MIND: Metadata Is Not Data

Virtually every aspect is different

We need to treat them as different things

Curtis Anderson
April 3, 2019

MIND: Metadata Is Not Data

panasas
Beyond Storage

GNU: acronym for “Gnu is Not Unix”
MIND: acronym for “Metadata Is Not Data”

The first one is real, the second is one I made up to as a mnemonic

Metadata is an afterthought in most storage systems

It is overhead that is really needed to manage the real value, the data

- **Eg: admins will turn off ATIME updates to increase performance**

“Archivists”: specialized people at the Library of Congress who care about metadata, and are responsible for holding and preserving precious data. They use the term "provenance“.

Now, metadata is a hot topic for the rest of us

The sheer scale of metadata in HPC storage, billions of things files and directories

The need to search for patterns in storage metadata

Virtually Every Aspect is Different

panasas
Beyond Storage

		Data	Metadata		
			XAttrs	Directories	Inodes
Meaning	**Is it Purely Payload?**	Uninterpreted and arbitrary	May be interpreted	Interpreted and acted upon	
Size	**Fixed/Variable**	Widely variable, typically large	Variable, Small	Variable, Very Small	Fixed
	Maximum	0 to 2^{64-1}	0 to 64KB	1 to 255B	512B
Access	**Pattern**	Mostly Sequential	Mostly Random	Sometimes Sequential	Random
	Minimum Granularity	Byte	Record		
ACID Rules	**Atomicity**	Read-to-write, but sometimes page-level atomicity only (eg: ext4)	Guaranteed		
	Consistency				
	Isolation				
	Durability	None unless flagged or flushed			
	Locking Options	Byte-range or whole-file	None		
	Referential Integrity	None	None	POSIX Rules	
Locality	**“Nearness”?**	Very high priority	Low priority	Moderate priority	Very low priority

Why have we treated them the same?

panasas
Beyond Storage

Metadata has always been part of the F/S, so filesystem APIs were used

POSIX APIs are focused on data access, metadata access is a byproduct of that

- **Eg: there are read(), readv(), pread(), preadv(), and preadv2()**

POSIX metadata APIs are not optimized for quick or powerful access

- **Eg: there's no "bulk stat()" or built-in "tree walk",**

The DMAPI (now XDSM) spec includes efficient bulk access to metadata

- **Even that is only a bulk load method, no filters or transformations are supported**

Metadata is stored "inside" a F/S, using the same techniques as used for data

Metadata has "locality" to the data, optimized for fast subsequent access to the data

Parallel F/S's Like PanFS Always Separated Metadata

panasas
Beyond Storage

Using the right tools for metadata vs. data

Director Nodes interpret and modify POSIX metadata

Storage Nodes hold data (*and uninterpreted metadata*)

Director S/W & H/W optimized for low-latency

Metadata is transactional: small items, atomic updates

Cache and logging algorithms optimized for that workload

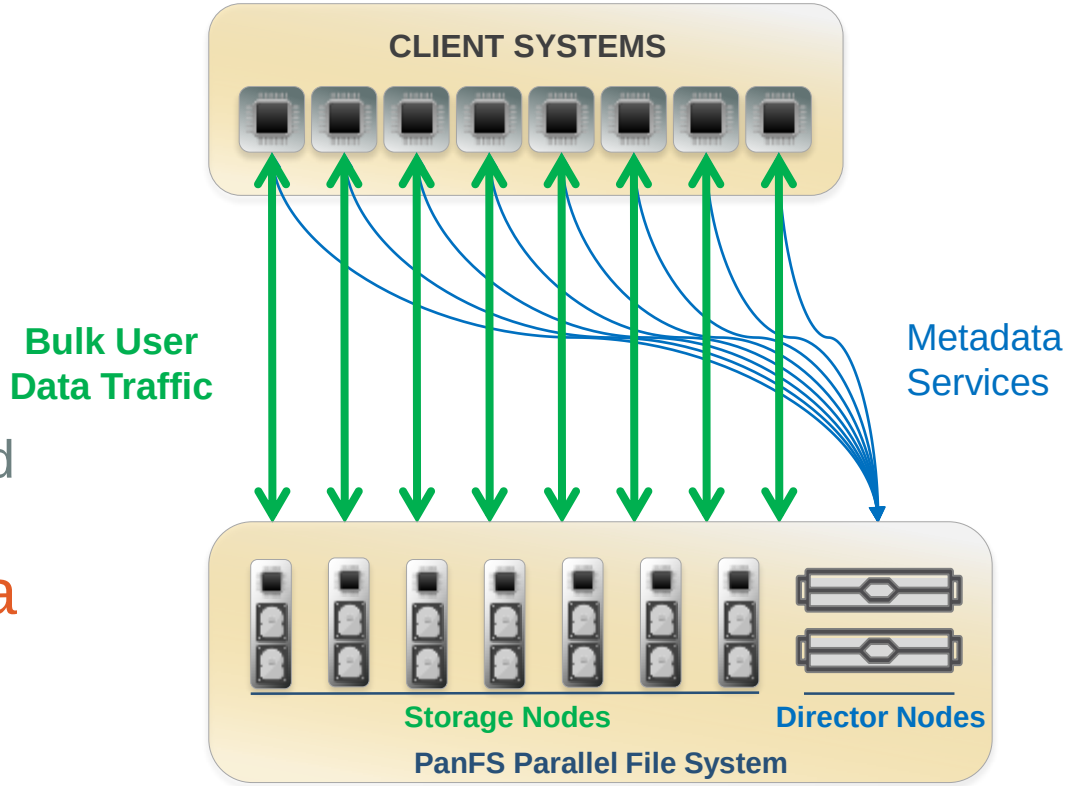
Continue path of separating metadata from data

Metadata access patterns stress Storage Nodes

- **Storage Nodes only do Objects, metadata access same as data**

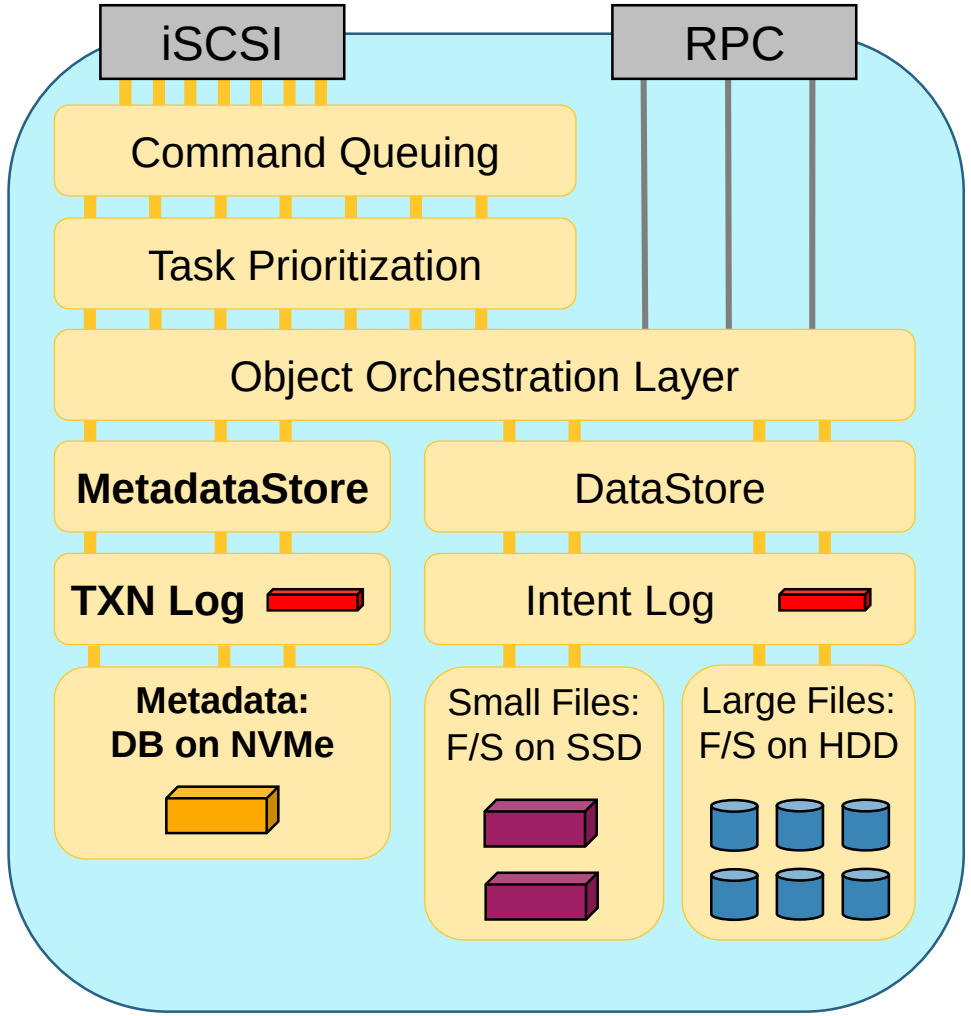
Share the hardware, but change the s/w access models

High Performance Panasas Storage



ActiveStor Ultra Upgrades Metadata Access Path

Use Case	S/W	H/W	Characteristics
Cache	Buffer Cache	DRAM	Typical caching of read-only data and metadata.
Transaction Logs	Intent Log	NVDIMM	Very fast and power-safe transaction completion.
Metadata	**Database or KVS**	**NVMe SSD**	**Fast atomic transactions, consistent performance, optimized caching, and intelligent queries.**
Small Files	Filesystem	SATA SSD	Small files on cost-effective zero-seek-time SATA SSDs.
Large Files	Filesystem	SATA HDD	HDDs are good at delivering bandwidth if they have large transfers to work on.




panasas
Beyond Storage
Thank You!