# Azure File Storage: ‘net use’ the cloud

David Goebel

Microsoft Corporation

# Azure File Storage

## Talk Topics:

1. The features and API surfaces: What
2. The scenarios enabled: Why
3. The design of an SMB server not backed by a conventional file system: How

# What: AFS[1] Fundamental Concepts

- AFS is not the Windows SMB server (srv2.sys) running on Azure nodes.
- AFS is a completely new SMB implementation which uses Azure Tables and Blobs as the backing store.
- AFS leverages the highly available and distributed architecture of Tables and Blobs to imbue those same qualities to the file share.

[1] Azure File Storage not CMU's Andrew File System. I wasn't on the naming committee.

# AFS Features

- SMB 2.1 in preview since last summer.
- SMB 3.0 in progress with encryption & persistent handles.
- Azure storage containers mapped as shares.
- SMB clients work unmodified out of the box.
- As AFS is built on top of Azure Tables and Blobs, the share namespace is coherently accessible through the Azure REST APIs.

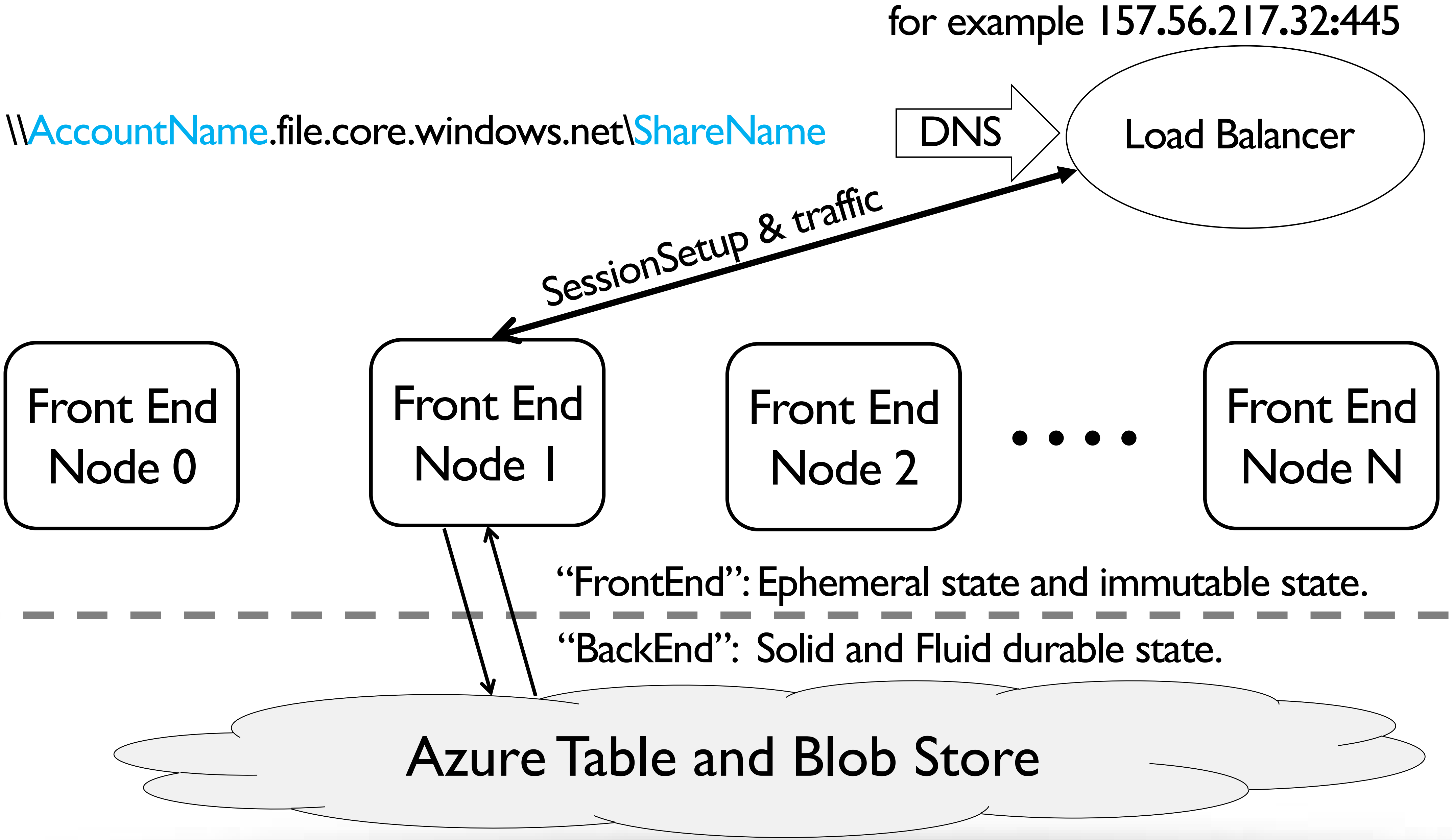
# [MS-SMB2]

- Very different from SMB1.x which had a long and circuitous evolution since the DOS days.
- [MS-SMB2] clearly designed to proxy the NT APIs over the network in a very clean way with compound commands added to reduce chatter.
- Anticipates, though doesn't require, a traditional file system on the other side.
- AFS uses Azure Tables (for metadata) & Blobs instead.

# SMB is a stateful protocol,

## but not all states require expensive distributed transactional semantics

- Some aspects of a file's state are immutable, such as FileId and whether it's a file or a directory.
- Some state is transient, such as open counts, and can be optimized if loss of this state is acceptable in a disaster.
- Some state is also maintained by the client, like CreateGuid, drastically reducing the cost of tracking clients.
- State associated with connection mechanics is ephemeral.



for example 157.56.217.32:445

\\AccountName.file.core.windows.net\ShareName

DNS

Load Balancer

SessionSetup & traffic

Front End Node 0

Front End Node 1

Front End Node 2

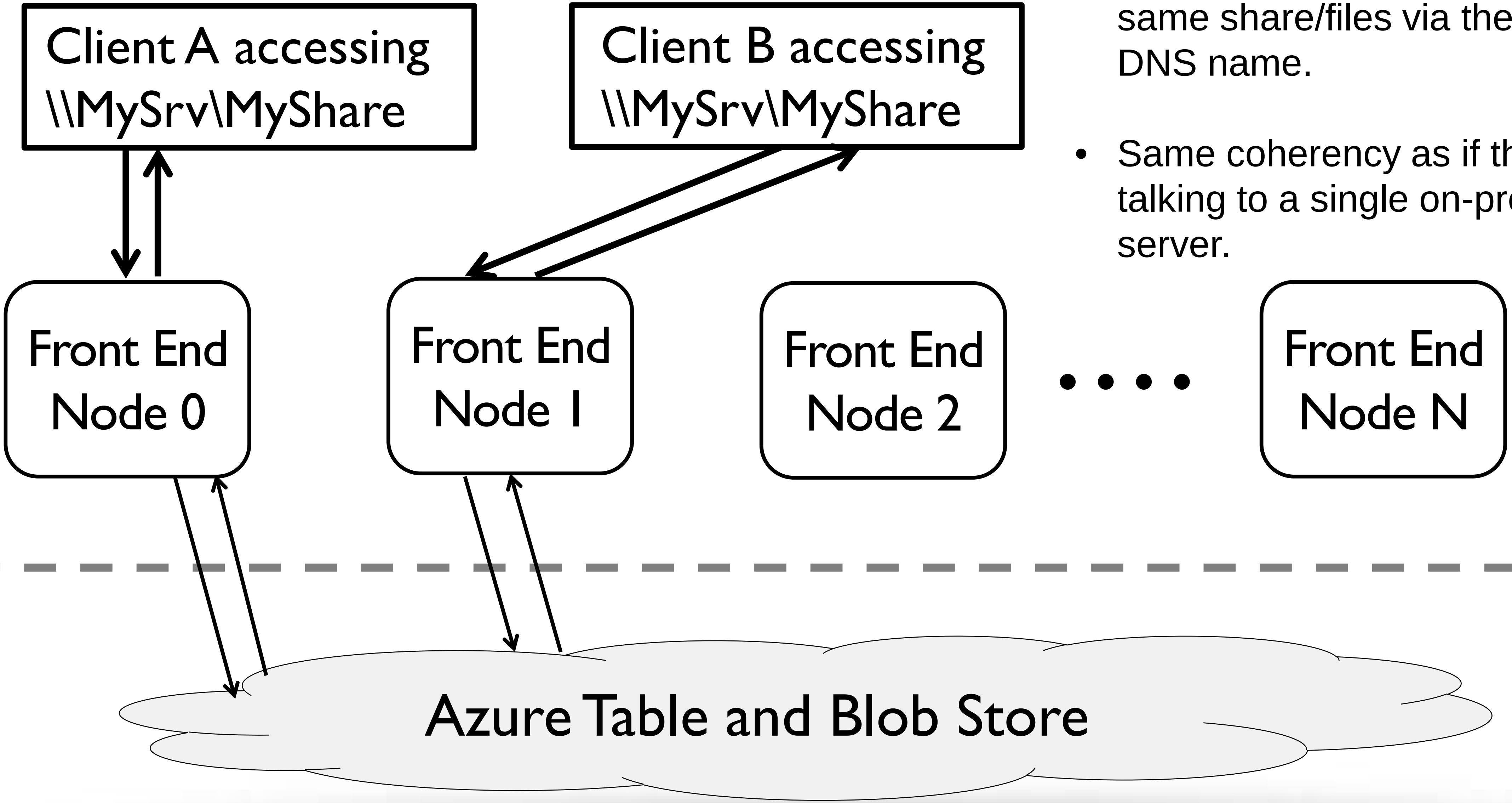
....

Front End Node N

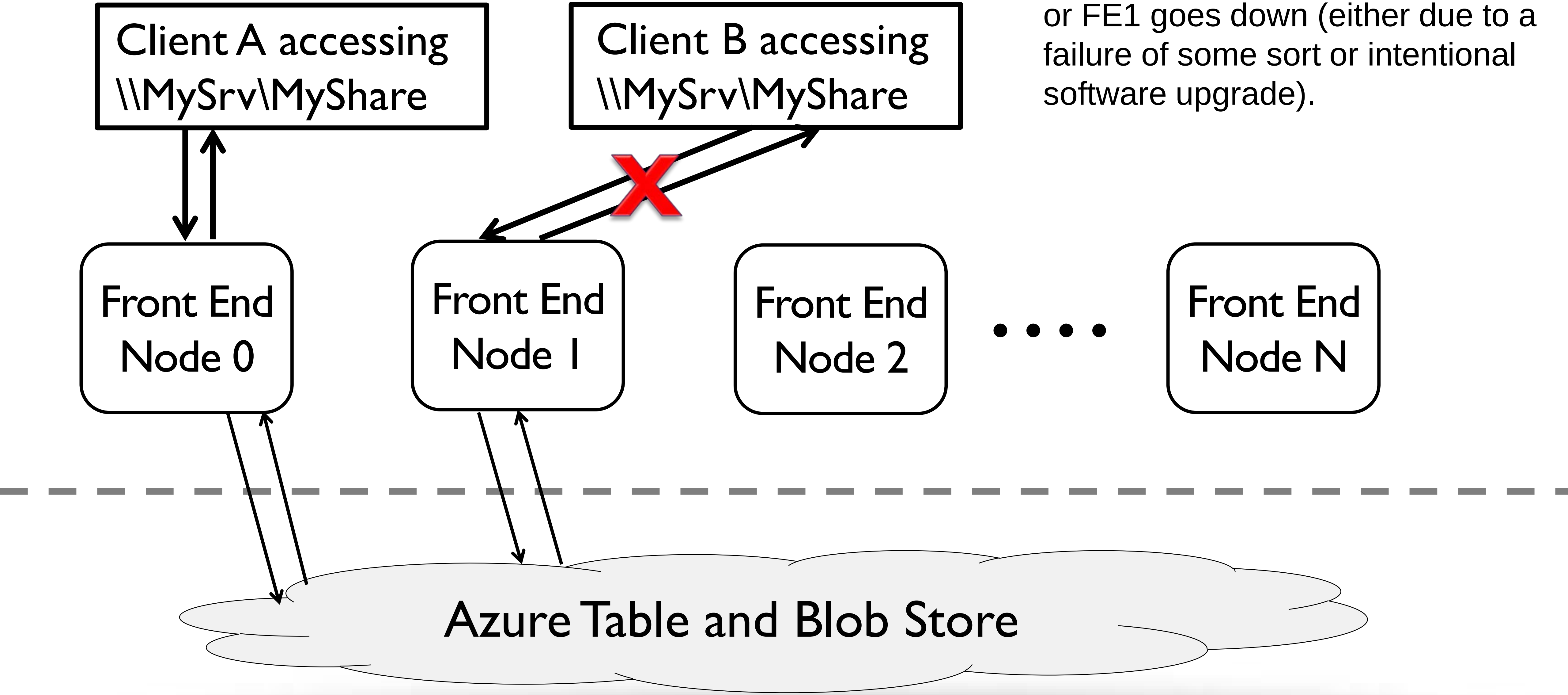
“FrontEnd”: Ephemeral state and immutable state.

“BackEnd”: Solid and Fluid durable state.

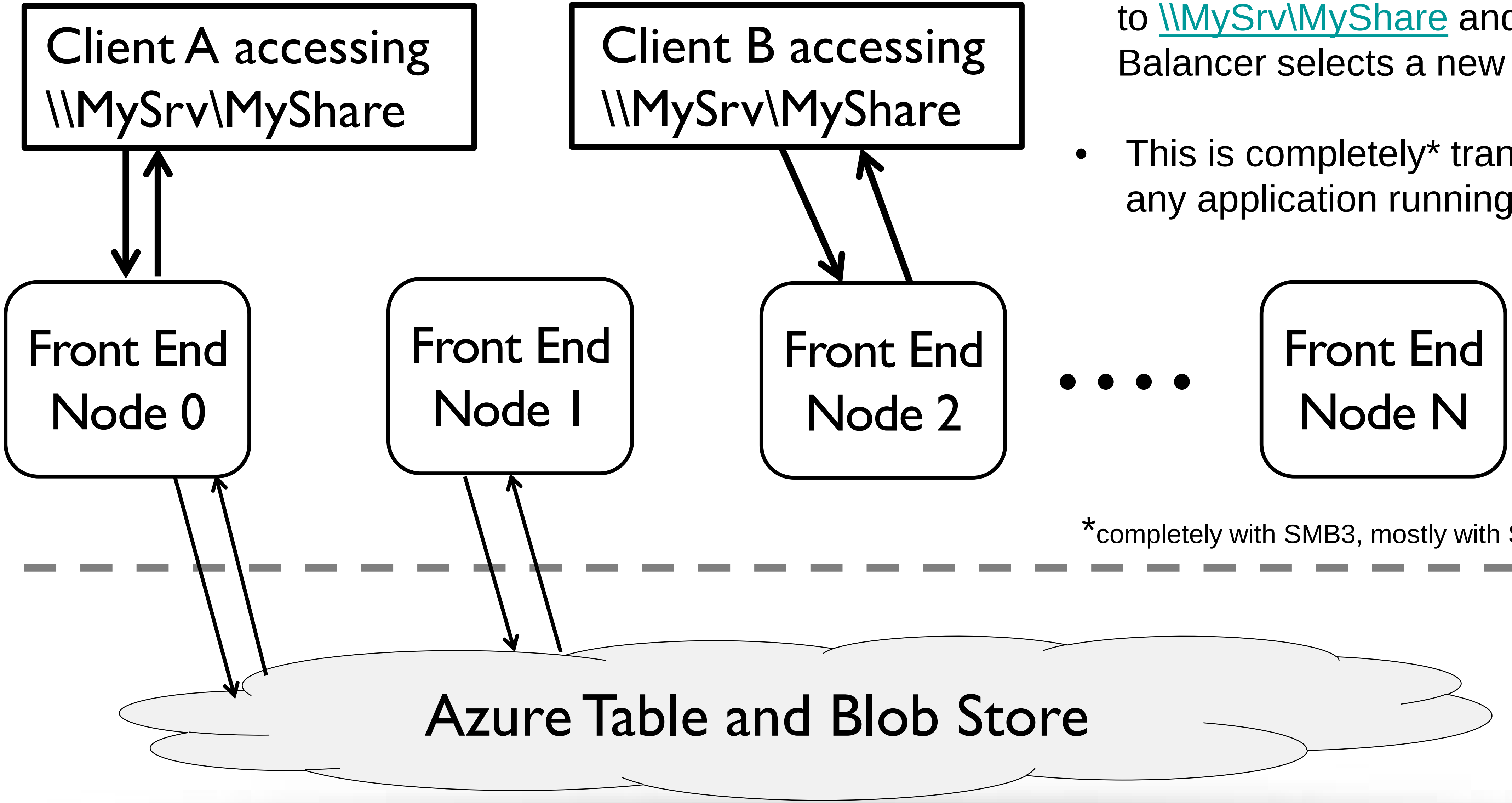
Azure Table and Blob Store


Client A accessing \\MySrv\MyShare
Client B accessing \\MySrv\MyShare
Front End Node 0
Front End Node 1
Front End Node 2
• • • •
Front End Node N
Azure Table and Blob Store

- Clients A & B both accessing the same share/files via the same DNS name.
- Same coherency as if they were talking to a single on-premis server.


Client A accessing \\MySrv\MyShare
Client B accessing \\MySrv\MyShare
Front End Node 0
Front End Node 1
Front End Node 2
• • • •
Front End Node N
Azure Table and Blob Store

- Clients B loses connection to FE1 or FE1 goes down (either due to a failure of some sort or intentional software upgrade).


Client A accessing \\MySrv\MyShare
Client B accessing \\MySrv\MyShare
Front End Node 0
Front End Node 1
Front End Node 2
• • • •
Front End Node N
Azure Table and Blob Store

- Client B automatically reconnects to \\MySrv\MyShare and the Load Balancer selects a new FE.
- This is completely* transparent to any application running on ClientB.

*completely with SMB3, mostly with SMB2.1

# Current AFS Preview State

- SMB 2.1. SMB 3.0 in the works.
- 5TB per share and 1TB per file.
- 1000 8k IOPS per share. 60 MB/sec per share.
- Some NTFS features not supported (see link to list on the Resources slide).
- Shared namespace with REST imposes some limitation on characters and path lengths due to HTTP restrictions.

# DEMO

# Current Linux Support

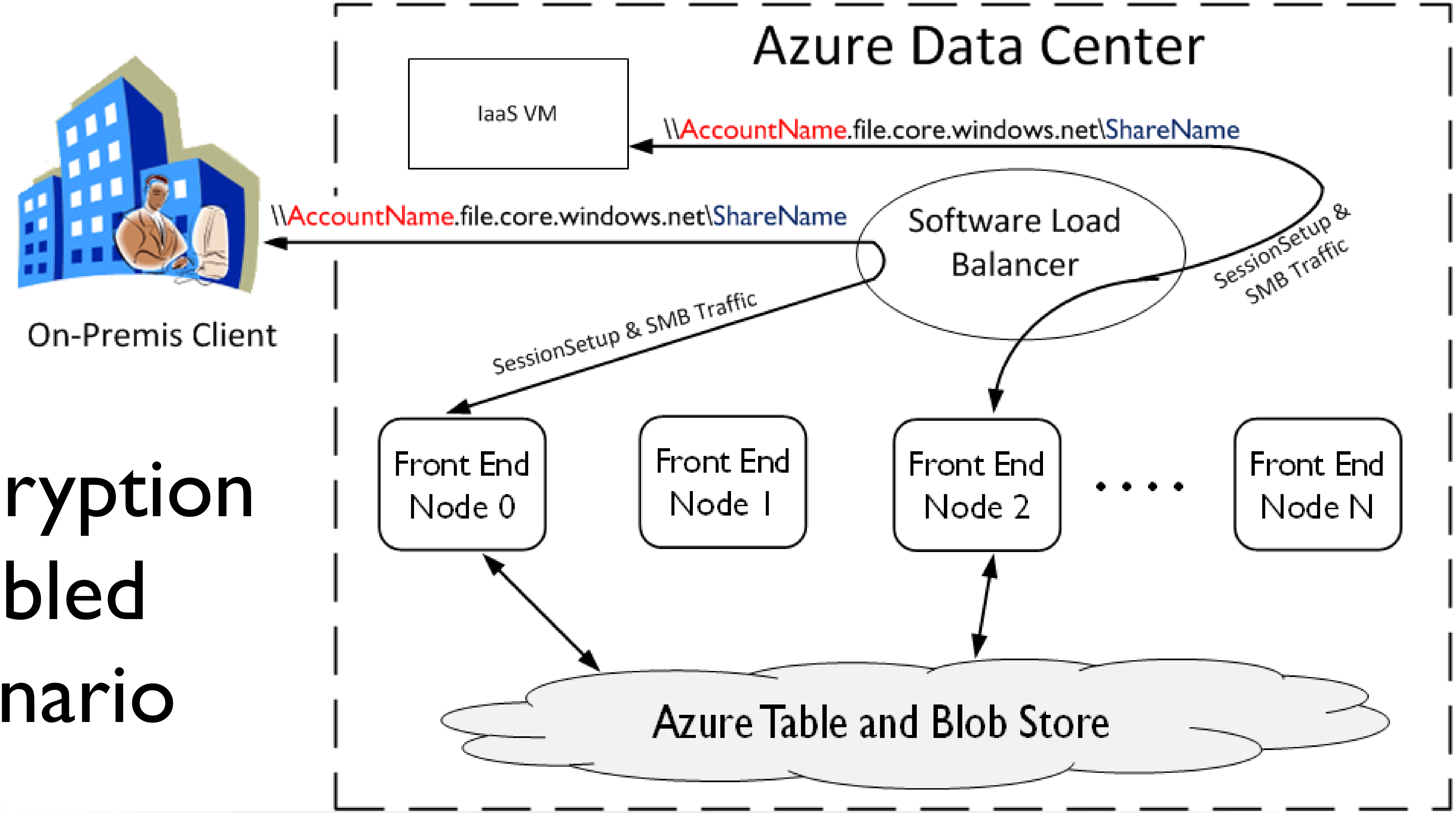
| Linux Distribution | Publisher | Kernel Version | CIFS Version | SMB2.1 |
|---|---|---|---|---|
| **Ubuntu Server 14.04 LTS** | Canonical | 3.16.0-31-generic | 2.03 | Pass |
| **Ubuntu Core 15.04 BETA** | Canonical | 3.19.0-15-generic | 2.06 | Pass |
| **CentOS 7.1** | OpenLogic | 3.10.0-229.1.2.el7.x86_6 | 2.03 | Pass |
| **Open SUSE 13.2** | SUSE | 3.16.6-2-default | 2.03 | Pass |
| **SUSE Linux Enterprise Server 12** | SUSE | 3.12.38-44-default | 2.02 | Pass |
| **SUSE Linux Enterprise Server 12 (Premium Image)** | SUSE | 3.12.38-44-default | 2.02 | Pass |

Note: "Pass" just means the volume mounts and very simple I/O works.

# Why: Scenarios Enabled By AFS

- Existing file I/O API (Win32, CRT, etc.) based applications, i.e. most business applications written over the last 30 years, "just work"®.
- A business can stage existing workloads seamlessly into the cloud without modification to mission critical applications.
- Some minor caveats that will become more minor over time.

# Encryption Enabled Scenario



Azure Data Center
IaaS VM
\\AccountName.file.core.windows.net\ShareName
\\AccountName.file.core.windows.net\ShareName
Software Load Balancer
SessionSetup & SMB Traffic
SessionSetup & SMB Traffic
On-Premis Client
Front End Node 0
Front End Node 1
Front End Node 2
....
Front End Node N
Azure Table and Blob Store

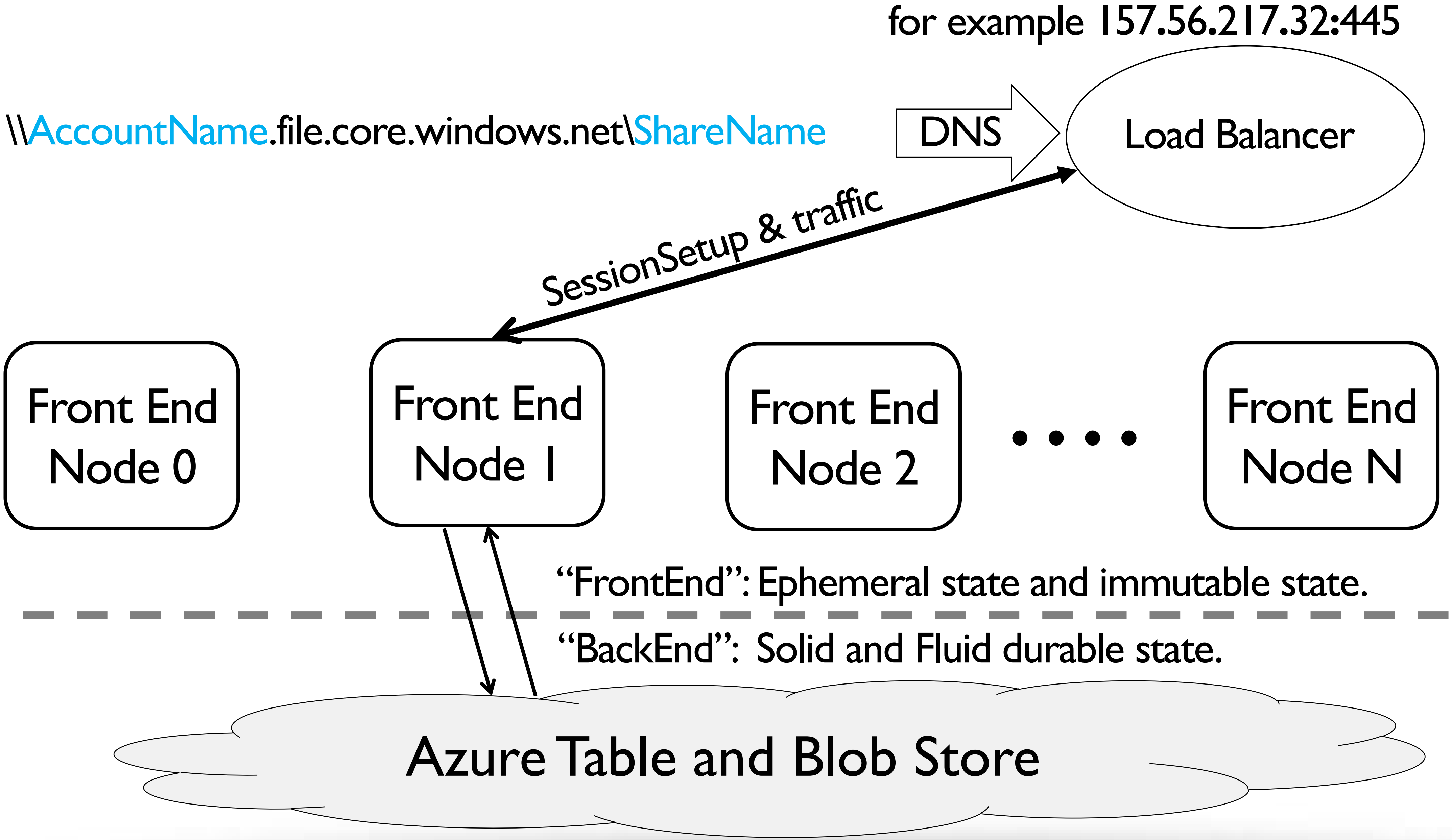
# What about REST?

If you’re a true believer in the benefits of statelessness, SMB and REST access the same data in the same namespace so a gradual application transition without disruption is possible.

- Container operations:
  Create, List, Delete, Get properties, Get/Set metadata
- Directory Operations:
  Create, Delete, Get Properties, List (Same as ListFiles)
- File operations:
  Create, List, Delete, Get/Set properties, Get/Set metadata, Get Contents, Put Ranges, List Ranges

# How: The Durability Game

- A conventional file server treats only actual file data and essential metadata (filesize, timestamps, etc) as needing to be durably committed before an operation is acknowledged to the client (and even then only if opened WriteThrough).
- For true active/active high availability and coherency between FrontEnd nodes, modified state that normally exists only in server memory must be durably committed.



for example 157.56.217.32:445

\\AccountName.file.core.windows.net\ShareName

DNS

Load Balancer

SessionSetup & traffic

Front End Node 0

Front End Node 1

Front End Node 2

. . . .

Front End Node N

“FrontEnd”: Ephemeral state and immutable state.

“BackEnd”: Solid and Fluid durable state.

Azure Table and Blob Store

# Examples of state tiering

- Ephemeral state: SMB2_FILEID.Volatile, credits, tcp socket details.
- Immutable state: 64bit actual FileId, IsDirectory
- Solid durable state: SMB2_FILEID.Persistent, SessionId
- Fluid durable state: Open counts, file names, file size, lease levels and many more. This is the largest group of states.

"Solid" here meaning the state is generated by AFS and not generally changeable by normal actions of the client/application while "Fluid" is fully changeable by File APIs.

# Example: Durable Handle Reconnect

- Intended for network hiccups as it assumes all state is still valid on the server.
- On AFS this state is durably persisted on our BackEnd so we're able to 'stretch' durable handles to recover from FrontEnd AFS failures (planned or otherwise) since it's transparent to the client.
- This is important as we're continually updating AFS code requiring AFS service restarts.

# Example: Persistent Handles

- Unlike durable handles, actually intended to support Transparent Failover when the server dies.
- Leverages state on the client for replay detection so that 'once only' operations are only executed once.
- More details about create requests durably committed.
- With durable handles SMB 2.1 protocol compliance required us to artificially limit our capability. With Persistent Handles we have seamless Transparent Failover.

# Resources:

- Getting started blog with many useful links: http://blogs.msdn.com/b/windowsazurestorage/archive/2014/05/12/introducing-microsoft-azure-file-service.aspx
- NTFS features currently not supported: https://msdn.microsoft.com/en-us/library/azure/dn744326.aspx
- Naming restrictions for REST compatibility: https://msdn.microsoft.com/library/azure/dn167011.aspx