The Storage-Centric Stack: Decoupling Compute, Standardizing on S3, and Lowering TCO

Executive Summary

Modern data architecture is increasingly centered around flexible storage, with object stores like Amazon S3 and compatible systems emerging as the backbone of analytics infrastructure. **Storage is becoming the core infrastructure layer**, decoupled from compute, and shared across diverse tools and engines. A growing body of evidence indicates that most organizations are standardizing on S3-compatible object storage APIs as a lingua franca for data, enabling multi-cloud and hybrid deployments. The Amazon S3 API has effectively become a de facto industry standard for large-scale storage: it is widely adopted by cloud providers and on-premises vendors alike . Even independent storage vendors report that the ecosystem of applications using the S3 API has exploded beyond AWS itself . This standardization means teams can write applications against a common object storage interface and run them on AWS, other public clouds, or on-premises with minimal changes, facilitating portability.

Hand in hand with storage-centric design, **open table formats like Apache Iceberg are on the rise**, challenging the need to lock data into proprietary database storage engines. Iceberg provides a layer of schema and ACID transaction management atop files in object storage (e.g. Parquet on S3), and many in the industry see it as a future default. Major enterprises such as Netflix, Apple, and Google have been early adopters of Iceberg in various projects , and vendors across the spectrum are adding support. Snowflake's platform now allows data to be stored in Iceberg tables on a customer's S3 buckets, Databricks has open-sourced Delta Lake and acquired an Iceberg-focused company, and even AWS Athena and Google BigQuery have introduced Iceberg support . This broad support signals momentum toward Iceberg as an *open standard for table storage*, though it remains an emerging technology. Analysts note that while Iceberg is **gaining adoption and could become the de facto standard**, the market is still in early stages – many organizations are only evaluating these formats and have yet to roll them out company-wide . In 2024 it sits at the "peak of inflated expectations" on the hype cycle, indicating enthusiasm is high but full production maturity is still developing .

The way data is queried is also shifting in favor of direct analysis on file-based data in object storage. An increasing percentage of analytic queries are executed *directly on Parquet files in S3* (or other cloud object stores) using SQL engines like Trino, Presto, Spark, or cloud query services. In fact, in an industry survey, 65% of IT/data professionals reported that **over half of their analytics workloads are running on a "data lakehouse"** – i.e. using object storage with

SQL engines rather than loading data into a traditional database . This suggests a majority of organizations have crossed a tipping point where more than 50% of their queries leverage a storage-centric approach. Amazon's own metrics underscore this trend: in 2024, Amazon Redshift users collectively scanned **over 77 exabytes of data directly from data lake storage** (S3) via Redshift Spectrum and related features . This enormous volume of S3-based querying illustrates how commonplace it has become to query data in situ on the data lake. Leading cloud data warehouses (Snowflake, BigQuery, Redshift, etc.) now all enable external tables or query federation to data in Parquet/ORC on cloud storage, further blurring the line – a recognition that users want the flexibility to analyze data without always importing it into a database. The **"lakehouse" architecture – combining cloud storage with open formats and multiple query engines – is increasingly the default** for new analytics platforms, often supplementing or even replacing traditional warehouses.

Crucially, organizations are **procuring and managing storage and compute resources independently** to maximize flexibility and cost efficiency. In contrast to legacy "monolithic" database systems where storage and compute scale together, today's best practices dictate a decoupling of these layers . Cloud object storage (like S3) can scale cheaply and indefinitely, while various compute clusters (SQL query engines, Spark, etc.) can be spun up as needed on top of the same data. This separation is now seen as an architectural ideal: it lets teams right-size compute independently of storage growth, avoiding over-provisioning and allowing each to be optimized (and budgeted) on its own curve . The approach was popularized by cloud data warehouses such as Snowflake, and has now become **"the de facto standard" for modern distributed databases and data platforms** . New distributed SQL systems (Amazon Aurora, Alibaba PolarDB, etc.) and analytics platforms all emphasize a decoupled storage-compute design . The result is that enterprises increasingly purchase bulk storage (often in the form of S3 or S3-compatible services) separately from compute engines. They might, for example, use a central object store for all data, and procure computing (cloud VMs, Kubernetes, analytic DB services) on-demand to run workloads against that data. This independent procurement changes organizational patterns: storage may be managed by a central data lake team, while compute is used in a self-service model by various analytics teams. It also encourages a **multi-engine ecosystem** (SQL engines, ML tools, etc. all sharing the same storage), rather than forcing all workloads into one monolithic data warehouse.

The dominance of S3 as an interface has additionally enabled **multi-cloud interoperability and "lift-and-shift" flexibility**. A rich ecosystem of S3-compatible storage offerings exists across clouds and on-premises – from Microsoft and Google's object stores (which can be accessed via compatibility layers or third-party gateways) to on-prem systems like MinIO, Cloudian, and others . Applications written against the S3 API can thus be ported with minimal friction. For example, an application using an S3-compatible API (such as MinIO's client library) can **seamlessly switch its backend from AWS S3 to Azure Blob Storage or to an on-prem MinIO server without code changes**, simply by pointing to the new endpoint . This interoperability is a major advantage of standardizing on the S3 API: it mitigates vendor lock-in and eases cloud repatriation or multi-cloud distribution of data. Organizations have taken advantage of this by developing **cloud-agnostic data architectures** – for instance, keeping a copy of critical data in an on-prem S3-compatible store to avoid excessive egress costs, or

moving workloads from AWS to alternative clouds using the same S3-based applications. The consistency of the API means teams don't have to rewrite their data access logic when migrating storage. However, it's worth noting that full lift-and-shift between clouds still entails moving large volumes of data (which can be slow and costly), but from an application integration standpoint, S3 compatibility ensures the *software* will work across environments. In short, S3 has become a lingua franca not just within AWS, but **across cloud and edge, enabling true portability of the storage layer**.

Finally, one of the driving forces behind this storage-centric paradigm is **cost and total cost of ownership (TCO)**. Evidence strongly suggests that a "storage-centric" stack – using commodity object storage plus modular compute – can deliver markedly lower TCO than a traditional "database-centric" stack where an expensive database engine owns all storage. In the Dremio 2024 State of the Lakehouse survey, over half of respondents (56%) expected **cost savings greater than 50%** by adopting a lakehouse architecture over legacy warehouses . Nearly 30% of large enterprises even anticipated **savings above 75%** with this shift . These estimates reflect the elimination of hefty data warehouse license costs and the cheaper scaling of storage. Object stores like S3 offer storage at a fraction of the price of proprietary database storage, especially for cold or infrequently accessed data, and they avoid the redundant copies often needed in warehousing. There are real-world cases corroborating massive savings. For example, 37Signals, Twitter (X), and Ahrefs each publicly shared that **repatriating large analytics workloads from cloud warehouses to private object storage** yielded on the order of 60% cost reduction in infrastructure spend . One company moved 500 PB from AWS to an S3-compatible private cloud and improved its **gross margin by 2–3%** for a multi-billion-dollar business – a striking business impact. Similarly, analysis by object storage vendor MinIO showed that managing 100 PB in AWS S3 could cost about $33M per year, whereas using MinIO with commodity hardware in a colo might cost around $9.5M in year one (including hardware purchase) and under $5M annually thereafter . Over five years, that's $30M vs $166M – an **82% reduction in TCO** for the storage layer . While this specific scenario is vendor-provided, it aligns with the survey sentiment that storage-centric architectures can be dramatically more cost-efficient at scale. In summary, the financial argument for a storage-centric (or lakehouse) stack is compelling: lower storage costs, pay-as-you-go compute, and avoidance of expensive data warehouse appliances or licenses lead to a **significantly lower TCO**, especially as data volumes explode.

In conclusion, the industry trend is clear – **data storage (particularly object storage) is becoming the foundational infrastructure**, with compute increasingly seen as an interchangeable, elastic layer on top of the data. Teams are converging on S3-compatible object stores as the standard repository for all data, and technologies like Apache Iceberg are bringing database-like features to these stores, reducing the need for specialized databases to manage analytic data. Most analytical queries can now be executed efficiently directly against data in S3/Parquet using a variety of engines, and organizations benefit from the flexibility of scaling storage and compute separately. Multi-cloud strategies are enabled by the ubiquity of the S3 API, and companies adopting these storage-centric architectures are often realizing major cost savings. This represents a paradigm shift from the database-centric world of the past: instead of data being captive inside each database, it lives in a central storage infrastructure and

databases or query engines come to the data. **Storage has effectively become the platform**. The following sections provide a detailed analysis of each facet of this trend, as well as counterpoints and challenges that have emerged.

Standardizing on S3-Compatible Object Storage APIs

One of the clearest signs that "storage becomes the infrastructure" is the widespread adoption of **S3-compatible object storage APIs** as a standard interface across organizations. Amazon's Simple Storage Service (S3) popularized a RESTful object storage API that has since become *the lingua franca* for modern storage systems. Today, **most teams are indeed standardizing on S3-compatible APIs** for object storage access, whether they are using AWS or not. This is evident in both public cloud and on-premises contexts. A 2023 industry overview by Cloudian (an on-prem object storage vendor) noted that the S3 API "is now the most common way in which data is stored, managed, and retrieved by object stores" and has become "the de facto standard for object storage, employed by vendors and cloud providers industry-wide." . In practice, this means that if an application is written to "speak" the S3 API, it can work with a huge range of storage backends. Cloudian's report highlights that the S3 API is now *widely supported among on-prem storage platforms*, not just by AWS – NetApp, Dell/EMC, Pure Storage, MinIO, VAST Data, Scality and others have all implemented S3-compatible interfaces on their storage products . Legacy file and block storage vendors have effectively acknowledged S3 as the new standard by making their systems speak this language.

From the public cloud perspective, while AWS's own S3 service doesn't need introduction, other clouds have also embraced compatibility (directly or via tools). Google Cloud Storage and Azure Blob Storage each have different native APIs, but there are gateways and configurations (for example, Google's Storage Transfer Service or third-party middleware) that allow S3 API access to those stores. More directly, many companies opt to **deploy open-source S3-compatible storage in the cloud or at the edge** – for instance running MinIO on Azure or GCP to provide an S3 endpoint backed by cloud disks. MinIO, an open-source object store known for S3 compatibility, has become especially prominent in multi-cloud setups. In a recent interview, the CEO of MinIO claimed that *MinIO's own adoption has surpassed AWS S3's adoption* in terms of instances deployed, precisely because so many organizations use MinIO to enable S3-style storage outside of AWS . While that claim is difficult to verify, it underlines the pervasiveness of the S3 API. He argued that **MinIO helped make the S3 API an industry standard beyond Amazon** – "all these AI applications…are largely built on MinIO's API," he noted, pointing out that many modern data stacks (even vector databases for AI) use S3-compatible storage under the hood . This might be somewhat self-serving, but independent analysts agree on the core point: **the S3 API is essentially the universal dialect of cloud storage today**.

The reasons behind this standardization are manifold. First, the S3 API offers *cloud-native features* that older storage protocols (like NFS or SMB) lack – such as built-in multi-tenancy, remote access over HTTP, and scalable object namespace handling . It was designed for internet-era scale and fits modern data needs. Second, by standardizing on S3, organizations attain *portability*. Applications coded for S3 can run anywhere a compatible store exists,

avoiding vendor lock-in. A tech blog on multi-cloud APIs gives a concrete example: if you write your app to use S3 calls (for example, via MinIO's library), you can **swap the backend from AWS S3 to Azure Blob or to an on-prem MinIO cluster without changing your code** . That interoperability has huge appeal in multi-cloud strategies – it provides leverage against any single provider. Third, a rich ecosystem of tools (backup software, data lake analytics engines, etc.) has grown up around S3. Many open-source and commercial data tools *assume* an S3-compatible storage. For instance, Hadoop, Spark, Presto, and others include connectors for S3 out-of-the-box. That ubiquity reinforces itself: if your storage speaks S3, it can immediately plug into countless frameworks.

It's important to clarify that "standardizing on S3 API" doesn't necessarily mean everyone uses **AWS S3 service** – rather, they use the API standard. As noted, there are dozens of S3-compatible storage solutions. Cloudian's review emphasizes that some implementations vary in completeness – S3 has added many extensions over the years, and not all vendors support 100% of the API . Compliance ranges from ~50% to >90% of the API across products . This sometimes leads to minor incompatibilities if an application uses a less-common API call. Nonetheless, the core operations (PUT, GET objects, list buckets, etc.) are universally implemented, and those cover the vast majority of use cases. Thus, **most teams find it viable to make S3 the default interface** for their data storage, even if behind the scenes they might be on Azure or on a private cloud. The S3 API's dominance is so pronounced that analysts call it *the only storage "language" born in the internet era* and inherently suited to today's needs . Other protocols (like NFS, SMB) were built for LAN environments decades ago and don't scale or integrate as easily over the internet. As a result, new applications – particularly those involving big data, AI data lakes, microservices, etc. – gravitate toward using object storage via S3 calls rather than setting up traditional file shares.

In practical terms, standardizing on S3-compatible storage means development teams can **treat storage as an infrastructure service** similar to compute. They write to an API without worrying about the hardware or vendor providing it. Many enterprises have built internal platforms where any team can request an S3 bucket to store data for a project, and behind the scenes it could be AWS, Google, or an internal object store – the difference is abstracted. This has also led to the idea of a *"data lake" on S3* becoming a common element of data architecture: all raw and processed data lands in an S3-based repository, serving as the single source of truth for the organization's data. Individual applications (or databases) might pull subsets of data from that lake as needed, but the authoritative copy lives in the object store. This inversion – where the durable storage is separate from the compute engines – relies fundamentally on having a standardized API like S3.

To summarize, **yes – most teams today do standardize on S3-compatible object APIs,** given the overwhelming industry support and benefits. S3 compatibility has become *table stakes* for storage products and a default requirement in data architectures, enabling the flexibility and portability that modern workloads demand. The next sections will explore how this storage-centric approach, built on S3, is influencing other layers of the stack, from table formats to query engines and beyond.

Rise of Apache Iceberg Tables vs. Database-Native Storage

As organizations shift toward a storage-centric paradigm using object stores, a natural question arises: how to manage *structured* data (tables) on this storage in a way that offers the reliability and usability of traditional databases? This is where **Apache Iceberg** and similar table formats come into play. The question posed is whether Iceberg tables are becoming the default choice over “DB-native” storage – meaning the proprietary storage layers inside databases or data warehouses. The emerging evidence suggests that **Apache Iceberg is rapidly gaining traction as a default open table format** for analytics, although it has not completely eclipsed native DB storage yet (and may coexist with it for some time).

Apache Iceberg is an open-source table format that sits on top of files in object storage (e.g. Parquet files on S3), adding a rich layer of metadata, schema evolution, partitioning, and ACID transaction support. In essence, it brings warehouse-like table management to the data lake. Instead of loading data into a specific database’s internal storage format, one can keep the data in Parquet files on S3 and define an Iceberg table on it, which multiple engines can read consistently. This concept has resonated strongly in the industry’s push toward open data architectures. **Iceberg’s adoption has accelerated among large tech companies and vendors alike.** A participant at an Iceberg summit noted that some of the world’s largest companies – Netflix (which originated Iceberg), Apple, Google, among others – are actively using Iceberg in production or piloting it in various projects . These companies have diverse needs (from streaming data to machine learning to traditional analytics), and Iceberg’s ability to handle large petabyte-scale tables with ACID guarantees appeals to them. Each might value different features – one might care about Iceberg’s **time travel and versioned data** for data science, another about avoiding the high cost of reloading data into warehouses – but collectively these benefits are “pushing Iceberg forward because it basically makes sense for everyone,” as the summit attendee observed . In other words, Iceberg hits a sweet spot of attributes (open format, cost-effective storage, query engine agnosticism, and features like schema flexibility) that address pain points across the board.

One measure of Iceberg’s momentum is the extent to which **commercial data platform vendors have embraced it.** Within the last 1-2 years, nearly every major analytics platform has announced support for Iceberg:

- **Snowflake** – long a closed cloud data warehouse – surprised the industry by adding support for Iceberg tables. Snowflake allows customers to create external tables referencing Iceberg data on S3, and in 2023 it announced “Project Polaris,” an open source Iceberg metastore implementation, signaling commitment to Iceberg’s ecosystem . Essentially, Snowflake acknowledged that some data will live outside its proprietary storage, and it wants to participate in that open ecosystem rather than be sidelined.

- **Databricks** – champion of the “lakehouse” concept with its Delta Lake format – initially was a competitor to Iceberg. However, in 2023 Databricks acquired Tabular (a company

dedicated to Iceberg) and has indicated some level of support or integration going forward . Databricks still pushes its own Delta format, but the acquisition suggests they foresee Iceberg's popularity and want a hand in its direction (or at least to ensure interoperability). The exact strategy is unclear, but it's noteworthy that even the Delta Lake vendor felt compelled to invest in Iceberg's future.

- **AWS** – has strongly backed Iceberg in its analytics services. Amazon Athena (serverless SQL query service) and AWS Glue Data Catalog both added official support for Apache Iceberg. According to community commentary, AWS seems to be "strongly preferencing Iceberg over Delta/Hudi" for its customers in services like Glue and Athena . This aligns with AWS's general support for open, vendor-neutral technologies (Delta Lake was initially tied to Databricks).

- **Google Cloud** – introduced BigLake and support for open table formats. BigQuery can query external tables in Iceberg (and Delta), and Google Cloud Data Catalog can interface with Iceberg metadata. Google also contributed to the Iceberg project.

- **Microsoft** – Azure Synapse Analytics and the new Microsoft Fabric analytics product have embraced open table formats as well. Notably, Microsoft Fabric's OneLake supports creating tables in Delta format (Databricks Delta) for now , but the openness trend suggests Iceberg support could follow; Microsoft is observing market demand.

- **Other players** – Confluent, a streaming platform company, integrated Iceberg as a first-class feature (their "Tableflow" product converts Kafka streams into Iceberg tables for downstream analytics) . This highlights Iceberg's reach beyond just SQL engines – it's seen as a universal table layer even for streaming/operational data bridges. Moreover, a slew of query engines like Apache Spark, Flink, Trino/Starburst, Impala, Druid, and others have built-in connectors for Iceberg . Essentially, if you have a favorite processing engine, it likely can read/write Iceberg tables by now.

This wave of support indicates that **Iceberg is emerging as the default "open table storage" format** for the industry, analogous to how S3 became the default storage API. The Kai Waehner blog post from July 2024 explicitly states: *"Apache Iceberg seems to become the de facto standard across vendors and cloud providers."* . However, the same sentence is tempered with "however, it is still at an early stage…with competing technologies trying to get momentum too" . This captures the state of play well: Iceberg is on a path to possibly win the "table format wars," but it hasn't completely won yet at the time of writing. Other formats like Delta Lake (open-sourced by Databricks) and Apache Hudi are still in use. In fact, Waehner notes that we are likely at the "innovation trigger" or rising slope of the Gartner Hype Cycle – lots of interest and adoption in pilots, but many organizations have not yet fully rolled out these technologies in production enterprise-wide . Most are in evaluation or early implementation phases. The **market adoption is early but growing rapidly**. As a point of comparison, he likens it to the "container wars" (Kubernetes vs Mesos vs Cloud Foundry) – ultimately one open technology (Kubernetes)

became the standard, and he suspects something similar will happen with Iceberg vs Hudi vs Delta . His personal bet is on Iceberg, largely because of the broad community and vendor backing it is amassing . Indeed, Iceberg has the advantage of being truly community-driven (not dominated by one company, unlike Delta Lake which is closely tied to Databricks), which appeals to neutral parties like AWS, Google, and open-source projects.

When comparing **Iceberg tables to traditional DB-native storage**, there are a few key considerations:

- **Interoperability vs Vendor Lock-in:** Iceberg tables are *portable across engines*. A table stored as Iceberg on S3 can be queried by Presto, Spark, Flink, Snowflake, etc. By contrast, a table inside a DB (e.g., in Snowflake's internal storage or in Oracle) can only be accessed by that DB's engine (or via its SQL interface). This interoperability is a huge advantage of Iceberg in multi-tool ecosystems. It effectively treats storage as infrastructure (any engine can use it) rather than binding storage to one compute engine. This resonates with teams that want to avoid lock-in and have flexibility to use different analytic tools . The Reddit summary from the Iceberg conference emphasized that "openness helps everyone… everyone was talking about avoiding vendor lock-in and retaining options" as a driver behind Iceberg's popularity .

- **Decoupling Compute/Storage & Independent Scaling:** Using Iceberg on object storage exemplifies the decoupled architecture – you can scale your data volume independently of compute clusters. Traditional DB storage often requires scaling the database engine or adding nodes as data grows. With Iceberg, you can dump a petabyte of data into S3 without having to provision a huge DB cluster; you only add compute when you actually query it. This ties into cost efficiency (storing lots of cold data in S3 is cheap, whereas in a database it could be very costly).

- **Feature Parity and Maturity:** One reason some may still choose DB-native storage is maturity. Decades of development have made traditional databases very robust in terms of query optimization, consistency, security features, etc. Iceberg (and similar) are newer and still catching up on certain features. For example, Iceberg provides *atomic append/replace transactions*, but it may not yet handle high-frequency upserts/deletes as well as an OLTP database. There are also "esoteric" SQL features or performance tricks that mature DBs have which the lakehouse stack is still adding. Waehner's blog notes we are in fast innovation; Iceberg, Delta, Hudi are adding improvements monthly . It's a moving target, but arguably *good enough* for many analytic workloads already. The question "are Iceberg tables the default over DB-native storage?" might imply: if starting a new analytic project, would one default to an Iceberg data lake or to a traditional warehouse? Increasingly, the answer appears to be Iceberg/lakehouse for many – but it can depend on use case.

- **Vendor Influence:** Some organizations remain on an all-in-one data warehouse (e.g. Snowflake's internal storage) because of convenience or existing investments. But even Snowflake is loosening that stance by integrating with Iceberg, acknowledging customer

interest in separating storage. On the other hand, Databricks' stance is interesting: they champion "open formats" but naturally prefer their Delta Lake. The competition between Iceberg and Delta Lake is still playing out; Databricks' acquisition of Tabular suggests they may aim to support both or at least benefit either way. For end users, it means there are multiple "table format" options, but Iceberg is the one not controlled by a single vendor, which likely gives it the community edge.

Today, one might say **Iceberg is on track to become the default table format for storage-centric architectures**, but we should be careful with the word "default." In cutting-edge data engineering circles, yes, Iceberg (and Delta/Hudi) are extremely popular and often the first choice for a new data lake implementation. For example, a Reddit discussion in late 2023 asked if the traditional warehouse is being skipped for lakehouse; one comment noted that "lakehouse tech has mostly caught up" to warehouses, with the likes of Databricks, Starburst, Dremio making great strides, and that ETL is definitely cheaper on the lakehouse . The same commenter boldly called Snowflake "a lake engine wrapped in warehouse limitations" , suggesting that even warehouses are internally recognizing the benefit of lake-style storage. However, another expert in that discussion cautioned: "Data lakehouse is still not mature enough to fully replace a data warehouse. Snowflake, Redshift, and BigQuery are still used a lot. Two-tier architecture (data lake + data warehouse) is also quite common." . This indicates that many enterprises hedge their bets – they keep a data warehouse for certain BI workloads that demand high performance or mature tooling, while using Iceberg/lakehouse for other workloads. Iceberg might be the default for new big data and AI pipelines, whereas the finance team might still use the trusty warehouse for monthly reports, for example.

In summary, **Apache Iceberg is rapidly becoming a standard for table storage on object stores,** supported by virtually all major data platforms – a strong indication that it is favored as the default open alternative to DB-native storage. Yet, its adoption at scale is still ramping up. It complements the "storage-centric" philosophy by liberating table data from specific database silos and allowing multiple compute engines to share one source of truth on S3. As Iceberg and its ecosystem mature, we can expect it to increasingly supplant proprietary storage formats. In sectors where it's already proven (e.g., streaming data integration via Kafka->Iceberg, or multi-engine analytics), teams are making it the default choice. Where extremely fine-tuned performance or certain legacy features are needed, traditional databases might hang on a bit longer. But the clear direction is that **open table formats like Iceberg are on track to be the new default for analytic data storage**, aligning with the broader trend that storage infrastructure is independent and shareable, rather than locked inside each database.

Prevalence of Query Engines on Parquet/S3 (Direct SQL on the Data Lake)

A key indicator of storage-centric architecture's rise is the volume of queries that run *directly on data in object storage* (e.g. Parquet files on S3) via distributed SQL engines, as opposed to

running inside a traditional database. The question posed – “What % of queries run directly on Parquet/S3 via SQL engines?” – points to understanding how common this approach has become. While an exact industry-wide percentage is hard to pin down, multiple datapoints and surveys suggest that **a significant share (perhaps a majority) of analytic queries are now executed against data in cloud object stores using “lakehouse” query engines.** In other words, querying data *in situ* on S3 (or similar) is no longer a niche – it’s mainstream for many organizations.

One strong piece of evidence comes from the **State of the Data Lakehouse 2024 survey** by Dremio. This survey found that *65% of respondents* (primarily IT and data professionals) said **more than half of their analytics workloads are running on a data lakehouse** rather than in a warehouse . In context, Dremio defines a “data lakehouse” as an architecture where analytics are done on a data lake (object storage with parquet/iceberg, etc.) with the ease of use approaching a warehouse. The fact that nearly two-thirds have crossed >50% of analytics in such an environment is telling. It implies that for those companies, **the majority of queries are indeed running directly on data in systems like S3**. The survey also indicated this trend is intensifying: at larger enterprises (10,000+ employees), 78% expected that within three years, *most* of their analytics (over 50%) would be on the lakehouse . This aligns with anecdotal evidence that new projects often default to data lake architectures, and only legacy or specialized workloads stay on older warehouses. The Dremio survey even suggested that lakehouses have “surpassed cloud data warehouses as the primary architecture for delivering analytics” in the industry perception .

From another angle, consider cloud provider metrics. **Amazon Athena**, a service that allows SQL queries directly on S3 data, has grown tremendously since its introduction – though exact usage stats are not public, AWS regularly highlights large customers using Athena for big data analysis. More concrete is **Amazon Redshift’s Spectrum** feature, which extends Redshift to query S3 data. AWS revealed that in 2024, Redshift customers collectively scanned **over 77 exabytes of data in data lakes (S3)** . This staggering number (77 EB in one year) underscores how frequently Redshift is being used not just as a warehouse but as a query engine for S3-resident data. It signals that AWS customers have tons of data in S3 which they query without loading into Redshift proper. Similarly, Google’s BigQuery Omni and BigLake features allow querying external cloud storage, and Microsoft’s Synapse and Fabric integrate data lake querying. The cloud vendors have essentially built capabilities to query Parquet on object storage because customers demand it – whether to avoid duplicating data or to analyze data that’s too large to move around easily.

Open-source SQL query engines like **Presto/Trino and Apache Spark SQL** started this trend earlier, particularly in tech companies. Facebook (where Presto originated) was famously running thousands of queries a day on a 300 PB data lake on HDFS (and now S3) as of mid-2010s . Netflix similarly built a platform where Spark and Presto queries run on an S3 data lake, handling the majority of their analytic workloads. These large tech firms demonstrated that *ad hoc* and even scheduled BI queries could be served from a decoupled storage layer effectively at scale. Over time, that approach trickled into mainstream enterprise. Today, we see even moderately sized companies using engines like Trino (perhaps via a vendor like Starburst)

to let analysts run SQL on the data lake. For instance, one case study noted Razorpay (a fintech company) had Trino clusters handling 100k queries per day from hundreds of users on their data lake . Such numbers rival the query counts one would expect in a data warehouse environment, showing that the lake query engines are up to the task.

So, while it's difficult to say "X% of all industry SQL queries hit Parquet on S3," the **directional answer** is that it's a large and growing proportion – likely *most new analytic queries are executed in that fashion* as opposed to within old-school warehouses. Many organizations now operate in a **hybrid mode**: some queries run in warehouses, some directly on S3. ChaosSearch's 2022 survey, for example, found 36% of enterprises were still exclusively using warehouses, meaning 64% had at least one other environment like a data lake in play . As of 2025, that exclusive warehouse percentage is surely even lower. This means the majority have the capability to query data on lakes. In Dremio's 2024 survey, respondents reported moving workloads off of warehouses onto the lakehouse – 42% said they moved data from cloud data warehouses into the lakehouse, and 35% from enterprise data warehouses, indicating a migration trend .

It's worth noting that **not all queries are equal**. Data lake engines excel at heavy analytical queries scanning large datasets. But for many quick, small queries (say a single record lookup or a small join), a traditional database might still be used. Thus, an organization might run 70% of its *data volume* through S3-based queries, but perhaps a higher number of *individual queries* still go to an OLTP or warehouse system for operational reporting. The question as phrased seems to be about percentage of queries, not data volume. There isn't a hard statistic on that, but given the above, we can infer: for analytics (OLAP-style) queries, we could be looking at roughly **half or more now executed on data lake storage** in many companies. Indeed, a knowledgeable Reddit user from data engineering consulting noted "95% of new pipelines are running on lakehouses/lakes" at their clients, with people adopting multi-compute patterns on a single data store . This suggests new development favors the lake model nearly universally.

However, a counterpoint is that many existing BI tools and dashboards still run queries against warehouses. So one might find that while new big data pipelines use Trino on S3, the finance team's Tableau dashboards still hit a Snowflake or Redshift database that's periodically synced from the lake. Thus the picture can vary by organization. The key insight remains: **direct SQL querying on Parquet in object storage is a widespread practice, not an edge case.** It's supported by all major vendors and often yields cost and flexibility benefits.

Why are so many queries shifting to run on S3/Parquet? The **benefits** include: eliminating costly data movement and duplication (you don't have to ETL data into a separate DB for querying – you query the source in the lake), leveraging elastic compute (you can spin up big query clusters only when needed), and unifying data access (all tools reading from one repository). The Dremio survey explicitly cites "reduced data movement and copies" and "unified data access" as perceived benefits of the lakehouse approach driving adoption . These translate into both agility and cost savings.

We should also mention performance: historically, one reason to copy data into a warehouse was to get fast queries via indexing, caching, etc. The gap has been closing – Parquet is a very efficient columnar format, and engines like Presto or Spark with distributed processing can achieve interactive performance for many analytics. There are still scenarios (concurrency, super low-latency needs) where dedicated warehouses shine, but improvements like data skipping, caching layers (Alluxio, etc.), and even new query acceleration services (e.g. AWS lake formation transactions with built-in indexing) are making direct lake queries faster. For example, Amazon's Redshift team implemented numerous optimizations for Iceberg/Parquet querying when stats are missing, achieving 2x faster performance in some TPC-DS queries on Iceberg tables . This indicates active efforts to make lake queries as fast as possible.

In conclusion, **a substantial and growing percentage of queries are executed directly on data in S3 using SQL engines.** Surveys and cloud usage metrics suggest that in many organizations, over 50% of analytical queries (and data processing jobs) now run on the data lake tier. The term "data lakehouse" has emerged precisely because so much querying is happening on the lake that it's functioning like a warehouse. As technologies mature, we expect this percentage to increase further, although traditional data warehouses will likely persist for specific use cases or as one layer in a larger ecosystem. The key takeaway is that querying Parquet on S3 is now a **normal, expected part of a modern analytics workflow**, underscoring the theme that storage (not the database) is where data "lives" and is directly analyzed.

Independent Procurement and Management of Compute vs Storage

In classic IT architecture, storage and compute were often provisioned together – e.g. buying a big database server that included storage disks, or an appliance with tightly coupled storage/compute. Today's cloud-first and data lake architectures have flipped that model: **compute and storage are procured and managed independently**, aligning with the principle of decoupled architecture. This means organizations treat storage as a separate resource (often a central pool of object storage) and compute as transient, scalable clusters or services that attach to that storage when needed. The evidence strongly indicates that this decoupling is not only happening, but is considered a best practice and is widely adopted in modern deployments.

From a cost and flexibility standpoint, the rationale was articulated in AWS's Well-Architected Framework: *"Decoupling storage from compute allows you to manage the cost of storage and compute separately"* and scale them independently to match different growth rates . Data volumes often grow exponentially, while compute needs might grow in a stepwise or flatter fashion. If you keep them coupled (for example, a data warehouse node with fixed storage per node), you end up over-provisioning one to satisfy the other. Thus, decoupling has become a design mantra. AWS explicitly advises to **use services that decouple compute from storage** so that you don't need to, say, add expensive compute nodes just to get more disk capacity . Nearly all cloud analytic services have moved in this direction: AWS Redshift's newer RA3 instances allow independent storage scaling (data is stored on S3, and you can add compute

nodes without moving storage) . Google BigQuery from the start separated storage (Colossus) and compute (Dremel execution), selling compute slots and storage separately. Snowflake's value prop was built on separate elastic warehouses and a central storage. The fact that even traditional OLTP databases are moving this way is telling – Amazon Aurora and similar cloud databases separate the storage layer (shared across AZs) from stateless compute nodes.

The question "Are compute and storage procured/managed independently?" implies an organizational shift as well. Indeed, many companies now have a **central data lake or storage team** that provides a large-scale storage service (maybe an internal S3 service or simply an AWS account with S3 buckets) and separate teams handle compute clusters (e.g., a data engineering team might manage a Spark cluster, an analytics team manages a Presto cluster, etc.). They budget for storage growth and compute usage separately. It's common to see cloud bills where S3 (storage) is one line item and EC2/EMR (compute) is another – teams track and optimize them independently. This is very different from, say, buying a Teradata appliance where you pay per integrated storage+compute unit.

From a technology trend perspective, **decoupled storage-compute architecture is now considered the new norm** for scalable systems. A Huawei cloud blog in late 2023 stated it plainly: *"Decoupled storage-compute architecture has become the de facto standard for distributed databases"* . The context there was transactional distributed databases in core banking systems – even in those traditionally conservative environments, decoupling is favored for elasticity and efficiency . The blog noted that as systems scaled, the old "coupled" approach (with data and compute on one server or tightly bound) led to massive redundancy and cost waste, whereas decoupling (using shared storage pools and independent compute) provided better high availability and easier expansion . Major new database products (Amazon Aurora, Alibaba PolarDB, Huawei GaussDB, etc.) have all adopted decoupled storage and compute, confirming it as the mainstream design . This trend in OLTP parallels what happened in analytics a few years prior.

For data analytics stacks, the decoupling manifests as **storage-centric architecture**: e.g., store all data in S3 (or similar), then use separate query engines (Presto, Spark, Flink, etc.) that read from S3. Those engines can be scaled up or down, turned off when not in use, or even completely swapped (use Trino today, maybe replace with Dremio tomorrow) without moving the data. Compute truly becomes a commodity layer that can be provisioned on-demand. A Medium blog by Andy Sawyer in 2023 mused on this, suggesting that as open table formats decouple storage, compute engines might become plug-and-play commodities – leading to competition on price/performance and value-add features since customers can switch engines easily when data is not locked inside them . This is a radical change from the past, where choosing a database meant committing your storage to it. Now, one could have data in Iceberg on S3 and try different SQL engines or processing frameworks on it to see which works best, without migrating the data.

In practice, **many organizations manage storage and compute lifecycles separately.** For example, an enterprise might decide on a multi-year strategy to consolidate all data in a data lake (object storage), investing in governance and cataloging for that layer, while treating

computing as ephemeral. Teams spin up compute clusters using infrastructure-as-code when needed, and tear them down after use. Storage persists as the durable layer; compute is transient. This also shows up in procurement: companies buy **reserved storage capacity** (since they know data will grow and live long-term), but use **spot or transient resources for compute** to save costs when possible. The question specifically asks if they are procured independently – the answer is largely yes, especially in cloud environments where you literally purchase storage (GB-months) separately from compute (vCPU-hours). On-prem, some companies have similarly moved to software-defined storage (like an on-prem S3 object store on a storage cluster) separate from compute servers (which might run Kubernetes or Hadoop, etc.).

It's worth noting that decoupling doesn't mean ignoring the relationship between compute and storage performance. For high performance, they must still be tuned together (network bandwidth, data locality caches, etc.). But management-wise, it's separate. For example, one can upgrade the storage system (add new disks, enable new replication) without having to overhaul compute nodes, and vice versa.

From an **organizational perspective**, decoupling sometimes created a shift in roles: storage admins and DBAs traditionally managed integrated systems. Now, you might have cloud architects managing S3 buckets policies and data engineers managing Spark clusters – they coordinate but can operate somewhat independently. There is also a trend of **central data platforms** providing self-service storage and compute. For instance, providing a "data lake as a service" internally where users get an S3 bucket and perhaps a recommended engine template, but they can choose what compute to apply. This independent management can accelerate innovation (teams don't have to ask the DBA team for more storage space on the database; they can just dump data to the lake, and later decide how to process it).

All this to say, **compute/storage independence is not just theory; it's happening at scale.** The combination of cloud billing models, architectural best practices, and supporting tech (like Kubernetes, object stores, etc.) has made it feasible. One strong piece of evidence: **73% of organizations that were initially only using on-prem data warehouses planned to migrate those to cloud (decoupled) in 1-3 years** , and for newer environments like data lakes and lakehouses, almost all are being deployed in cloud models rather than on integrated on-prem stacks . This was from the ChaosSearch survey, highlighting that the trend to cloud (which inherently uses decoupled services) is well underway.

It's instructive that even traditionally coupled architectures like Hadoop HDFS+YARN (which tied storage and compute in a cluster) have given way to decoupled ones. Hadoop's tight coupling required scaling all nodes together; now many have replaced HDFS with S3 and run ephemeral compute clusters on top (EMR, Dataproc, etc.). This yields better resource utilization and simplicity in scaling each tier.

To directly answer: **Yes, in modern deployments compute and storage are procured and managed as separate resources.** This decoupling is a core design principle of "cloud-native" data architecture. It provides flexibility (scale either as needed), cost transparency (separate

optimization of storage vs CPU costs), and supports the multi-engine usage patterns described earlier. The popularity of open table formats (Iceberg/Delta) actually reinforces this – by keeping compute independent of storage formats, you *enable* multiple compute engines. Conversely, adopting those formats necessitates decoupled thinking. The industry consensus is so strong that, as noted, decoupled architecture is basically considered the **"new default" or standard** for any large-scale system .

Multi-Cloud Interoperability via S3-Compatibility (Lift-and-Shift Portability)

One practical outcome of S3 becoming the standard storage interface is the ability to achieve **interoperability across clouds**. The question asks: "Are S3-compatible clouds interoperable for lift-and-shift?" In essence: if you have your data and applications using the S3 API, can you easily lift your entire data stack from one cloud and run it on another (or on-prem) with minimal changes? The answer is largely **yes – S3 compatibility provides a high degree of portability that facilitates lift-and-shift of storage-centric workloads.** Many organizations leverage this to avoid lock-in and to flexibly move between environments, although there are caveats (mostly around data transfer costs and some differences in services).

First, let's clarify "S3-compatible clouds." This refers to any storage service that implements the S3 API. Beyond AWS S3 itself, this includes a plethora of others: on-prem solutions (MinIO, Cloudian HyperStore, Ceph RGW, etc.), cloud vendor offerings or third-party layers (Wasabi cloud storage, Backblaze B2 has an S3 API, Google's "Interoperability API" for GCS, etc.), and even edge or private cloud deployments. Many private cloud vendors explicitly advertise "S3-compatible storage" to indicate that software written for AWS S3 will run on their platform .

The interoperability comes from the fact that the S3 API acts as an *abstraction layer*. If an application uses only standard S3 operations, it doesn't need to know or care who the provider is. **For example**, as cited earlier, an app using MinIO's S3 SDK can point to AWS S3 one day, and with a configuration change (endpoint URL and credentials) point to Azure Blob via an S3 gateway or to a MinIO server on-prem the next, with no code changes . This means data and applications are not tied to proprietary storage APIs of any single cloud. A concrete scenario: imagine a company develops an analytics pipeline that writes data to S3 and then runs Athena queries. If they later decide to repatriate that data to a private cloud to save costs, they could deploy MinIO on their private cloud, copy the data over (the heaviest part, but tools exist for bulk transfer), and then run an Presto/Trino cluster against MinIO with the same S3 calls that Athena was using. Their SQL code and data format (Parquet, Iceberg) remain identical – thus the stack is lifted-and-shifted fairly seamlessly. In fact, **cloud repatriation strategies often revolve around S3-compatibility**: companies first ensure their workloads use S3 APIs and standard formats, then they move data out of AWS to cheaper hosting when scale makes it worthwhile. The MinIO blog "Make it Rain" describes a customer who did exactly this at 500 PB scale, moving from AWS to Equinix colocation with MinIO, presumably leveraging the S3 API

compatibility to keep applications running . The result was massive savings, as noted earlier, and it was feasible because the core interfaces didn't change.

Similarly, multi-cloud analytics becomes easier with S3 compatibility. Some organizations maintain active data copies in multiple clouds (for redundancy or specialized processing). If all clouds support S3, the *same application* can run in AWS or Azure depending on needs, reading from whichever copy of the data is local – again requiring just a config/endpoint change rather than a rewrite. This is how **cloud-agnostic deployments** are designed: use S3 API for storage, Kubernetes for compute (cloud-agnostic container orchestrator), etc., and you have a mostly portable stack. A technical reference on multi-cloud APIs underscores this: APIs act as the glue; by using "cloud-agnostic APIs (e.g., S3-compatible storage APIs) teams reduce dependency on any single provider" . The specific example given: an app using S3 API via MinIO can switch between AWS S3, Azure Blob, or on-prem seamlessly . It's important to note Azure Blob is not natively S3-compatible (it has its own API), but one can run MinIO gateways or similar translation layers on Azure to present an S3 interface to the app. And Azure itself now has a S3-compatible endpoint (currently in preview via their Storage API interoperability or community solutions). So practically, yes you can use S3 API on all major clouds one way or another.

There's also a question of *data* interoperability: S3 API ensures you can programmatically move data out of one cloud into another easily (no need to convert it to a new format or through a different API). For example, you can use the same aws s3 cp commands or third-party tools to copy between AWS and another S3 endpoint. Many migration tools support S3-to-S3 moves (like AWS DataSync, or even rclone open source tool). If the target is S3-compatible, these tools work out-of-the-box. This means lift-and-shift isn't impeded by format conversion at the storage layer – it's just a transfer problem.

From an interoperability perspective, **even on-prem vendors highlight S3 for cloud integration**. Cloudian's marketing, for instance, points out that having on-prem S3 storage means you can "seamlessly move data and applications from on-prem to cloud" because the same API and data formats can be used . That is, an app could be developed on Cloudian S3 on-prem and later burst to AWS S3, or vice versa, without issues. This bi-directional portability is a selling point for hybrid cloud strategies.

Are there any limitations to this interoperability? A few subtle ones:

- **Consistency and Features:** AWS S3 has eventual consistency (though now strong read-after-write for new puts), and not all S3 implementations behave 100% the same in consistency guarantees. There might be edge-case differences, e.g., how versioning or locking works. If an application uses advanced S3 features (like object locking, bucket policies, etc.), those need to be supported on the target. Most major implementations do support the core S3 feature set, but minor differences can exist.

- **Performance tuning:** Applications might need tuning for different backend performance. For example, AWS S3 might have higher throughput than a small on-prem cluster, so batch sizes or parallelism could be adjusted after a move. But this is not a functional

incompatibility, just an optimization.

- **Ecosystem services:** While storage can move, any ecosystem around it (like AWS Athena which is serverless Presto on S3, or AWS Glue catalog) might not exist in the same form in another environment. In multi-cloud, you'd replace Athena with something like Trino or Presto on the other side. But since the data and format are portable, that's usually straightforward. It just means the *exact service* might differ, but the concept remains – e.g., you use open SQL engines instead of proprietary ones.

- **Data egress costs and network:** Lifting data out of one cloud incurs egress fees and time. This is often the biggest hurdle in "lift-and-shift" – not technical incompatibility, but plain logistics of moving many terabytes/petabytes across the internet. Some companies mitigate by using portable drives or direct connects, etc. The cost can be significant (cloud providers charge for outbound transfer), which ironically is something that motivates repatriation to avoid ongoing charges. But once moved, you've freed yourself of those.

Interoperability of S3-based clouds also means **workload portability**. For instance, if tomorrow a new cloud provider offers cheaper storage, an S3-using application could migrate data there and operate. We also see *cloud-neutral storage services*: e.g., Wasabi offers S3 storage at lower cost than AWS – many backup and big data apps support Wasabi by virtue of supporting S3 API. This gave customers leverage to choose non-AWS storage if desired. MinIO even suggests in their materials that using MinIO gives you the cloud operating model anywhere, allowing you to treat different environments as substitutable based on economics . That is exactly the interoperability story – the *cloud model runs anywhere* because the same interfaces exist anywhere.

A real-world demonstration of S3 interoperability was the case of **37Signals (Basecamp)** deciding to leave the cloud. They had their app data in S3 and databases in RDS; they pulled data out from AWS to their own data center. Because they relied on standard tech (they mentioned using object storage in their own DC as well), they were able to largely lift-and-shift with some re-engineering. Their blog and others cited by MinIO claim 60%+ cost savings and only moderate effort to switch . Interoperability was key to making that feasible.

In summary, **S3-compatible environments are highly interoperable, enabling true lift-and-shift of storage-centric workloads.** By adhering to the S3 API and open data formats, organizations ensure that their data layer is portable across clouds and on-prem. This reduces risks of cloud provider lock-in. Essentially, the S3 API has become a universal contract – any cloud or system implementing it can serve as a target or source for the data and applications. As a result, we see multi-cloud architectures where data replication happens via S3 API, cloud migration projects focusing on S3 data transfer, and hybrid setups using a common S3 interface for consistency. This interoperability not only exists in theory but is actively leveraged: **it is a driving factor behind multi-cloud strategies and cloud exit strategies** in many enterprises .

Thus, to directly answer: **Yes, using S3-compatible object storage creates interoperability that makes lift-and-shift between clouds practical.** Organizations using S3 APIs can port their data and apps to any other S3-speaking platform relatively smoothly, which is a significant advantage of storage-centric design.

TCO: “Storage-Centric” vs “DB-Centric” Data Stacks

One of the major considerations when adopting any architecture is cost. The question here asks whether total cost of ownership (TCO) is lower for “storage-centric” stacks (i.e., architectures built around a data lake with object storage + compute separated) versus “DB-centric” stacks (traditional databases or data warehouses that manage storage internally). The evidence from both surveys and case studies strongly indicates that **storage-centric architectures tend to have significantly lower TCO, especially at scale, than their database-centric counterparts.** This is, in fact, a primary driver for many organizations shifting to data lakehouse approaches.

Let’s break down where the cost differences come from. In a traditional database-centric stack, you often pay for an integrated solution – whether it’s a cloud data warehouse (e.g. Snowflake, where you pay for compute and also for storage at a marked-up rate) or an on-prem database appliance (where you buy expensive proprietary hardware). Storage in such systems can be costly (both in raw $/TB and in needing multiple copies of data for different uses). Also, you typically need to scale compute and storage together, potentially leading to inefficiencies as discussed. Conversely, a storage-centric stack uses commodity storage (object store) which is very cheap per TB, and open-source or flexible compute engines which can run on commodity hardware or spot instances. It also avoids duplicate copies – ideally one copy of data in the lake serving many purposes, rather than each department making its own warehouse extracts. All these factors contribute to cost savings.

Survey data: The 2024 Dremio survey provides a compelling high-level stat: *56% of respondents said adopting a lakehouse yielded (or will yield) savings of >50% in analytics costs*, and nearly 30% of large enterprise respondents expected >75% savings . Over half expecting to cut costs in half or better is huge. It implies that many have experienced that running workloads on a data lake (storage-centric) drastically reduced spend compared to the previous warehouse-centric approach. Furthermore, over one-third of respondents from big companies anticipated more than three-quarters cost reduction , which shows how dramatic they think the difference is at scale. These expectations likely come from pilots or partial migrations that demonstrated notable savings. The survey also noted that **cost efficiency was the top motivation (tied with ease of use) for adopting a lakehouse** . In their breakdown of motivations, 18% cited cost efficiency explicitly, and when combined with performance (scale/speed), it was a primary factor for 36% . This underscores that economics are front-of-mind; many felt the older warehouse model was simply too expensive or inefficient (“architectural inefficiencies” and “high costs associated with pipelines and copies” overshadowed their benefits).

Case studies and reports: There have been several public examples of companies moving away from cloud warehouses or big databases due to cost and achieving major savings. The MinIO blog references a few:

- **37signals (Basecamp)**: They left AWS, including moving off Amazon Aurora and S3 to self-hosted. They reported saving about $7M over 5 years by doing so, roughly a 60% savings in their cloud spend . While they moved *off* S3 in that case (to their own storage), they still used a storage-centric approach (open source object storage + MySQL on cheaper hardware). It exemplifies that paying cloud premiums for fully-managed DB/storage can be far more expensive than DIY with commodity parts if you have the scale.

- **Ahrefs**: An SEO company, moved from AWS to on-prem and reportedly saved large amounts (they publicly said they built their own data center for $400M with an 80% cost reduction over cloud). That figure is enormous but aligns with the notion of commodity vs managed pricing differences .

- **Twitter (X)**: After being acquired by Elon Musk, there was an effort to cut costs by reducing cloud dependency. They reportedly pulled some workloads from AWS and Google Cloud back to their own infrastructure. MinIO's blog alludes that X achieved substantial savings (though details are scarce) .

Another angle: cloud providers have implicitly validated the cost advantage of storage-centric models by offering cheaper storage tiers. For example, separating compute lets you shut compute off when idle – something not possible if storage/compute are tied. Many companies found their warehouse clusters ran 24/7 but were only busy part of the time, wasting money. With a decoupled stack, they could spin down compute when not in use, saving costs. This kind of operational cost optimization contributes to TCO improvement.

Object storage vs data warehouse storage cost: Raw object storage (S3) is quite inexpensive ($20–25 per TB per month for frequently accessed, and as low as $1–5 per TB per month for cold/archive tiers). In contrast, storing data in a data warehouse can effectively cost more because you often pay for compute nodes that include storage, or you pay a premium for performance. Snowflake, for instance, charges separately for storage at a rate slightly above raw cloud storage, and charges for compute time. But if data volume grows, your query costs often grow too as queries scan more. In a lakehouse, you might park a lot of data on cheap storage and only incur compute cost when actually scanning it (and you can choose cheaper/slower compute if needed for rarely accessed data). This flexibility drives down *total* cost.

Let's consider a specific scenario: the MinIO analysis of **100 PB over 5 years**. They estimated that storing and accessing 100 PB in AWS in a typical way (with S3 storage and some egress usage) would cost $33M per year for S3 plus additional for egress (which at that scale was $3M/year) . Over 5 years that's $166M. In contrast, using MinIO on-prem: $5M one-time for

hardware (NVMe storage servers) + $4.3M/yr software/operations, totaling about $30M over 5 years . That's about **1/5th the cost** (18%) of AWS's cost – an 82% reduction . They explicitly note these are "proven savings" echoed by others . Now, MinIO has an incentive to show on-prem is cheaper than cloud, but even within cloud, if one uses open-source compute on AWS and optimizes storage tiers, one can beat the costs of an all-in-one solution. Another internal detail: the breakdown shows AWS's big costs were egress and S3 charges, which are profit centers. In a storage-centric open stack, egress between your storage and compute might be within the same network (no charge), and storage is at cost. Cloud warehouses also often keep multiple copies of data (e.g., Snowflake might store data compressed, plus staging, etc.). With a single lake copy, you avoid duplicates.

Real-world user feedback often mentions "license cost" or "per-query cost" of warehouses as a pain point. If your usage is spiky or growing, those costs can skyrocket. In the Reddit discussion on skipping warehouses, one user said "ETL is most definitely cheaper on Lakehouse" and that warehouses had advantages in serving but now lakehouse options caught up . This aligns with the cost argument: doing transformations and heavy lifting on a lake is cheaper (no expensive DB compute hours). Another user from that thread pointed out one remaining challenge: super fast serving of BI dashboards sometimes still benefits from a warehouse, because if you query object storage directly for many small queries, it can be slow . Some organizations address that by doing light aggregation or caching for dashboards but still not fully copy data into a warehouse.

Counterpoint: We will detail in the counter-evidence section some challenges, but one to mention here: TCO isn't just infrastructure cost – it includes people and complexity. A potential argument against storage-centric could be that it requires more engineering effort (stitching together tools) whereas a one-stop warehouse might reduce development cost. However, many modern solutions (Databricks, AWS Lake Formation, etc.) aim to make the lakehouse approach as user-friendly as warehouses. And given the strong survey results about cost, it seems the industry believes the cost saving outweighs any additional engineering overhead. In fact, 21% in the Dremio survey cited "ease of use" as a benefit of lakehouse , showing that they aren't necessarily finding it more difficult; the gap is closing.

To directly answer the question: **Yes, a storage-centric stack generally yields lower TCO than a DB-centric stack, often dramatically so (50%+ cheaper), particularly at large scale.** The combination of cheap object storage, independent scaling, avoidance of vendor lock-in premiums, and reduction of redundant data copies drives costs down. This is validated by industry surveys and case studies of companies that have switched and saved substantially. Of course, actual savings depend on scale and how well-managed the storage-centric approach is – small-scale workloads might not see as much absolute difference, but at scale, the cost curve of a warehouse can become prohibitive, whereas a lake-based solution grows more linearly and utilises commodity economics.

In summary, **the TCO argument heavily favors storage-centric architectures** for big data and analytics. This is a key reason we see the rapid adoption of lakehouse designs: it's not just about technical elegance, it's about dollars and cents. The ability to cut analytics infrastructure

costs by half or more is a powerful incentive for enterprises to shift away from purely DB-centric stacks to ones where storage is central and shared.

Counter-Evidence and Challenges

While the trend and benefits of storage-centric, S3-based architectures are clear, it's important to examine **counter-evidence and contrary viewpoints**. Not every organization has fully embraced this model, and there are **areas where the traditional database-centric approach still holds or where the new approach has limitations**. Here we compile some of the key counterpoints and challenges:

1. Many Workloads Still Rely on Traditional Data Warehouses:

Despite the rise of data lakes and Iceberg, a substantial portion of organizations continue to use conventional data warehouses for important analytics. In ChaosSearch's 2022 survey, 82% of respondents were using data warehouses in production, and 36% were *exclusively* relying on warehouses without any data lake or alternative . This indicates that over a third of companies had not yet adopted a storage-centric or lakehouse approach as of that time. Even by 2023/24, industry observers note that **Snowflake, Amazon Redshift, and Google BigQuery remain heavily used "workhorses"** for analytics . A data engineer on Reddit argued in 2023 that "data lakehouse is still not mature enough to fully replace a data warehouse. Snowflake, Redshift and BigQuery are still used a lot. Two-tier architecture (data lake + data warehouse) is also quite common." . This suggests that many organizations hedge by using a lake for raw data storage and experimentation, but still funnel critical curated data into a warehouse for final reporting and dashboarding. The inertia of existing solutions, familiarity of tools, and certain capabilities of warehouses mean the transition is gradual. It's counter-evidence to any notion that "everyone" has moved to storage-centric; in reality, it's a spectrum and co-existence at present.

2. Performance and User Experience Concerns:

One reason some teams stick with a DB-centric approach is the performance and consistency it can offer for certain workloads. Query engines on object storage, like Trino/Presto or Spark, can struggle with **concurrent low-latency queries** that data warehouses handle gracefully. As one practitioner put it, queries can "take a bit too long and usability suffers if you're using something like Trino/Dremio/Athena directly on top of object storage without a little aggregation" for BI dashboards . Traditional warehouses often have sophisticated optimizers, caching, and indexing that make repetitive dashboard queries fast. By contrast, querying raw Parquet for every dashboard hit can be slower, necessitating workarounds like summary tables or result caches. This is essentially a **performance trade-off**: the lakehouse might be cheaper and more flexible, but you might need to invest effort to match the interactive performance of a dedicated warehouse for certain use cases. End-users (analysts) are sensitive to query latency; if switching to a lakehouse makes their BI tools slower or less responsive, there can be resistance. This constitutes counter-evidence in that not all workloads are best served directly from storage – some find a database endpoint still useful to achieve sub-second query times on

smaller aggregated data. However, technologies like materialized views on lakehouse data, or new query acceleration layers, are emerging to mitigate this gap.

3. Maturity of Tooling and Ecosystem:

The ecosystem around data warehouses (BI tools, reporting, governance) is mature and user-friendly. The storage-centric ecosystem, while rapidly advancing, is younger. There is a **lack of a clear, vendor-agnostic definition of "lakehouse"** architecture, which can lead to confusion or integration challenges . Each vendor (Databricks, Dremio, etc.) has their spin, and not all tools plug-and-play as smoothly as with a well-trodden warehouse like SQL Server or Oracle. For example, some enterprise BI platforms had to add connectors for engines like Presto or for Iceberg – in earlier stages, support was limited. This means a company may hesitate to abandon a warehouse if their favorite BI tool doesn't fully support the new stack's features (say, time travel queries on Iceberg or something). **Data governance** can also be trickier across a distributed lakehouse environment compared to an all-in-one warehouse with built-in controls. These maturity issues are gradually being solved (modern data catalogs and governance tools now handle lakehouse metadata, etc.), but they present short-term challenges (i.e., "gaps" where the warehouse is still easier to manage). As Waehner noted, we are in early stages; not all organizations have the in-house skills to operate a best-of-breed open data stack, whereas a managed warehouse might be simpler for them.

4. Iceberg and Table Format Adoption is Not Universal Yet:

Although momentum is behind Iceberg, **it's still early in adoption lifecycle**. Some companies are taking a "wait and see" approach while the "table format wars" play out . The presence of three competing formats (Iceberg, Delta, Hudi) can cause hesitation – one might worry about picking the "wrong" one or having to switch later. So, some stick to their existing database or basic parquet files until a winner is clear. Additionally, the Iceberg project itself is evolving; features like advanced caching indices (Puffin), or certain write performance optimizations are still being developed . A conservative enterprise might require those capabilities to match their current DB's performance before fully switching. **In short, the default is not yet Iceberg everywhere; many are only piloting it.** Waehner's assessment that "most organizations are still evaluating, not adopting these table formats in production across the organization yet" rings true . It's a caution that while Iceberg is promising, we shouldn't assume everyone has flipped to it overnight. There will be a period where some still rely on tried-and-true database storage, especially for critical data, until they gain confidence in the newer tech.

5. S3 API Limitations and Cloud Nuances:

While S3 compatibility is great for interoperability, some critics point out that not every cloud service or application is truly interchangeable via S3. For example, **Azure's object storage (Blob Storage) doesn't natively support all S3 semantics** (though workarounds exist). This means a lift-and-shift from AWS S3 to Azure might require an intermediate layer or adjustments. If an organization heavily uses AWS-specific features (like S3 access control policies or specific AWS IAM roles attached to S3), moving to another S3 system means reimplementing those

security controls differently. There's also the fact that **data gravity is real** – large datasets are hard to move, so interoperability is limited by the bandwidth and costs. Some companies find they are effectively "stuck" in one cloud because the egress cost to move petabytes is too high or because the downtime to transfer is unacceptable. In those cases, the promise of easy lift-and-shift is theoretical – practically, they might re-architect gradually or adopt multi-cloud in new projects rather than moving everything. So, while S3 provides a path for portability, in reality full **cloud interoperability can be constrained by non-technical factors** like data transfer costs and cloud-specific integrations.

6. Human and Process Factors:

Adopting a storage-centric stack can require a culture shift and new skills. Teams accustomed to relational databases must learn about object storage, distributed query planning, and data format management. There can be missteps – e.g., a "data swamp" scenario where a data lake is implemented poorly leading to disorganized data that's hard to use. One tongue-in-cheek comment was "Data swamp is the industry standard" – implying many data lakes fail to deliver value due to lack of proper governance. A poorly managed storage-centric approach could indeed have a higher TCO in terms of people time (data scientists struggling to find and trust data, etc.) compared to a well-managed warehouse. Thus, **the benefits touted assume competent implementation**. Not every organization may achieve that easily. Some might overspend on cloud storage or make too many copies in the lake due to lack of discipline, negating savings. Others might find their data engineers spending more time on pipeline maintenance in a DIY stack than they did with a turnkey warehouse solution. These factors can act as counterpoints, showing that a storage-centric model is not a silver bullet – it must be paired with strong data management practices to truly pay off.

7. Specific Use Cases Still Needing Integrated Solutions:

Certain workloads – especially **transactional or real-time workloads** – are not suited to an S3-based architecture. For example, if you need millisecond-level updates and queries (high-frequency trading, real-time analytics dashboards with constant updates), a specialized database or streaming database might be necessary. The storage-centric approach (object store + eventual consistency) isn't designed for high concurrency OLTP or sub-second updates. Thus, organizations still maintain OLTP databases for operational data, and that's part of the overall data stack cost. One could argue this is outside the analytics scope, but it shows there are domains where the DB-centric model remains essential. Even in analytics, some scenarios (complex multi-hop queries or very high user concurrency) might perform better on a tuned data warehouse.

In summary, the counter-evidence highlights that **while "storage becomes the infrastructure" is a strong trend, it is not without challenges and it hasn't completely displaced traditional approaches yet**. Many enterprises adopt a hybrid approach, keeping what works from the old while exploring the new. They might land all raw data in S3 (embracing storage-centric for storage), but still push final curated data into a warehouse for easy consumption (a nod to DB-centric convenience). The journey to fully storage-centric can be

gradual and must overcome issues of performance, tooling maturity, and internal skills. Moreover, cost advantages, though real, assume certain scale and expertise – smaller shops or those without skilled data engineers might actually find a managed warehouse cheaper in total if they can't efficiently operate a complex open-source stack.

In conclusion, the counterpoints do not disprove the trend but rather provide nuance: they remind us that *one size may not fit all*, and that the best architecture might combine elements. Storage-centric infrastructure offers great promise and is clearly gaining ground for big data, but enterprises must weigh performance needs, existing investments, and team capabilities. It's likely that for the foreseeable future, **storage-centric and DB-centric paradigms will coexist**, each used where appropriate, even as the balance keeps shifting toward the storage-centric model as technology matures.

Conclusion

The data and analysis above indicate a clear industry movement toward treating **storage as the foundational infrastructure layer** for data platforms, with compute layered on top as needed. Object storage – frequently accessed via the S3 API – has effectively become a ubiquitous substrate, akin to a new "data operating system" on which various processing engines run. We see that most organizations are now gravitating to this model to capitalize on its flexibility, interoperability, and cost advantages. Apache Iceberg and similar table formats have emerged to fill the gap between the simplicity of file storage and the conveniences of databases, enabling robust analytics directly on the storage layer. This, in turn, allows a proliferation of query engines and tools to access the same underlying data, breaking the monopoly of monolithic databases over data access.

Crucially, this paradigm shift decouples growth in data from growth in compute, letting each scale on its own economic curve. Companies can store ever-increasing volumes of data without linearly increasing their compute spend, using cheap storage for bulk data and spinning up compute only when value needs to be extracted. The **multi-cloud and hybrid cloud benefits** of this approach are also significant – by adhering to standard APIs and formats, data becomes portable and not locked into any single vendor's ecosystem. This gives organizations unprecedented freedom to choose or change their cloud providers and to optimize for cost or performance by moving workloads around.

The evidence of lower total cost of ownership, in particular, is driving even hesitant organizations to re-evaluate legacy database-centric architectures. When more than half of industry professionals report expecting over 50% cost savings with a lakehouse approach , CFOs and CIOs take notice. In an era of ever-growing data (with AI and IoT fueling exponential expansion), the traditional approach of copying all data into expensive warehouses is increasingly unsustainable. Storage-centric architectures offer a way to **scale affordably**, using commodity storage and open source software, without sacrificing analytical power. Early adopters that have successfully transitioned – or at least partially transitioned – are reaping substantial savings and flexibility gains, as seen in the case studies.

However, the counterpoints remind us that this transition is a journey. The **future likely lies in a balanced approach**: using the storage-centric philosophy as the default, especially for new big data and AI workloads, while still integrating optimized engines or warehouses for the pieces they serve best (for instance, extremely low-latency BI or certain regulated data management scenarios). Over time, as open data technologies mature, we can expect the need for separate legacy warehouses to diminish further. The trajectory is similar to other technology evolutions (consider how microservices and cloud broke apart the old single-vendor monoliths in application infrastructure; a similar unbundling is happening in data infrastructure).

In summary, the research direction “Storage becomes the infrastructure” is strongly validated by current trends. **Most data teams are indeed rallying around object storage (S3 API) as the common denominator**, enabling everything from machine learning pipelines to SQL analytics in a unified storage layer. **Apache Iceberg and data lakehouse engines are rapidly climbing to parity with traditional databases** for many analytic use cases, and their adoption is accelerating. **Compute and storage are now procured independently and scaled on demand**, giving organizations more control over cost and performance tuning. **Portability across environments is greatly enhanced**, reducing long-term risks and fostering a more open data ecosystem. And importantly, **the economics favor this approach**, particularly at scale, with large organizations standing to save 50–75% on analytics infrastructure costs by embracing storage-centric stacks .

As with any significant shift, challenges exist – from ensuring performance to managing cultural change – but the momentum and continual improvements in technology are addressing these year by year. Many experts predict that in the coming years, we will see a consolidation around one primary open table format (likely Iceberg), deeper integration of lakehouse capabilities in BI tools, and further reduction of any performance gap via intelligent caching and query acceleration on data lakes. At that point, the reasons to maintain siloed, database-centric stacks will diminish further.

For now, the prudent strategy for organizations is to **build their data architecture around a robust storage core (an “enterprise data lake” on S3 or equivalent), adopt open formats, and enable multiple compute frameworks to access it**. This provides maximal flexibility – allowing them to plug in new cloud services, advanced analytics, or AI tools directly to the same data repository. It future-proofs the infrastructure against changing technology trends (since the data is not locked in any one tool) and positions them to take advantage of multi-cloud or hybrid deployments if needed. Essentially, by making storage the anchor, organizations can navigate the rapidly evolving data technology landscape with confidence that their most precious asset – the data – remains controlled, accessible, and cost-effective.

Thus, the research conclusion is that **yes, storage is becoming the key infrastructure layer** for data systems. The industry is steadily answering the guiding questions in the affirmative: S3-like APIs are standard, Iceberg-like tables are on track to become default, a large share of queries now hit S3/Parquet directly, compute and storage are procured separately, S3-compatible clouds enable easy workload moves, and the storage-centric approach generally provides lower TCO than legacy approaches. This represents a paradigm shift in how we design

and think about data platforms – one that decentralizes and commoditizes the processing, while centralizing and standardizing the storage. It's an exciting time, as this new equilibrium promises more openness, innovation, and efficiency in data management than ever before.

Footnotes: (References are provided as Chicago-style footnotes with numeric indices, corresponding to sources supporting the statements made.)

1. Cloudian, *"S3 Compatible Storage: On-Prem Solutions Compared,"* Cloudian Blog (2021), explaining that the Amazon S3 API has become the de facto standard interface for object storage across the industry .

2. Cloudian, ibid., noting that most large-scale cloud storage is object storage using the S3 API, and listing on-premises vendors (NetApp, Dell/EMC, etc.) that support S3 compatibility .

3. CRN, *"MinIO CEO: For AI Data 'It's Us Versus AWS',"* interview with AB Periasamy (2024), in which the CEO claims MinIO's embrace of the S3 API helped make it an industry standard beyond AWS, with MinIO's adoption now "larger than Amazon S3's" in terms of deployments .

4. Milvus (Zilliz), *"What is the role of APIs in multi-cloud strategies?"* (n.d.), discussing how using cloud-agnostic APIs like S3 enables portability; gives example that an app using MinIO's S3-compatible API can switch between AWS S3, Azure Blob, or on-prem MinIO without code changes .

5. Cloudian, *"S3 Compatible Storage,"* Cloudian Blog, highlighting that only the S3 API was designed in the internet era with features like multi-part upload for reliability over WAN, which older protocols lack .

6. Kai Waehner, *"Apache Iceberg – The Open Table Format for Lakehouse AND Data Streaming,"* personal blog (July 13, 2024). The author notes Apache Iceberg is on track to become a de facto standard for table formats across vendors, but that we are still in early stages with most organizations evaluating rather than fully adopting these formats yet .

7. Kai Waehner, ibid., comparing the Iceberg vs Hudi vs Delta "table format wars" to past "container wars," and predicting Iceberg likely to win due to broad community support and many vendors implementing it .

8. Reddit r/dataengineering, *"What's next for Apache Iceberg?"* discussion thread (1 year ago, c. 2023). One summary from an attendee of an Iceberg conference stated that Iceberg is being adopted by some of the largest companies (Netflix, Apple, Google) and that its combination of benefits (cost savings on S3, time travel, etc.) "makes sense for everyone," driving its momentum .

9. Reddit r/dataengineering, ibid., where participants note that AWS is favoring Iceberg in its services (Glue/Athena) over alternatives .

10. Snowflake, *"How Apache Iceberg Is Changing the Face of Data Lakes,"* Snowflake Blog (n.d.), describing Iceberg as an open table format at the core of the open data lakehouse revolution (Snowflake now supports Iceberg tables externally). [Snowflake's blog corroborates vendor support but was not directly excerpted above].

11. Dremio, *"2024 State of the Data Lakehouse"* (Whitepaper, Nov 2023), reporting that 65% of surveyed companies have more than half of their analytics running on a data lakehouse (object storage + open formats), surpassing cloud data warehouses as the primary analytics architecture .

12. Dremio, ibid., noting that 56% of respondents expect >50% cost savings from lakehouse adoption, and 36% of respondents (and 78% of large-enterprise respondents) already see more than half of analytics on the lakehouse .

13. AWS Big Data Blog, *"Amazon Redshift out-of-the-box performance innovations for data lake queries"* (July 31, 2025), revealing that in 2024 Redshift customers queried over 77 exabytes of data stored in data lakes (S3) . This demonstrates the huge volume of queries going against S3 via Redshift Spectrum and related features.

14. ChaosSearch, *"2022 Data Delivery and Consumption Patterns Survey: Highlights"* (May 26, 2022). This survey found 82% of enterprises use data warehouses, and 36% rely exclusively on them with no data lakes yet . It also indicated a trend towards cloud and data lakes in the next 1-3 years .

15. Reddit r/dataengineering, *"Is there a trend to skip the warehouse and build on lakehouse instead?"* discussion (archived, c. 2023). A user notes "95% of new pipelines are running on lakehouses/lakes" in their experience , but another cautions that "data lakehouse is still not mature enough to fully replace a data warehouse… Snowflake, Redshift and BigQuery are still used a lot" , highlighting that two-tier (lake + warehouse) architectures remain common.

16. Reddit r/dataengineering, ibid., discussion about performance where a user sciencewarrior explains that direct queries on object storage (Trino/Dremio/Athena) can be slow for dashboards unless aggregated, whereas warehouses handle those faster . This points out current performance gaps.

17. Andy Sawyer (Medium), *"Decoupling Compute and Storage with Open Table Formats"* (Oct 2023), observing that the decoupling trend is accelerating and that Snowflake, Databricks, Microsoft, etc. all are embracing open formats (Iceberg/Delta) which decouple the data storage from the compute engine . He cites a Snowflake earnings call noting enterprise customers' preference for open table formats and mentions Snowflake

allowing Iceberg on customers' S3, Databricks open-sourcing Delta, Microsoft Fabric using Delta, and Fivetran loading to Iceberg/Delta . This indicates broad industry moves towards decoupled data storage.

18. Huawei Blog, *"Decoupled Storage-Compute Architecture: The New De facto Standard for Distributed Databases"* (Nov 30, 2023). It states plainly that decoupled storage-compute has become the de facto standard for distributed databases and notes that major products (Aurora, PolarDB, etc.) have shifted to this architecture . Also, that major banks are rebuilding core systems with decoupled architecture, citing benefits in cost and scalability .

19. AWS Well-Architected Framework – Analytics Lens, Best Practice 11.1 *"Decouple storage from compute"* (AWS Documentation, 2022). It advises using services that allow independent scaling of storage and compute to optimize costs, noting that data often grows faster than compute needs .

20. MinIO Blog, *"Make it Rain: How Repatriating Your Public Cloud Workload Can Deliver Millions in Savings"* (Jonathan Symonds, April 30, 2024). This piece provides a cost comparison: 100 PB in AWS S3 vs on-prem MinIO, showing ~$166M vs $30M over 5 years (an ~82% cost reduction by moving off cloud) . It also references examples like 37signals, Twitter (X), Ahrefs achieving ~60% or more cost savings from cloud repatriation (even though those were not using MinIO specifically) .

21. Dremio, *"2024 State of the Lakehouse"* (Whitepaper), highlighting motivations for lakehouse adoption: cost-efficiency was a top factor (edging out or equal to ease of use) especially in larger organizations . Also notes 42% of respondents moved data from a cloud data warehouse into the lakehouse, illustrating migration away from warehouses .

22. Reddit r/dataengineering, *"Is there a trend to skip the warehouse…"* thread, where one commenter quips about data lakes turning into "data swamps" , reflecting that poorly managed lakes can become unstructured and hard to use (a risk/critique of storage-centric approach if governance is weak).

23. Reddit r/dataengineering, ibid., user proverbialbunny points out lack of a vendor-agnostic specification for "lakehouse" and confusion around the term, implying some conceptual/standards muddiness compared to well-defined warehouses .

24. Gartner (via Upsolver), *"Cut Data Warehouse Costs in Half with Apache Iceberg"* (Upsolver blog, referencing a survey). It suggests organizations believe lakehouse can result in 50%+ cost savings. [Not directly cited above, but aligns with the Dremio survey].

25. InfoWorld, *"Repatriating from the cloud saves 60% or more"* (article referenced in MinIO blog) . This likely covers cases like 37signals repatriation story, providing independent

journalism confirmation of those savings.

These footnotes correspond to the in-text bracketed citations, providing source and context as per Chicago style footnote conventions.