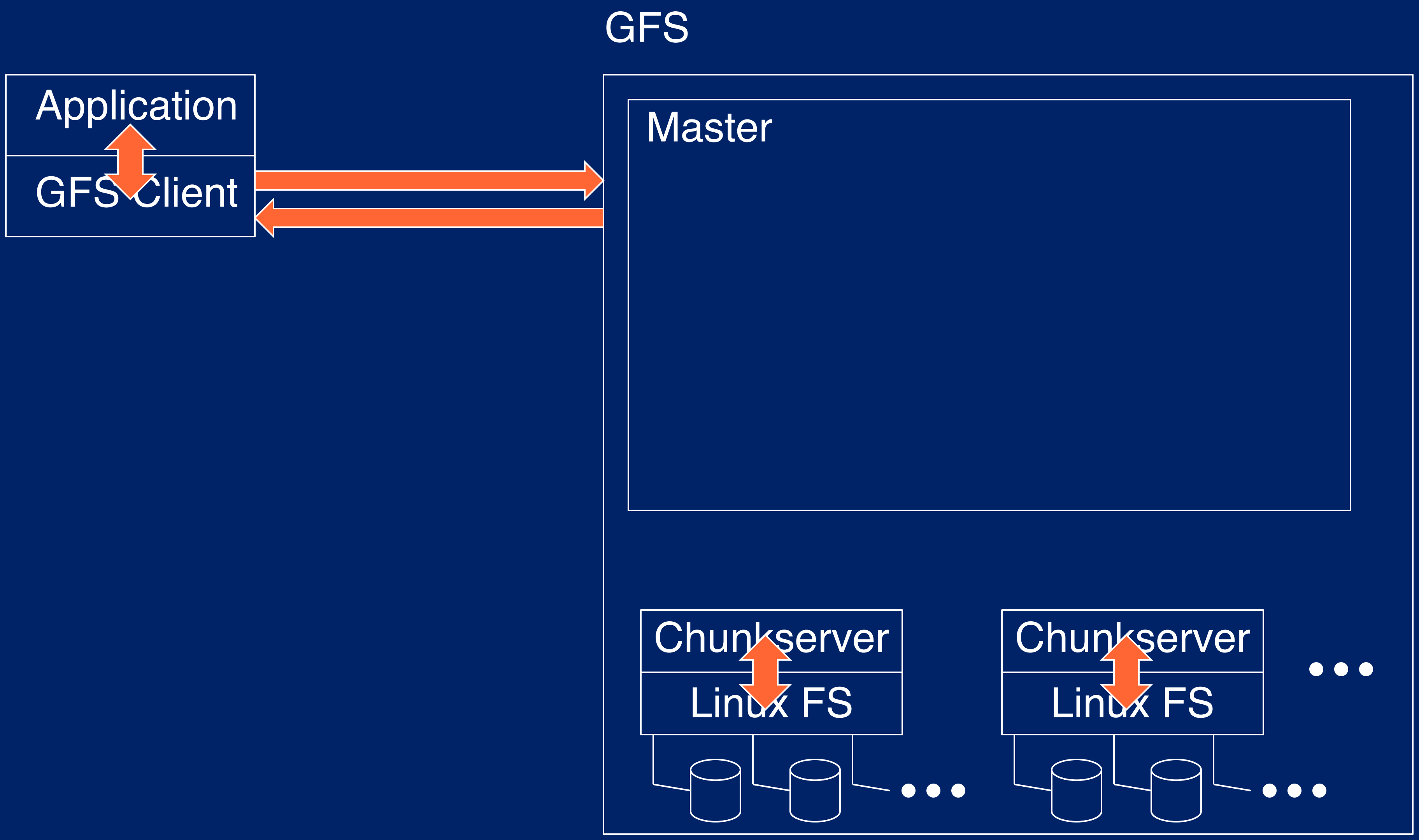


## GFS Architecture (from Fig.1 of Ghemawat, Gobioff, and Leung)



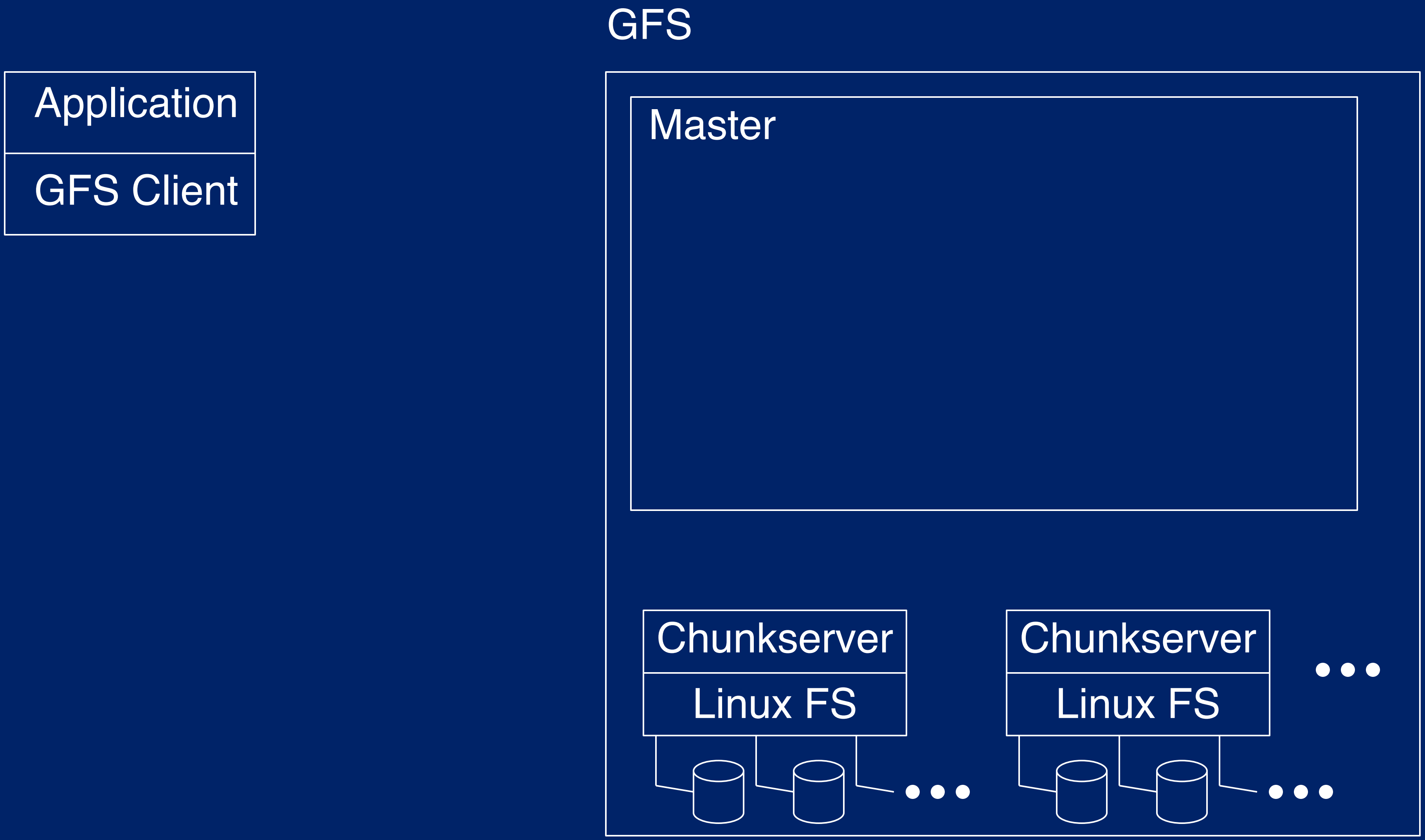
## GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)



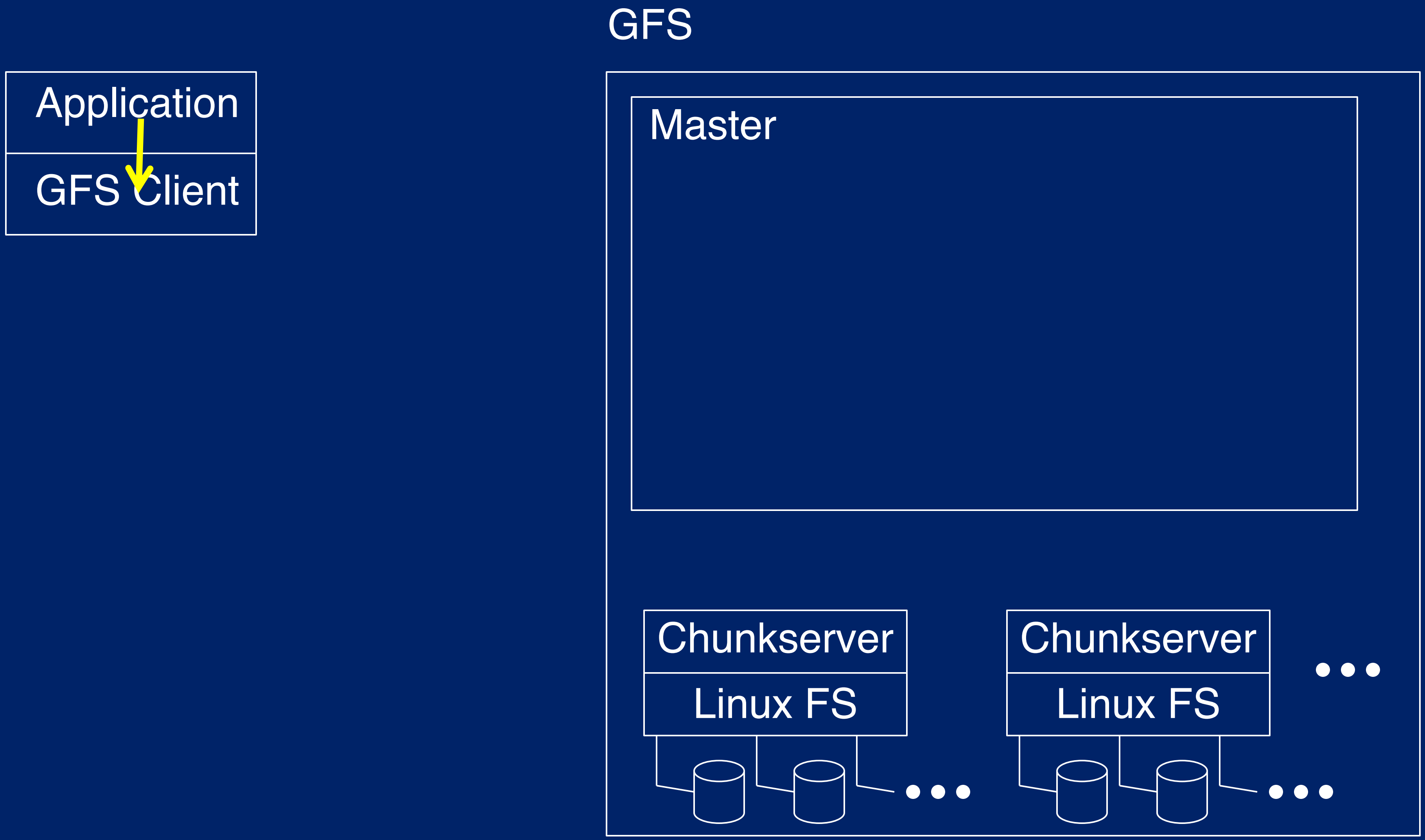
## GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)

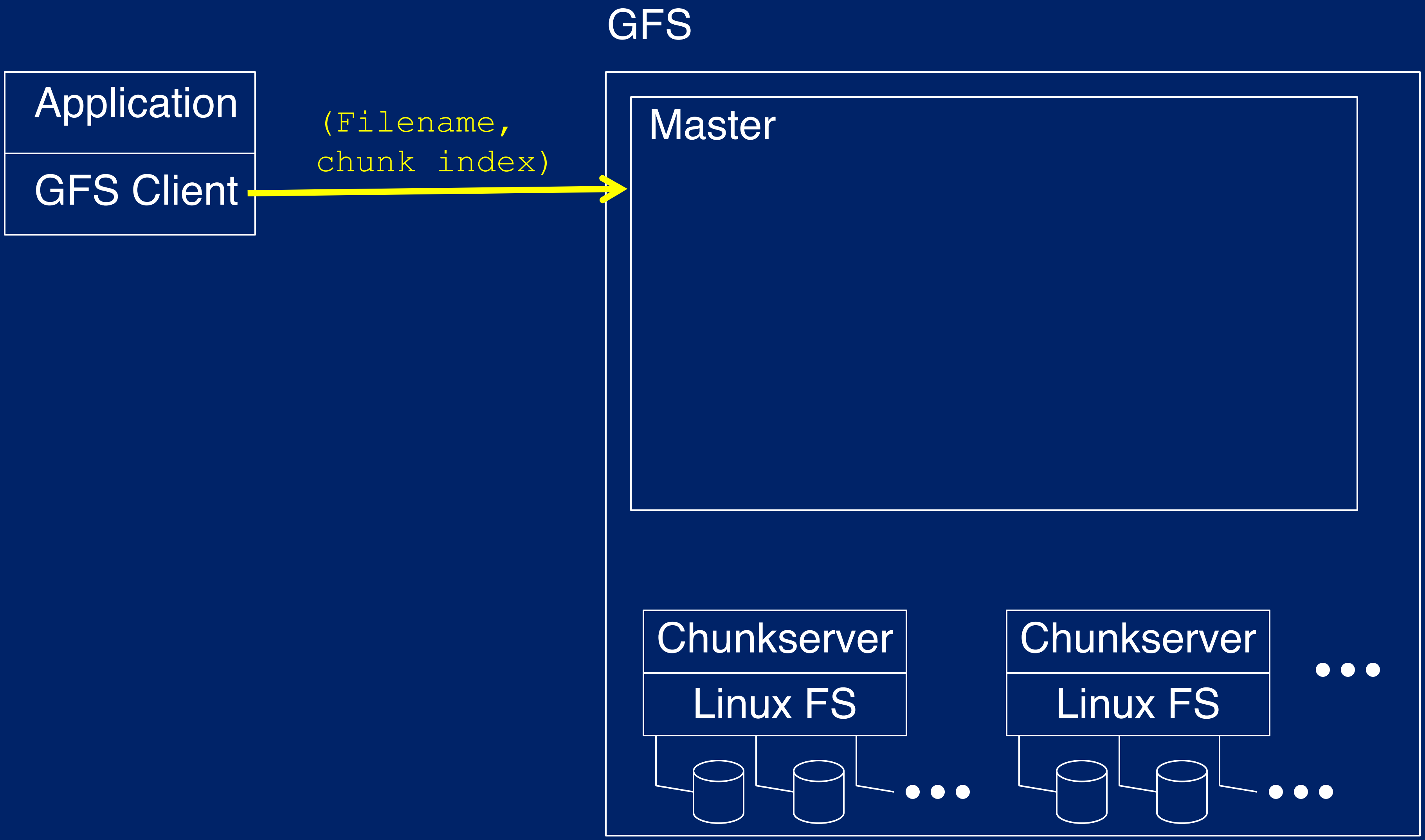


## GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)

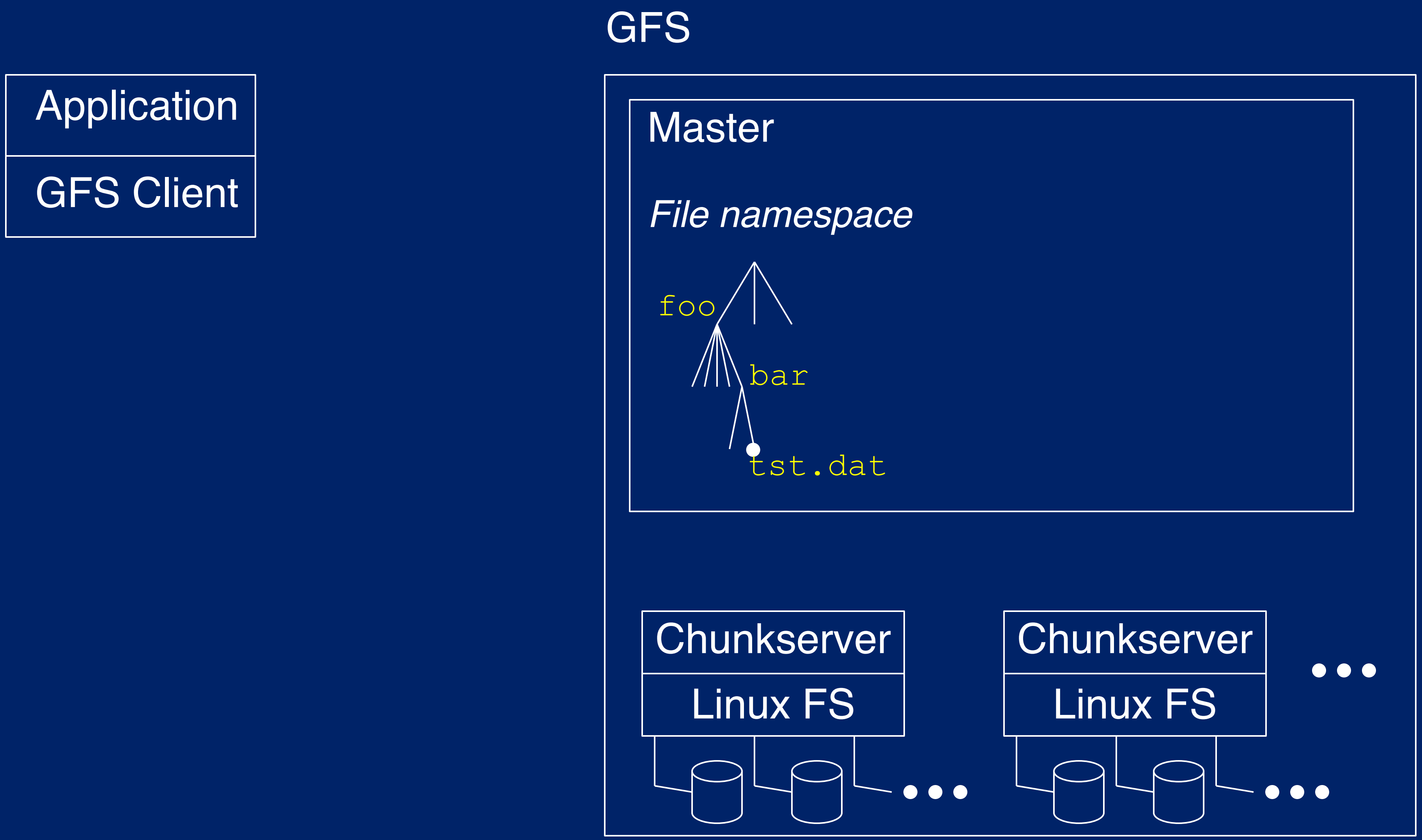


## GFS Architecture (from Fig.1 of Ghemawat, Gobioff, and Leung)



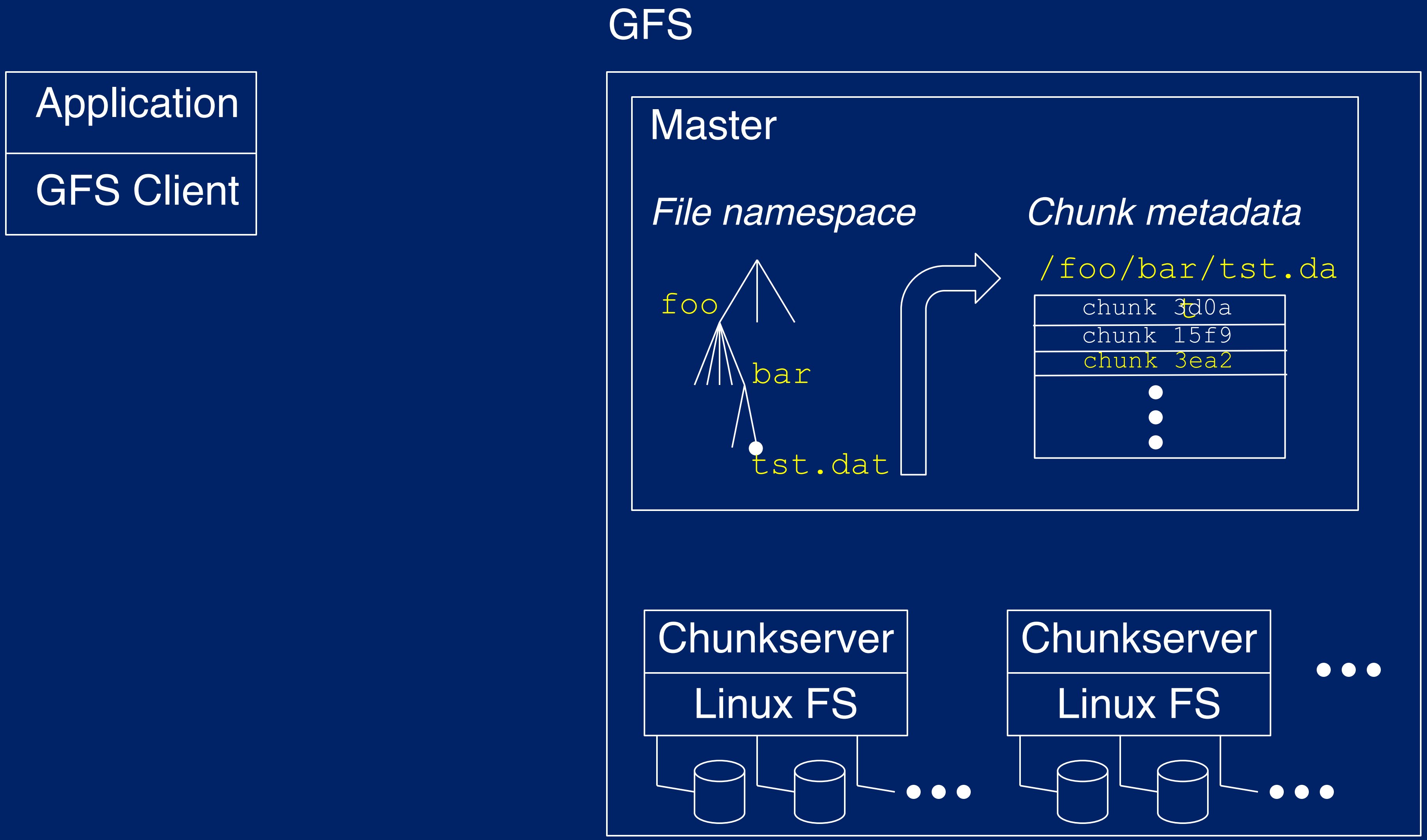
# GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)



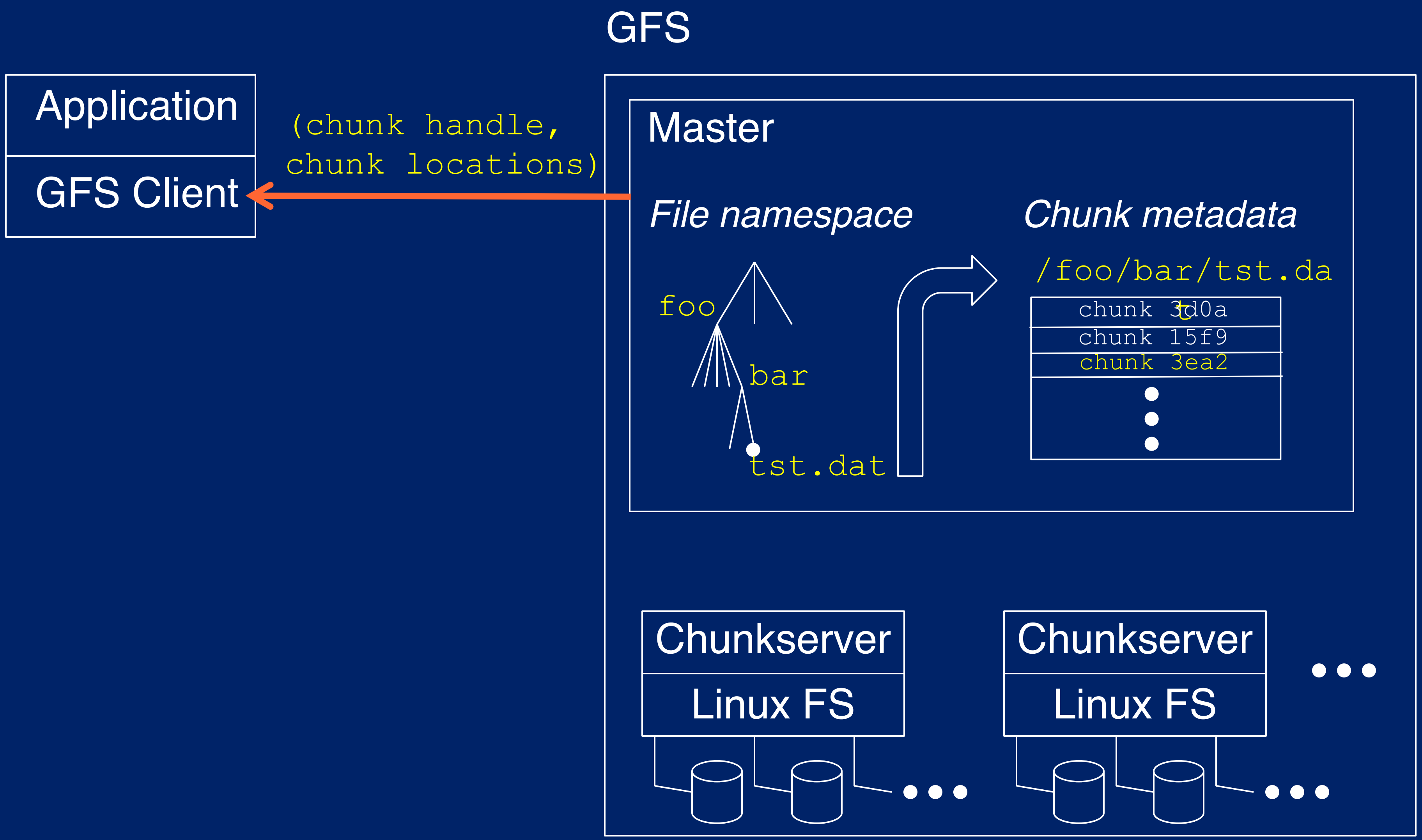
# GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)



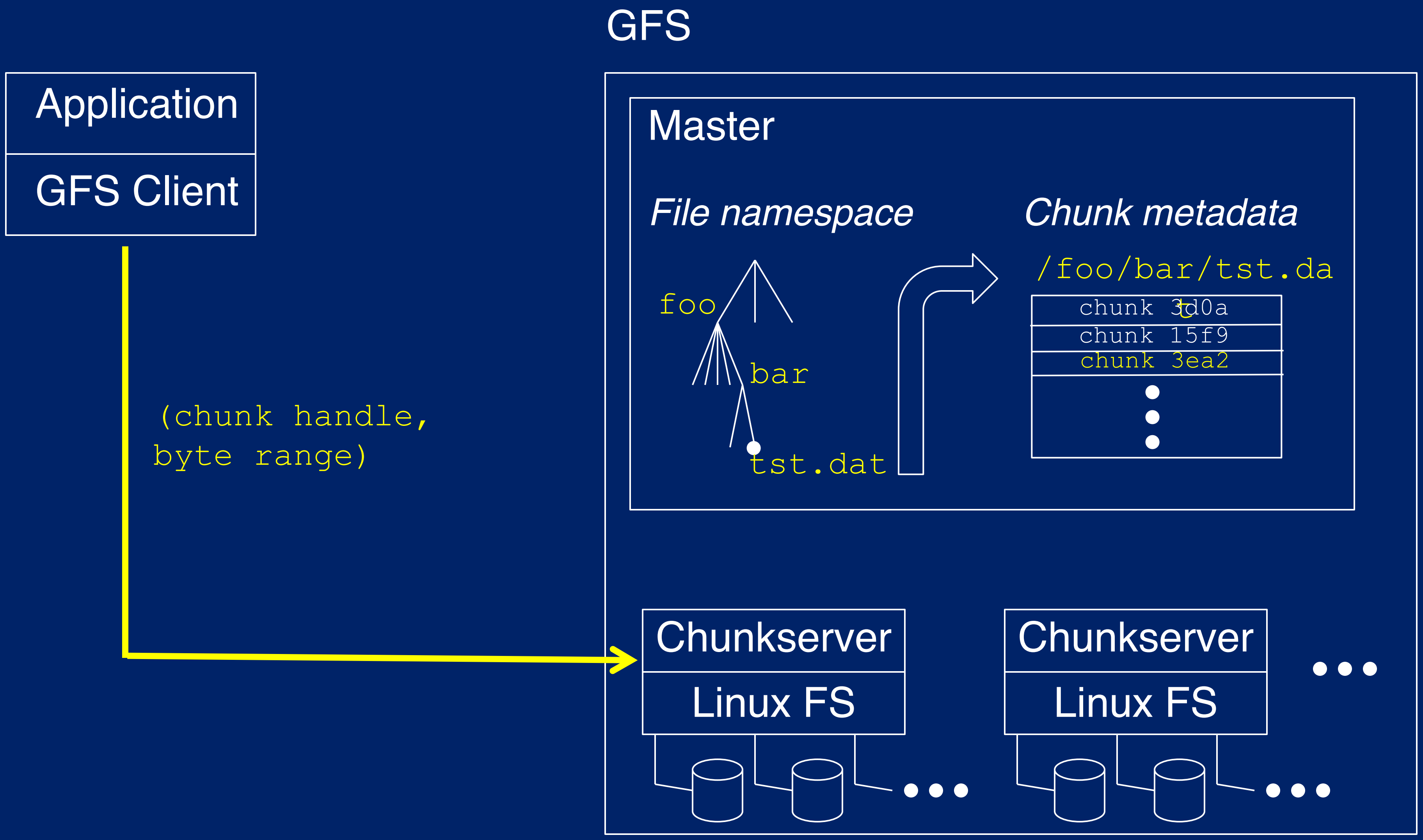
# GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)



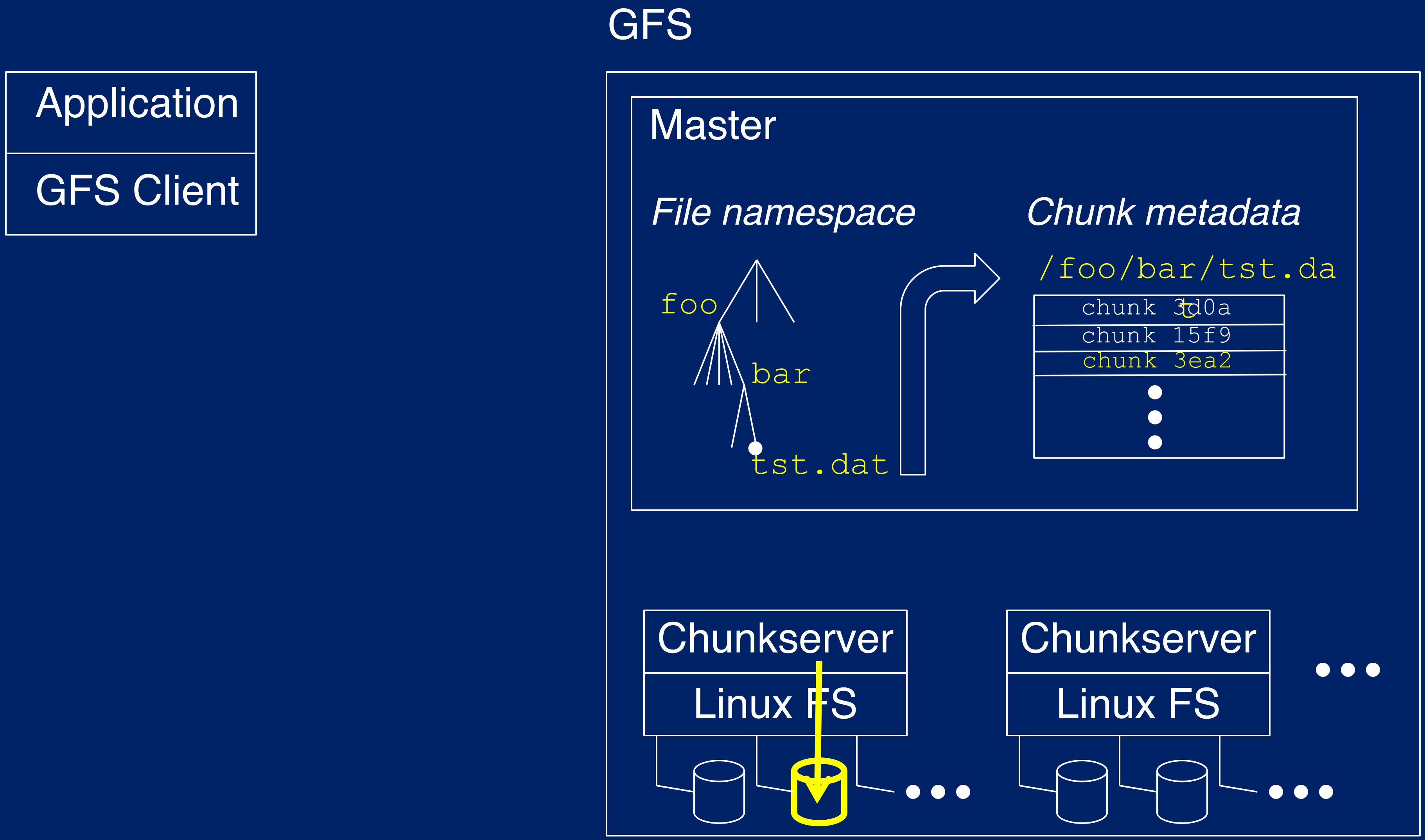
## GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)

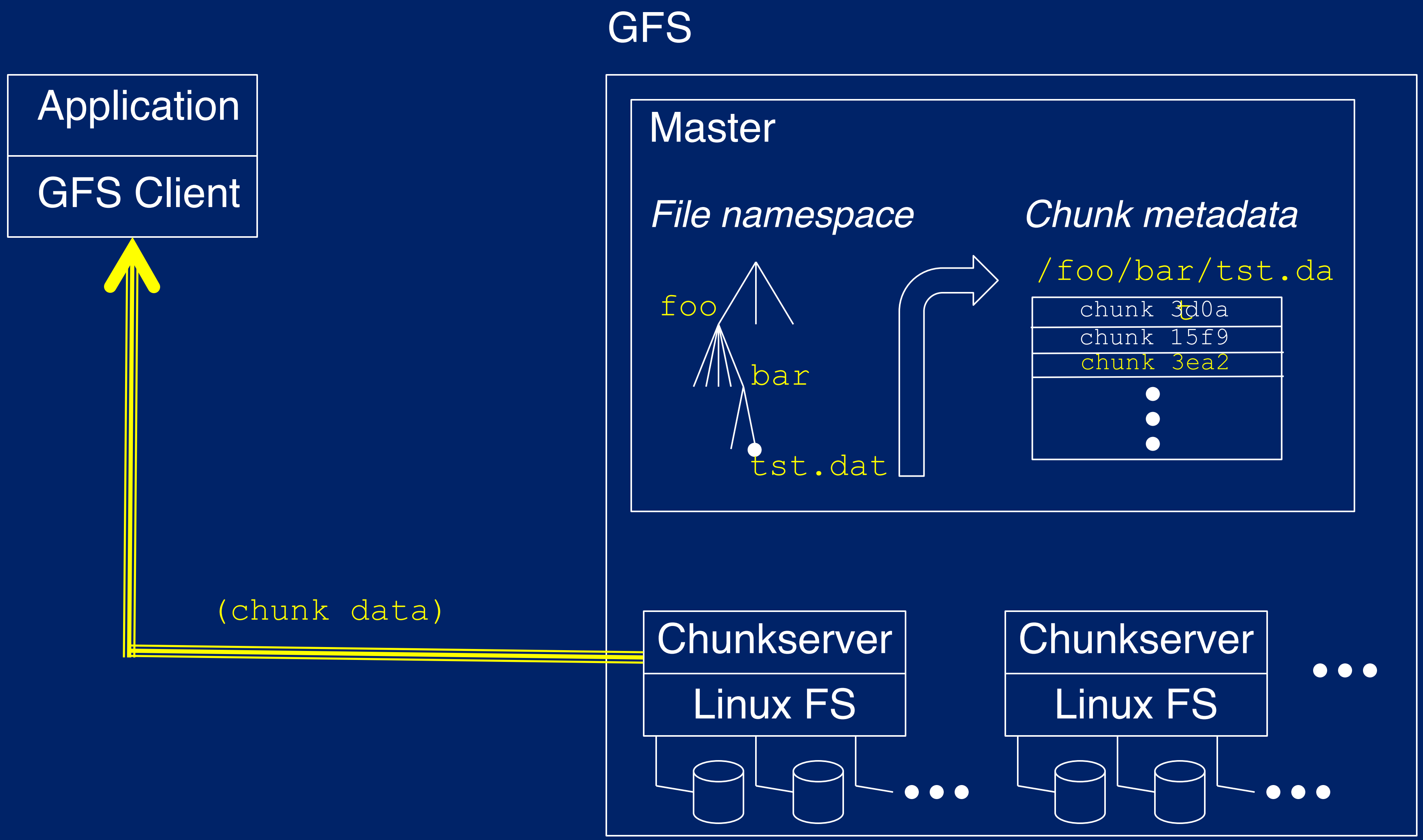


# GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)

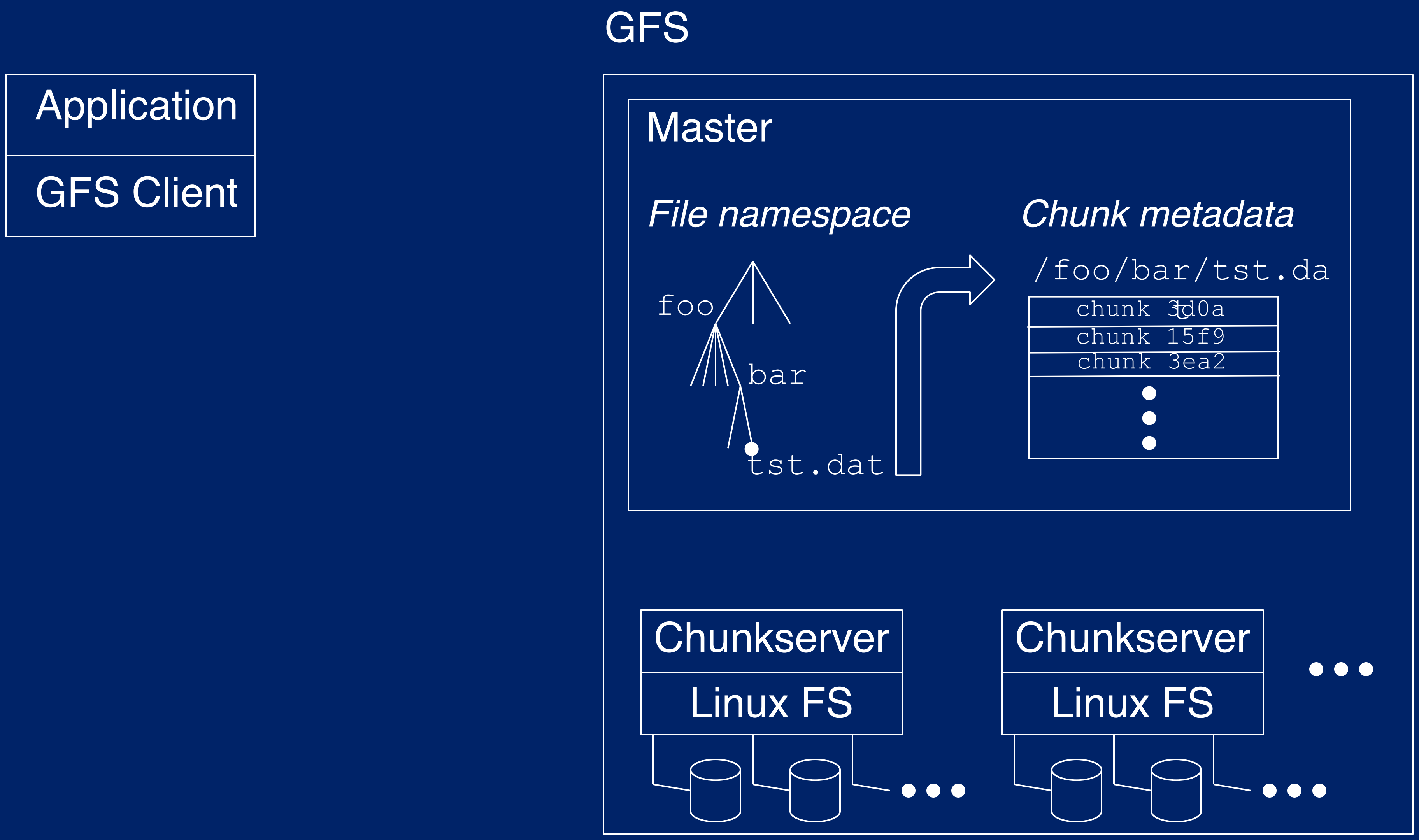


## GFS Architecture (from Fig.1 of Ghemawat, Gobioff, and Leung)



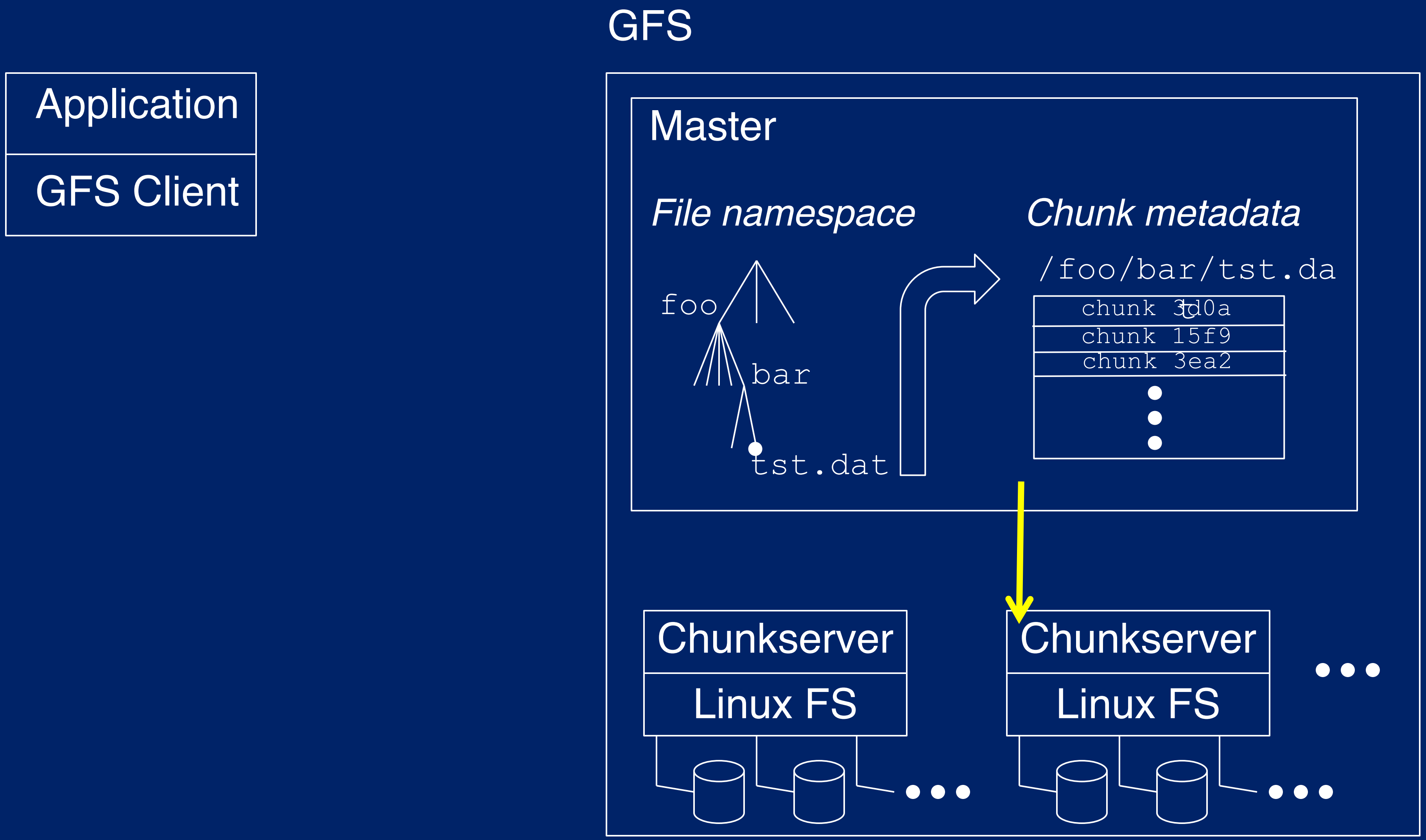
# GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)



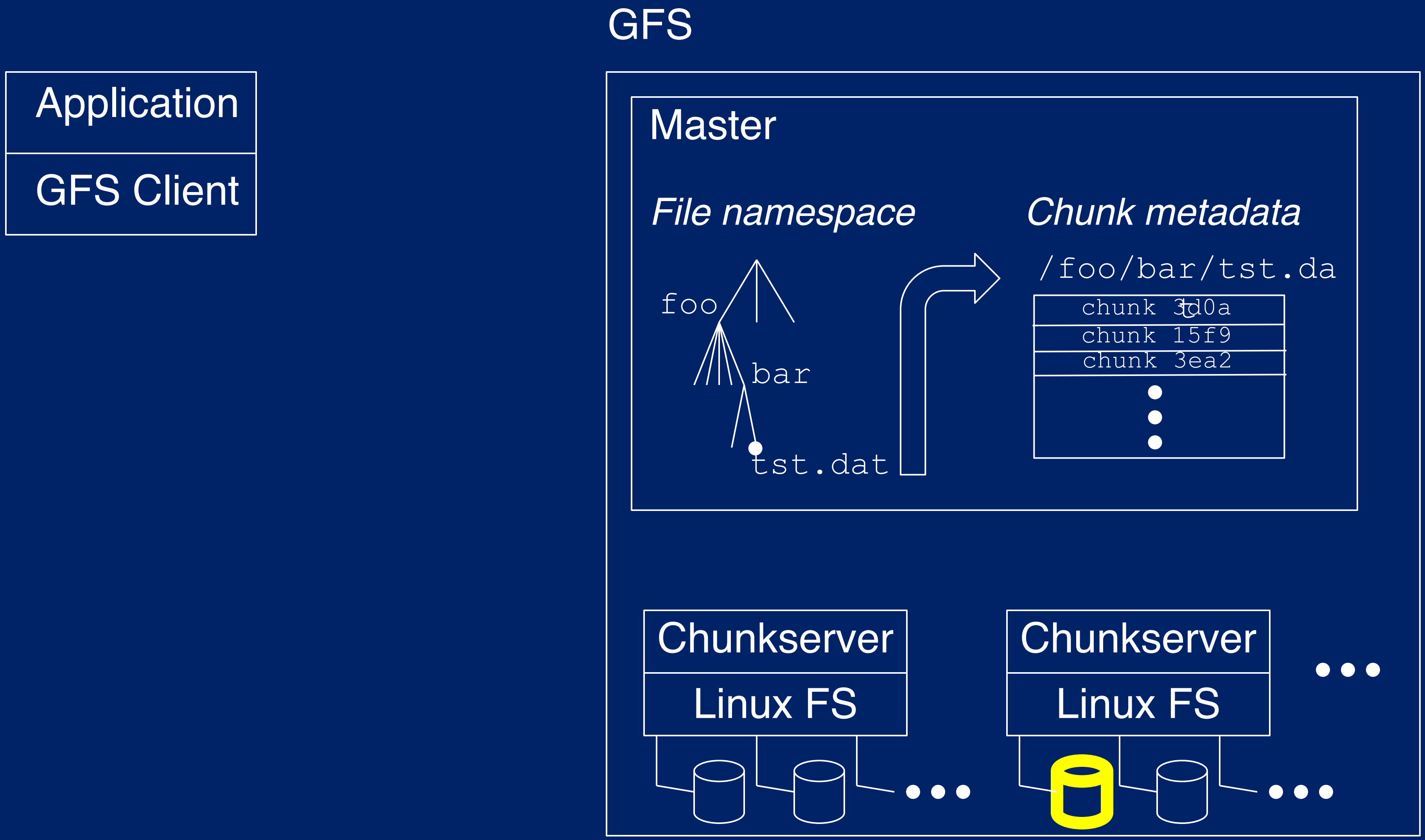
## GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)



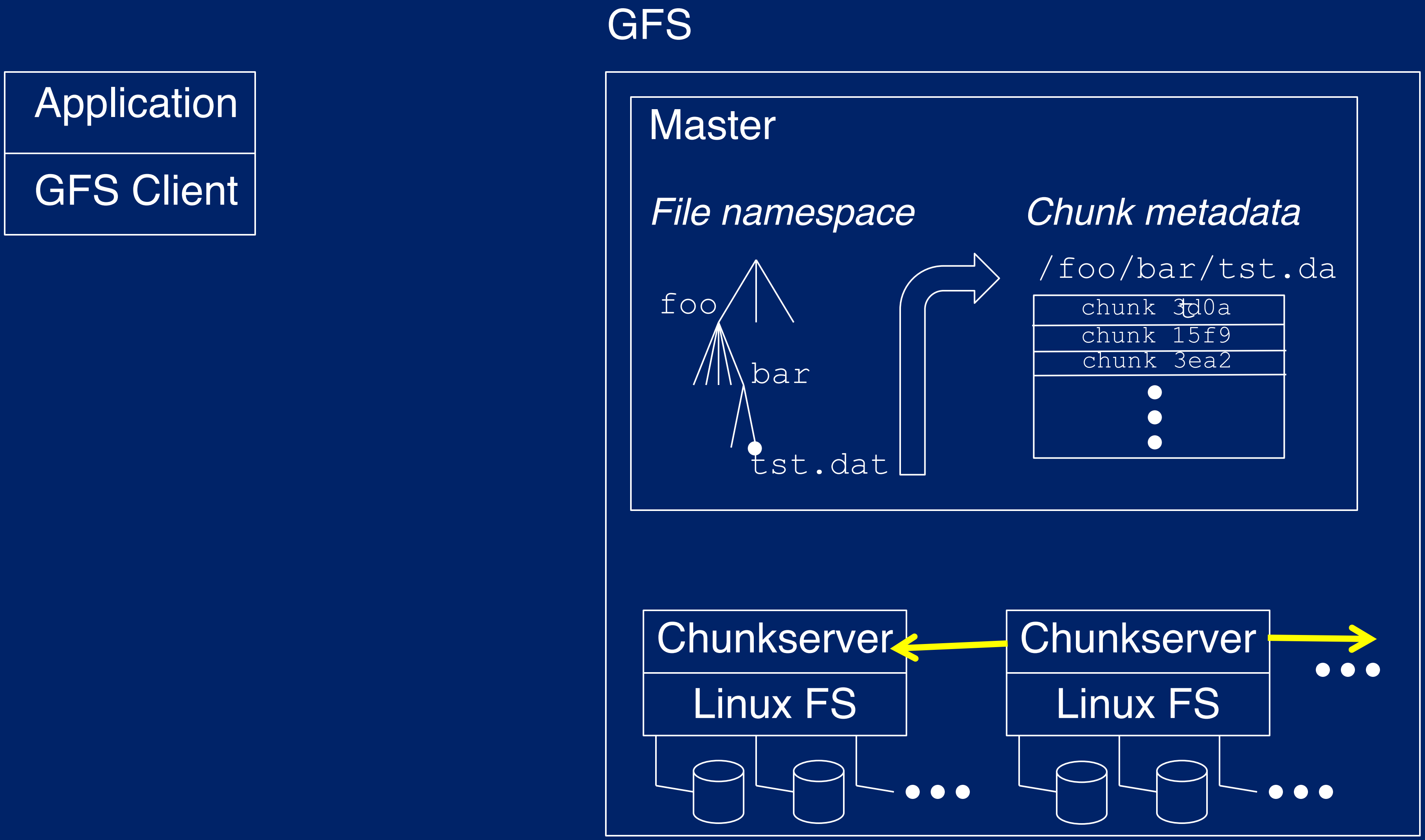
## GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)



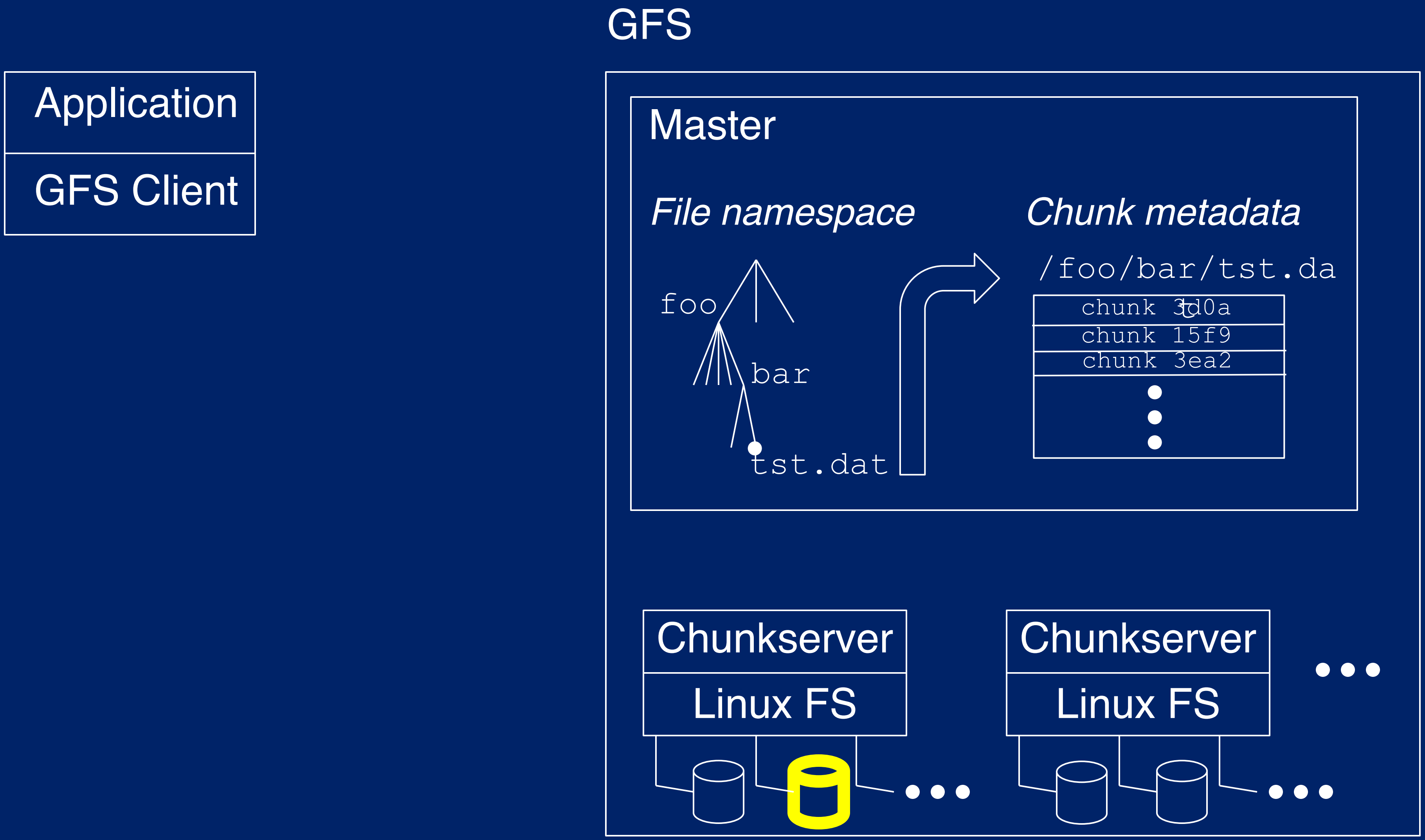
# GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)

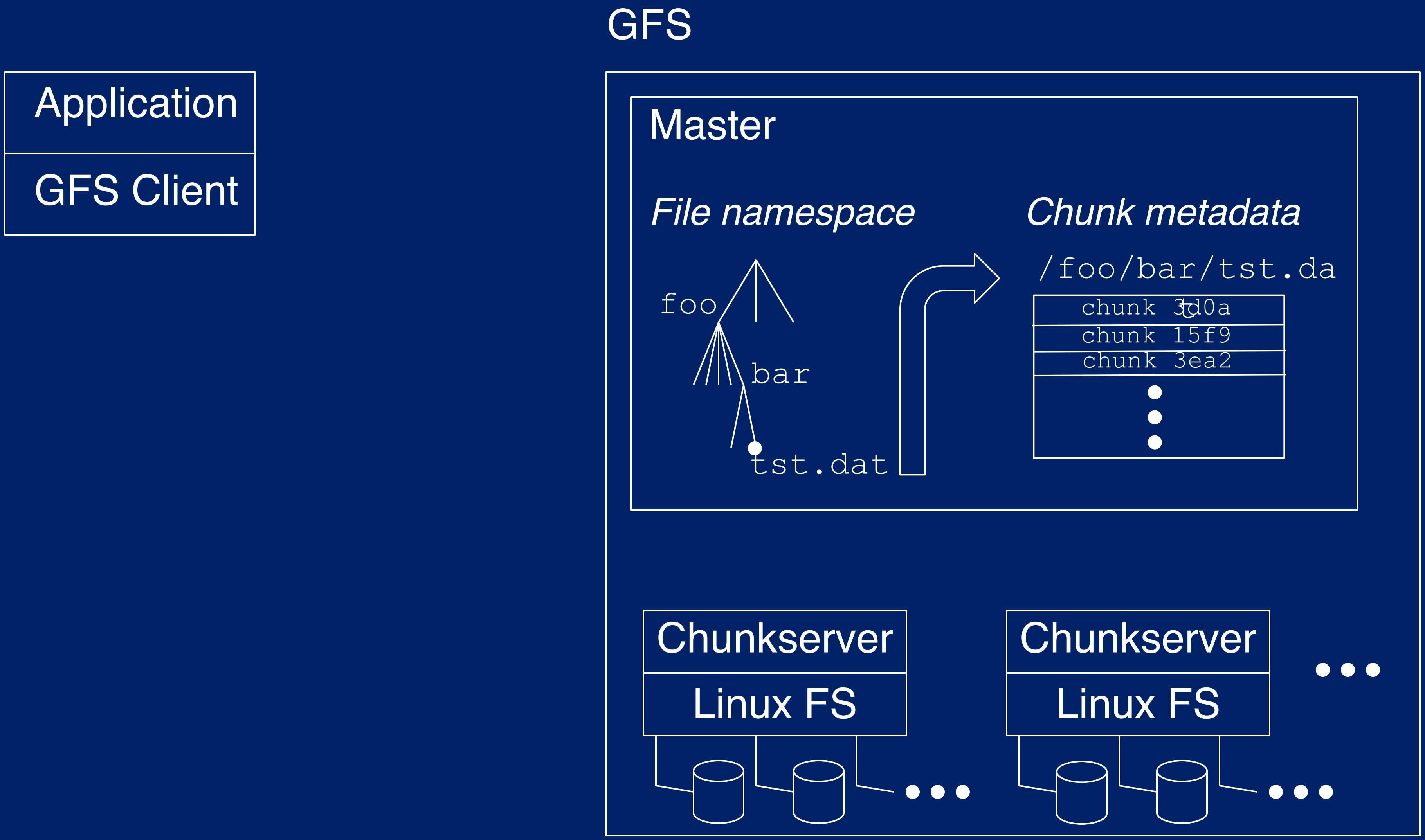


# GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)

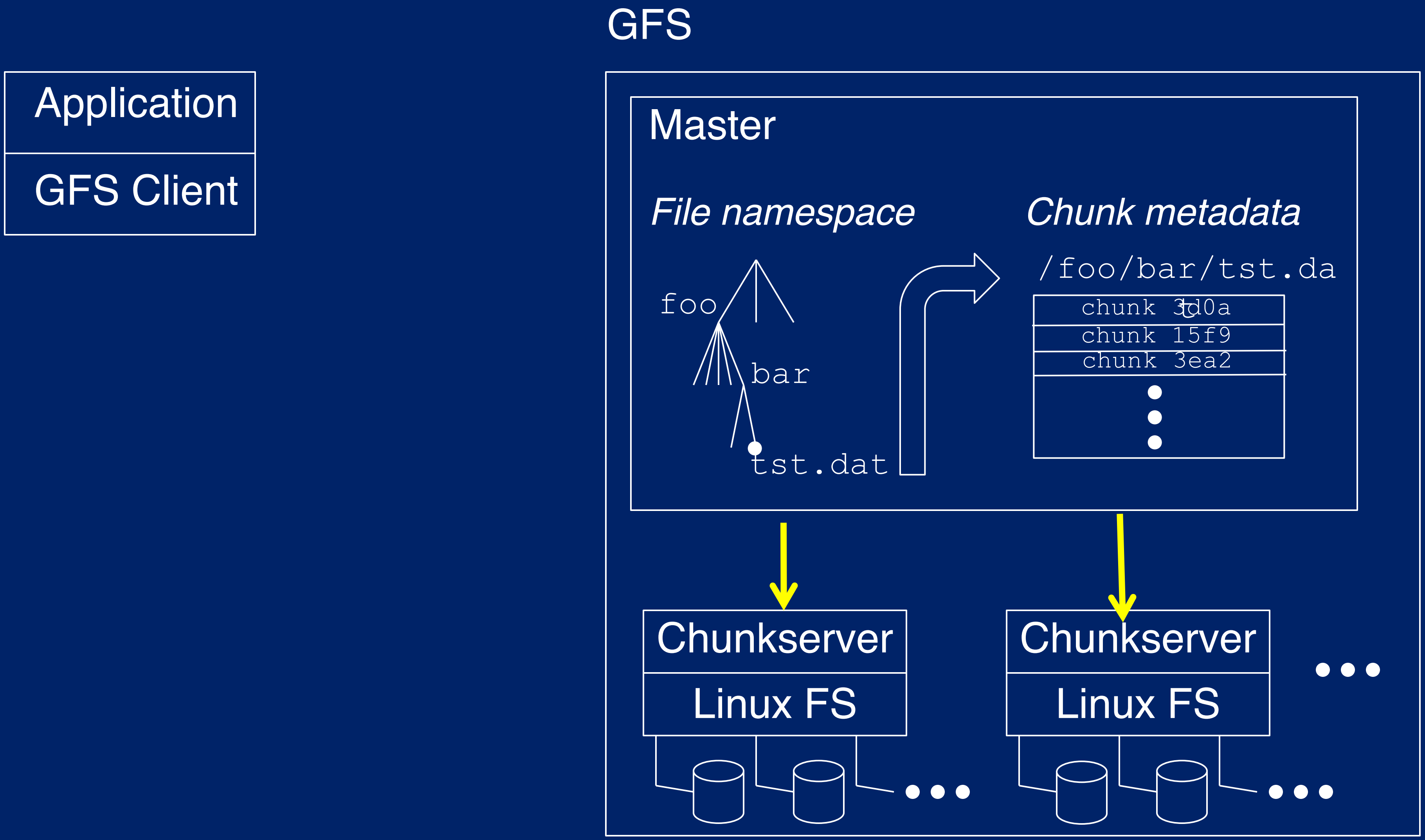


## GFS Architecture (from Fig.1 of Ghemawat, Gobioff, and Leung)



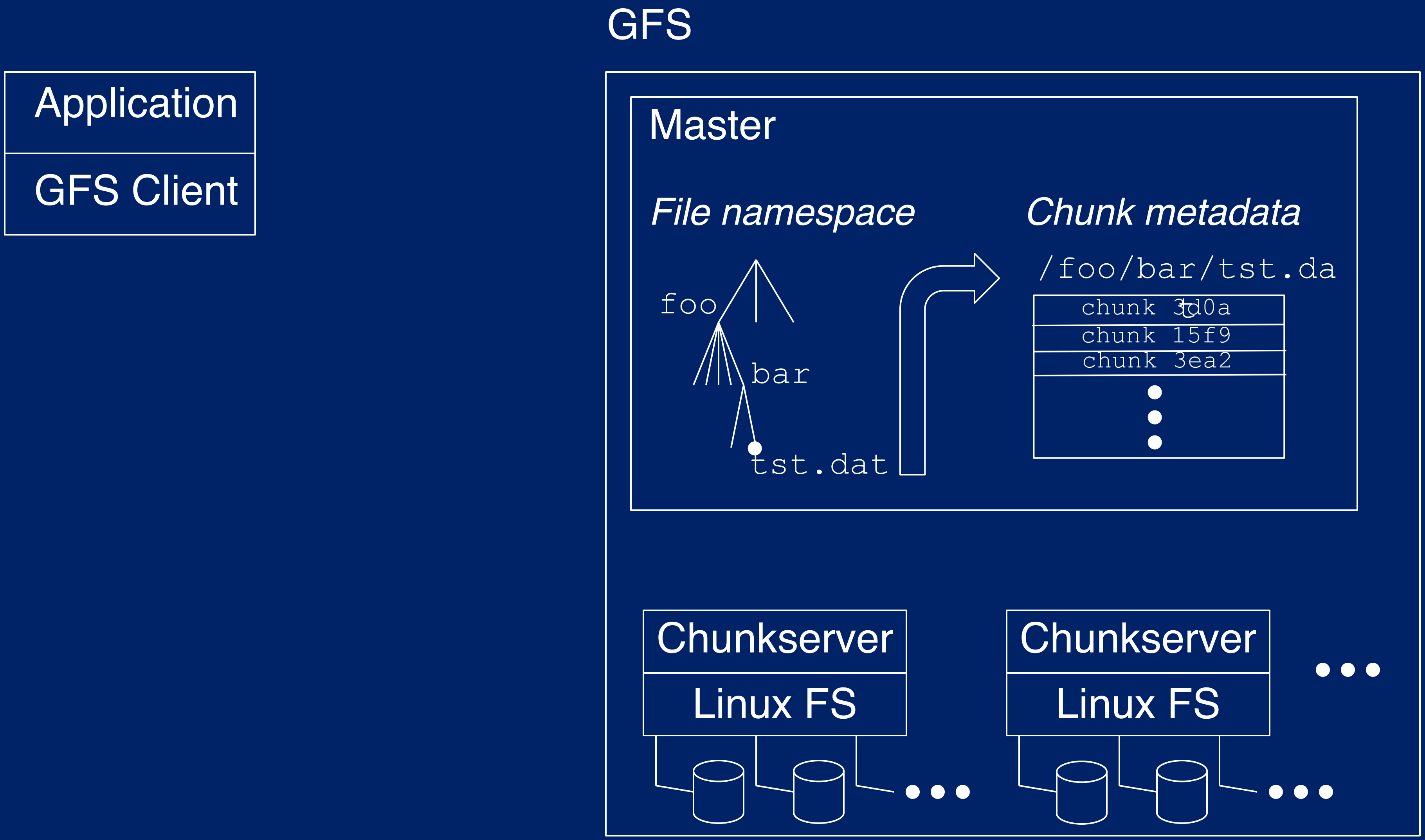
# GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)



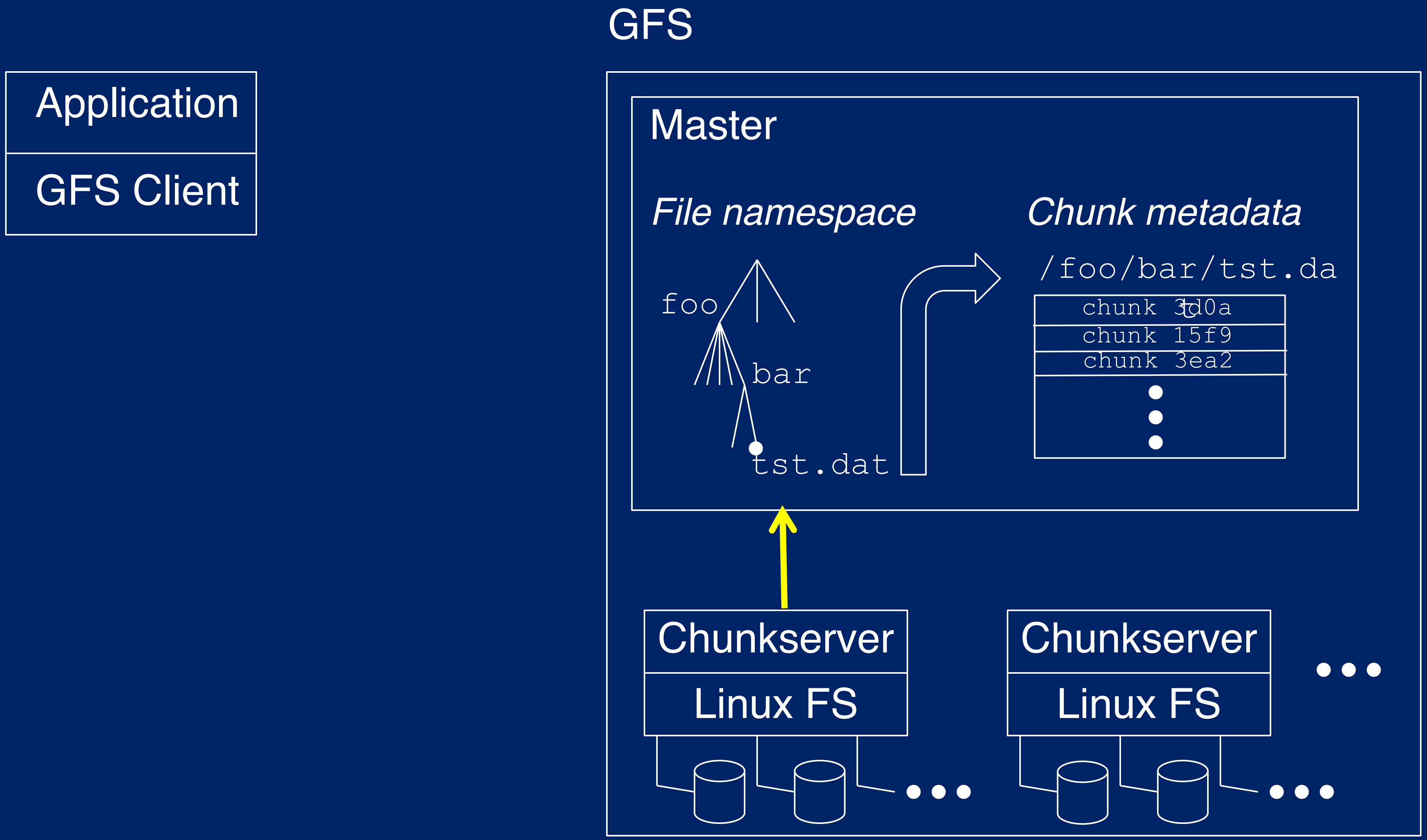
# GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)



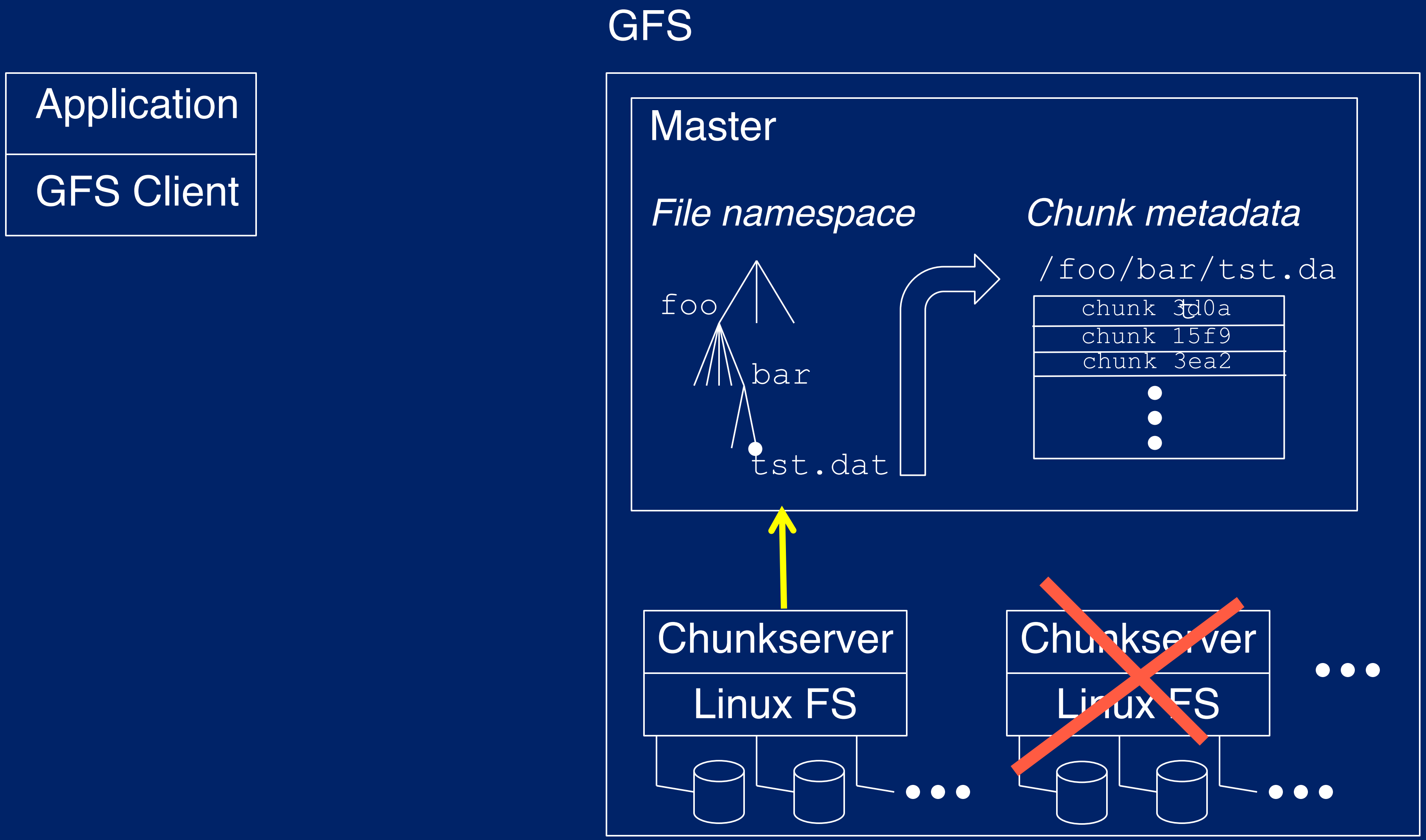
# GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)

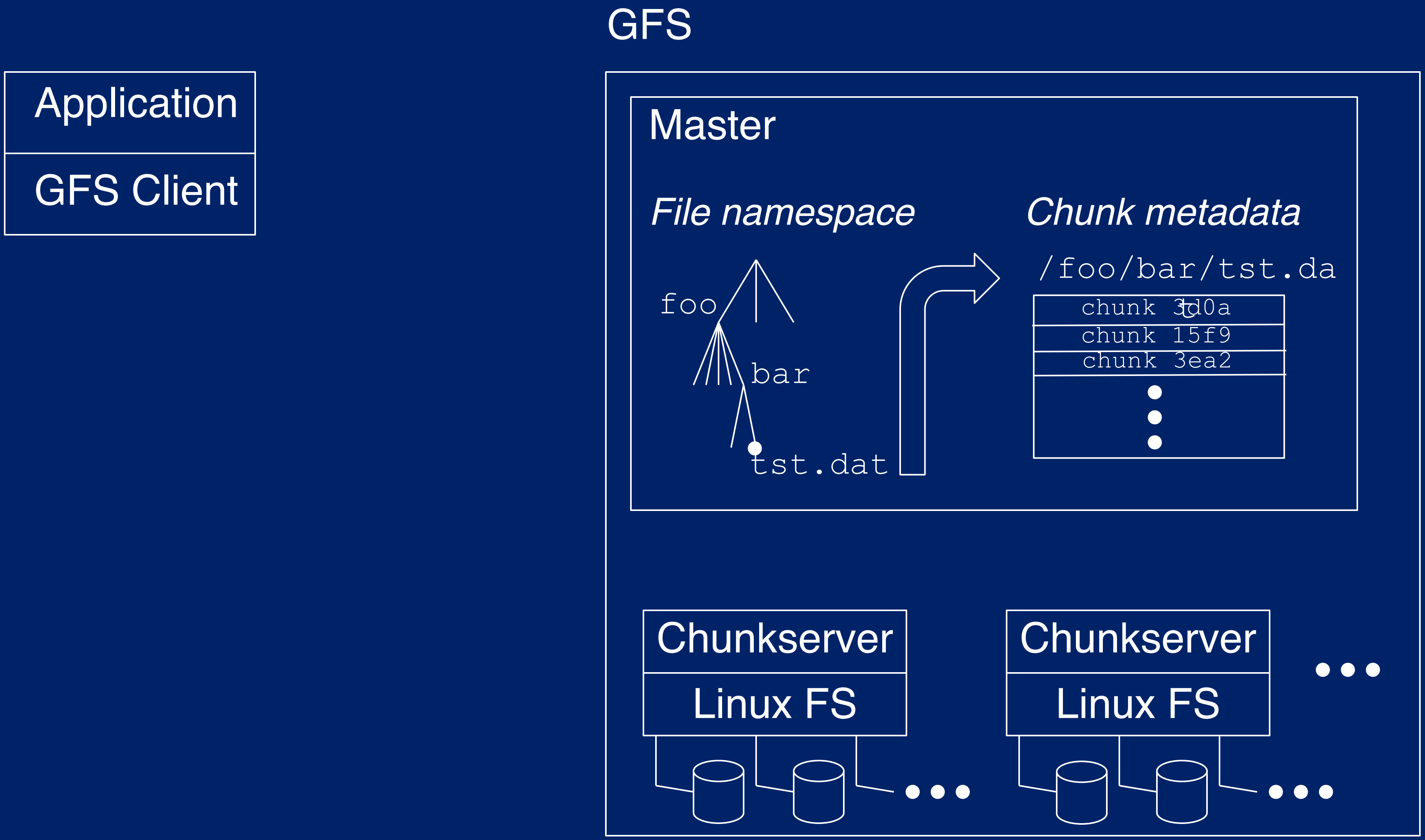


# GFS Architecture

(from Fig.1 of Ghemawat, Gobioff, and Leung)



## GFS Architecture (from Fig.1 of Ghemawat, Gobioff, and Leung)



## BigTable