WHITEPAPER

High-Performance Applications With Distributed Caching

Get integrated caching from a complete NoSQL solution

Couchbase

TABLE OF CONTENTS

Executive summary

Caching can boost application performance as well as reduce costs.

For many web, mobile, and Internet of Things (IoT) applications, distributed caching is a key requirement, for improving performance and reducing cost. By caching frequently accessed data – rather than making round trips to the backend database – applications can deliver highly responsive experiences. And by reducing workloads on backend resources and network calls to the backend, caching can significantly lower capital and operating costs.

Distributed caching solutions solve for three common problems – performance, manageability, scalability – in order to gain effective access to data in high-quality applications.

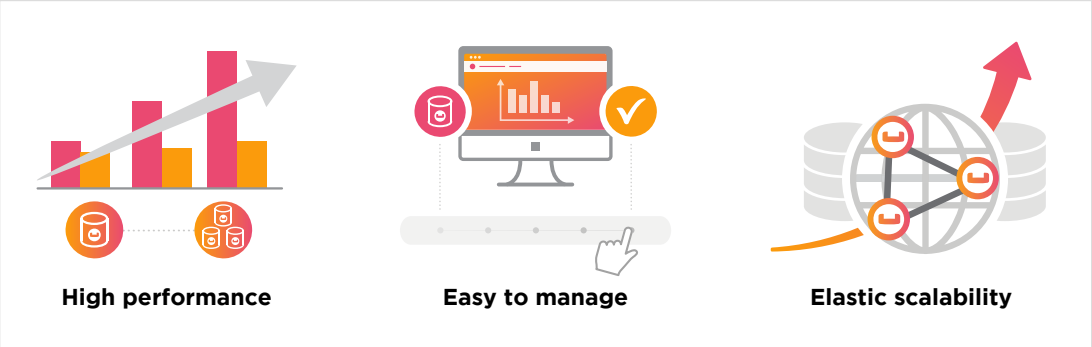

Figure 1: The three common requirements for distributed caching solutions

High performance is a given, because the primary goal of caching is to alleviate the bottlenecks that come with traditional databases. This is not limited to relational databases, however. NoSQL databases like MongoDB™ also have to make up for their performance problems by recommending a third-party cache, such as Redis, to service large numbers of requests in a timely manner.

Caching solutions must be easy to manage, but often are not. Whether it's being able to easily add a new node, or to resize existing services, it needs to be quick and easy to configure. The best solutions provide command line, GUI, DBaaS (database-as-a-service), and REST APIs to help keep things manageable.

Elastic scalability refers not only to the ability to grow a cluster as needed, but also refers to the ability to replicate across multiple data centers (cloud and/or on-prem). Cross data center replication (XDCR) is a feature that is often missing or performs poorly across many caching technologies. To achieve this scalability, several products often have to be glued together, thereby decreasing manageability and greatly increasing cost.

Based on Couchbase's experience with leading enterprises, the remainder of this document:

- Explains the value of caching and describes common caching use cases
- Details the key requirements of an effective, highly available, distributed cache
- Describes how Couchbase Server provides a high-performance, low-cost, and easy-to-manage caching solution
- Explains key differences in architecture and capabilities between Couchbase Server, Redis, and Memcached

Caching vs Buffering

Caching and buffering are techniques that are often conflated. While many databases make heavy use of memory for buffering, this does not mean they have a managed cache. Buffering stores transitory data in memory temporarily while it's being read or written. Caching stores data in memory until it's evicted.

Importance of a cache in enterprise architectures

Today's web, mobile, and IoT applications often need to operate at large scale: thousands to millions of users, terabytes (or even petabytes) of data, submillisecond response times, multiple device types, and global reach. To meet these requirements, modern applications are built to run on clusters in distributed computing environments, either in enterprise data centers or on public clouds such as Microsoft Azure, Amazon Web Services (AWS), or Google Cloud Platform (GCP).

Caching is a technology to boost application performance as well as reduce costs. By caching frequently used data in memory – rather than making database round trips and incurring disk IO overhead – application response times can be dramatically improved, typically by orders of magnitude.

In addition, caching can substantially lower capital and operating costs by reducing workloads on backend systems and reducing network usage. In particular, if the application runs on a relational database like Oracle – which requires high-end, costly hardware in order to scale vertically – a distributed, horizontally scaling caching solution that runs on low-cost commodity servers can reduce the need to buy and manage expensive resources.

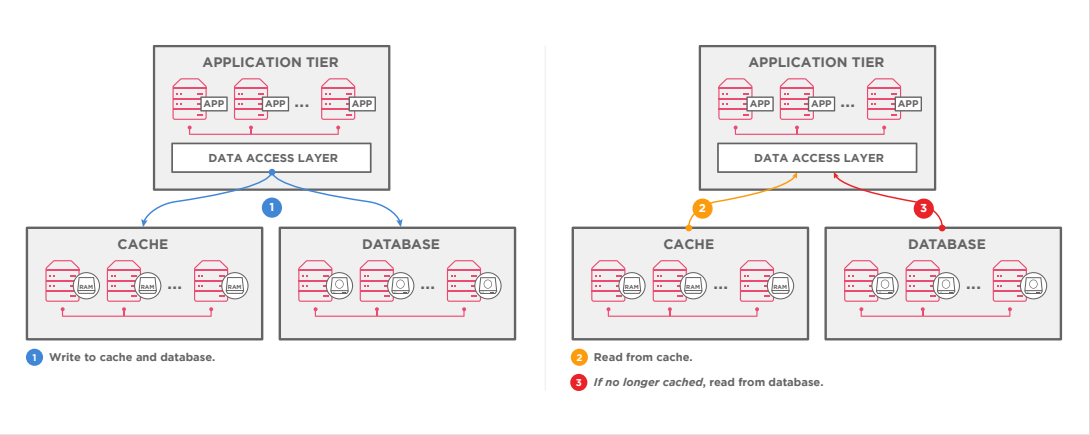

Figure 2: High-level database cache architecture

Common use cases

Caching is used across numerous applications and use cases, including:

- **Speeding up RDBMS** – Many web and mobile applications need to access data from a backend relational database management system (RDBMS) – for example, inventory data for an online product catalog. However, relational systems struggle with large scale, and can be easily overwhelmed by the volume of requests from web and mobile applications, particularly as usage grows over time. Caching data from the RDBMS in memory is a cost-effective technique to speed up the backend.

- **Managing usage spikes** – Web and mobile applications often experience spikes in usage (for example, seasonal surges like Black Friday, Cyber Monday, during prime time television, etc.). Caching can prevent the application from being overwhelmed and can help avoid the need to add expensive backend resources.
- **Mainframe offloading** – Mainframes are still widely used in many industries, including financial services, government, retail, airlines, and heavy manufacturing. A cache is used to offload workloads from a backend mainframe, thereby reducing MIPS costs (i.e., mainframe usage fees charged on a "millions of instructions per second" basis), as well as enabling completely new services otherwise not possible or cost prohibitive utilizing just the mainframe.
- **Token caching** – In this use case, tokens are cached in memory in order to deliver high-performance user authentication and validation. eBay, for example, deploys Couchbase Server to cache token data for its buyers and sellers (over 100 million active users globally, who are served more than 2 billion page views a day).
- **Web session store** – Session data and web history are kept in memory – for example, as inputs to a shopping cart, real-time recommendation engine on an e-commerce site, or player history in a game.

The use of a cache should not place undue burden on the operations team. It should be reasonably quick to deploy and easy to monitor and manage.

Key requirements

Enterprises generally factor six key criteria into their evaluation. How you weigh them depends on your specific situation.

1. **Performance:** Specific performance requirements are driven by the underlying application. For a given workload, the cache must meet and sustain the application's required steady-state targets for latency and throughput. Efficiency of performance is a related factor that impacts cost, complexity, and manageability. How much hardware (RAM, servers) is needed to meet the required level of performance?
2. **Scalability:** As the workload increases (e.g., more users, more data requests, more operations), the cache must continue delivering the same steady-state performance. The cache must be able to scale linearly, easily, affordably, and without adversely impacting application performance and availability.
3. **Availability:** Data needs to be always available during both unplanned and planned interruptions, whether due to hardware failure or scheduled system maintenance, so the cache must ensure availability of data with as much uptime as possible, and be kept as "warm" as possible to ensure performance.
4. **Manageability:** The use of a cache should not place undue burden on the operations team. It should be reasonably quick to deploy and easy to monitor and manage. All other things equal, simplicity is always better. Adding a cache to your deployment should not introduce unnecessary complexity or make more work for developers.
5. **Affordability:** Cost is always a consideration with any IT decision, both upfront implementation as well as ongoing costs. Your evaluation should consider total cost of ownership, including license fees as well as hardware, services, maintenance, and support.

What is a "warm" cache?

A "cold" cache is an empty, or near-empty cache that is yet to be filled with active data. The benefits of caching will not be reached until the cache gets "warmer", i.e. starts to fill up with active data. Since a cache uses memory, any reboots or crashes, especially with a non-distributed system, will result in restarting with a "cold" cache.

In designing Couchbase Server, the Memcached engineers extended its high performance and simplicity to incorporate high availability and easy manageability.

Distributed caching with Couchbase Server

Couchbase Server (and Couchbase Capella™ DBaaS) has become an attractive alternative to caching tools like Redis and Memcached. It's the only solution that fully meets the requirements of modern web, mobile, and IoT applications that need to support thousands to millions of users, handles large amounts of data, and provides highly responsive experiences on any device.

For many enterprises, Couchbase hits the sweet spot by delivering performance, scalability, and availability, while being easy to deploy and manage. Couchbase is an affordable choice, with enterprise support available from Couchbase.

General-purpose NoSQL database with Memcached roots

Couchbase Server is a general-purpose, document-oriented NoSQL database and has a strong caching heritage. Couchbase founders include the engineers who drove Memcached development in conjunction with the engineers who open sourced it at LiveJournal and were using it at Facebook. LiveJournal was one of the internet's first social networks, before Facebook and Twitter, and it faced frequent usage spikes as well as continuously growing workloads that overwhelmed backend resources.

To solve those issues, LiveJournal engineers built Memcached as a high-performance cache that's "dead simple" to use. While it squarely met the goals for high performance and simplicity, Memcached was not designed as a *high availability* caching solution, so features like auto failover and cross data center replication (XDCR) were not built into the product.

Architectural advantages

Couchbase Server was built for *distributed* caching with a focus on agility, manageability, and scalability for mission-critical applications.

A benchmark run on Google Cloud Platform showed 50 nodes of Couchbase Server sustained 1.1 million operations per second. To deliver comparable performance, Apache Cassandra needed 300 nodes.

Perform at any scale

- **Memory and network-centric:** Couchbase's memory-first architecture, with integrated document cache, was designed to deliver high-throughput rates in distributed computing environments while providing submillisecond latency and resource efficiency. The network-centric architecture with a high-performance replication backbone allows new workloads to be added while maintaining performance at scale.
- **Always-on, edge-to-cloud:** Couchbase is designed to be fault tolerant and highly resilient at any scale and on any platform – physical or virtual – delivering always-on availability in case of hardware failures, network outages, or planned maintenance windows.
- **Consistent performance at any scale:** Couchbase is designed to provide linear, elastic scalability for web, mobile, and IoT applications using intelligent, direct application-to-node data access without additional routing and proxying configuration and overhead.
- **Workload isolation and optimization:** Adding or removing nodes can be done without any downtime or code changes. Couchbase's Multi-Dimensional Scaling (MDS) allows users to isolate their workloads while incrementally increasing access to specific services on the cluster resources as needed.

Manage with ease

- **Global deployment with low write latency:** Couchbase is often selected specifically because of its simple and powerful active-active cross data center replication (XDCR) capabilities that support varying types of replication topologies (unidirectional and bidirectional in any combination).
- **Flexible deployment options:** Multiple methods of deployment are supported including on-prem, hybrid, cloud, Kubernetes, and Couchbase Capella DBaaS.
- **Consistent performance when adding microservices:** Couchbase eases management with automatic sharding, replication, and failover for easy scale out and high availability. Autonomously maintain application availability across upgrades, node failures, network failures, or even cloud provider failures (via XDCR). All functionality is made available across physical, virtualized, public cloud, container, and DBaaS environments.
- **Full-stack security:** End-to-end encryption of Couchbase data is available both over the wire and at rest. Flexible security options are possible with role-based authentication that supports LDAP, PAM, and X.509. Embedded data and administrative auditing tools allow for robust control of enterprise data.
- **Affordability:** Licensing costs of Couchbase is typically a fraction of other solutions like Oracle Coherence, often as much as 80% less. Couchbase Server can be freely downloaded without any license fees, allowing you to prototype and experiment with zero cost or risk. Couchbase Capella has a free trial, as well as Basic, Developer Pro, and Enterprise pricing plans. Couchbase is designed to run efficiently with data volumes that are larger than memory, not requiring costly scale-out to more nodes just to fit more data in memory (like memory-only caches, including Redis). And because Couchbase is far less complex to deploy and manage, it takes fewer resources to support it.

Develop with agility

- **Flexible schema for continuous delivery:** Couchbase can handle both simple and complex JSON documents. Developers can access data through a flexible data model that adjusts as needed. A new field can be easily added and then made available to queries. Schema changes are not onerous and do not result in complex remapping or downtime while testing new data structures.
- **Full-featured SQL for JSON:** Standard SQL has been extended for JSON querying and analytics to allow developers to use familiar database skills with Couchbase.
- **Versatile data access patterns:** Couchbase's set of data access methods include key-value lookup, SQL++ querying, full-text search, real-time analytics, and server-side eventing – available across cloud, mobile, and edge devices.
- **No hassle scale out:** Application code using Couchbase does not need to change when a cluster grows in size – from development laptop to a multi-node production deployment. No manual re-sharding or re-balancing is required by any application, and cluster configuration information is all managed behind the scenes by the topology-aware clients.
- **Simplicity and ease of development:** It's easy for developers to work with through the officially supported SDKs that are available for all popular languages (Java, .NET, Python, PHP, Node.js, Go, and C). Rich integration is available via frameworks and components such as Spring Data, Apache Spark, LINQ, and more.

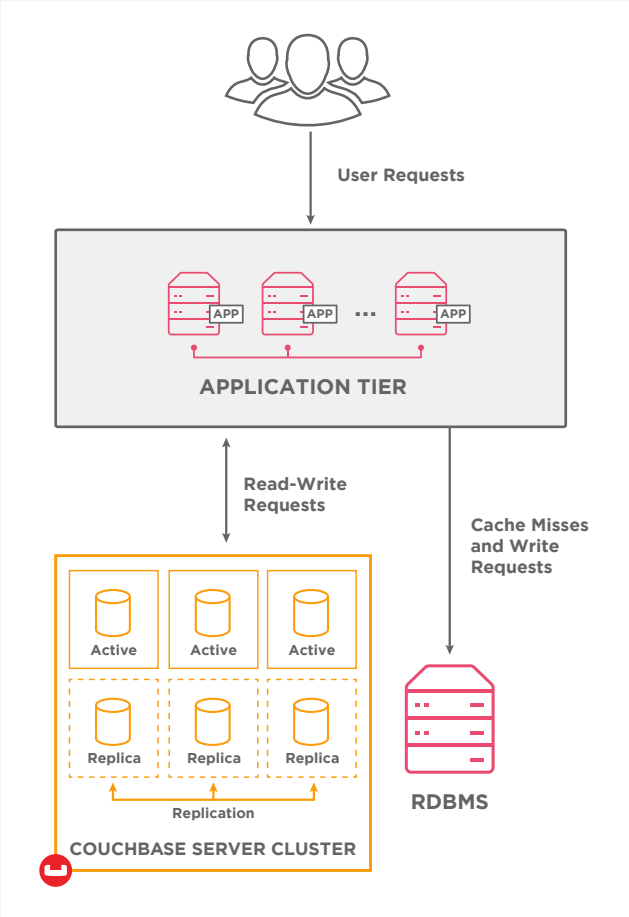

Figure 3: Architecture of deploying a Couchbase Server cluster as a caching layer

Caching and document performance benchmarking

Couchbase supports typical caching use cases, and also supports more challenging document database scenarios as well; in both of these scenarios, it outperforms the competition.

Benchmark analysis has been performed by third-party consulting company Altoros. They ran various benchmarks against Couchbase and other NoSQL products (MongoDB and Dynamo) that are generally not used as caching solutions. Couchbase outperformed these products for best-in-class cache as well as highest performing document database.

Results shown in Figure 4 demonstrate how strongly Couchbase competes with other NoSQL vendors in multiple clustering scenarios. One of the use cases tested was for caching scenarios in particular, with a common high-volume, key-value workload.

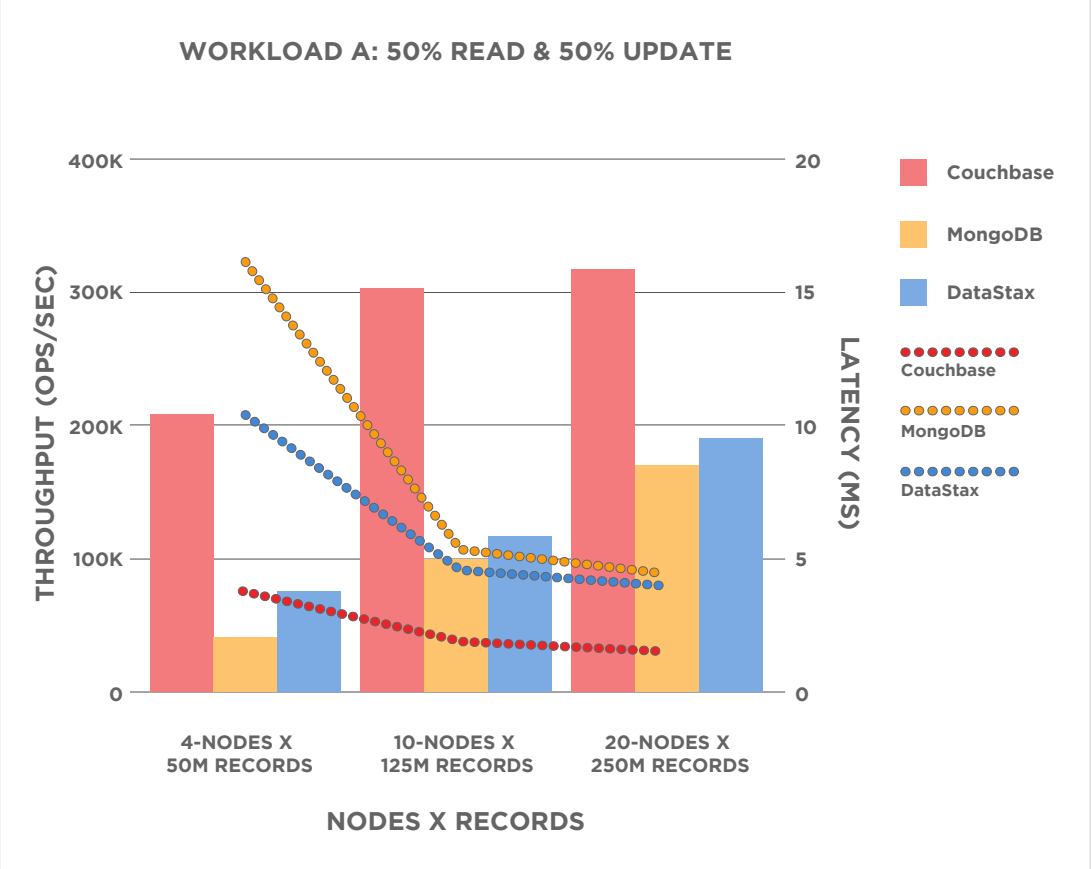

Figure 4: Altoros benchmark comparing Couchbase, MongoDB, and DynamoDB performance with a cached key-value lookup and active read/write workload

In addition to caching, there are other workloads in the benchmark that serve as examples of how Couchbase solves other common scenarios such as serving as a database for an enterprise source of truth or system of record solution.

System of record

Operating as a system of record for enterprise data is another distinct role that Couchbase can serve. In this case, Couchbase operates as both a cache *and* the authoritative primary database for applications, providing the durability and stability that is needed for any primary database application. This is the domain of traditional relational databases but has become increasingly popular for NoSQL databases to address, especially on cloud and web platforms.

There is a natural evolution from caching for a database application to aggregating from other database sources as a system of record and, then, ultimately to moving source databases over into Couchbase. In all cases, key Couchbase features help users easily make those transitions while minimizing risk and unlocking value.

SQL++ is the next-generation query language for managing JSON data using familiar SQL-based syntax.

Key features such as SQL queries, ACID transactions, relational-structure mirroring, full-text searches, and real-time SQL++ analytics across a range of internal sources all factor into building more than just a caching system.

To learn more about how well these types of queries perform on Couchbase, versus other NoSQL products, see the charts, queries, and testing approaches used in benchmark reports at couchbase.com/benchmarks.

Why companies choose Couchbase

Couchbase is a great fit for many caching scenarios. Many leading companies have deployed Couchbase Server for mission-critical applications, including many of the world's leading enterprises:

LinkedIn – With over 300 million members, LinkedIn uses Couchbase to cache over 8 million real-time metrics (over 12TB of data). Over 16 million entries are loaded into Couchbase every 5 minutes.

Marriott – Supporting 6,700 global hotel properties, Marriott moved its reservations system from a relational database to Couchbase. The result: reduced costs while maintaining 30 million documents and 4,000 transactions per second.

Amadeus – Amadeus, the leading provider of travel software and technology solutions for the global travel industry, moved to Couchbase after running Memcached on top of MySQL to maintain high performance. The company now processes 7 million requests per sec. at <2.5 ms response times.

eBay – The world's largest online auction marketplace uses Couchbase to cache over 100 million authentication tokens per day to ensure session validity. eBay achieves over 300,000 writes per second with Couchbase.

So why have these enterprises chosen Couchbase over the alternatives?

Many caching solutions are simple key-value stores with in-memory capabilities and some ability to scale out. Couchbase is instead built from the ground up to deliver elastic performance at scale – the very foundation of a superior caching tier.

In addition, Couchbase builds on this performance with a complete document database. High availability, powerful SQL-based query, native mobile integration, ad hoc analytics, and text search combine to empower enterprises beyond simple caching.

Combined technical advantage

When you combine all the architectural advantages of Couchbase, you have a comprehensive, high-performance NoSQL database platform to build future use cases with. While caching is a great use case to get started, once your data is in Couchbase, there is so much more that is possible.

Other features include SQL querying using the Couchbase NoSQL query language (SQL++) – effectively letting you query JSON data without having to enforce a schema or transform your data to behave a certain way just to get answers to queries.

Advanced real-time analytic queries are also possible – as well as full-text searches. Many developer-centric features exist in Couchbase, including server-side event processing, operation tracing, ACID transactions, scope/collection organization, and automatic application failover between clusters.

These are all features that the most demanding teams require. The remainder of this paper explores these concepts further and contrasts them with other solutions within the overall context of caching solutions.

Couchbase alternatives

Memcached and Redis are two examples of solutions that are part of the broader landscape including both key-value databases and caching solutions. Many other caching-related products exist, including GemFire, Hazelcast, and Oracle Coherence. They attempt to solve similar problems, but do not necessarily aim to be a comprehensive database solution to service caching and other use cases. This paper will focus on Memcached and Redis, however, the architectural considerations apply to all NoSQL databases and caching solutions.

Redis
For businesses using MongoDB, Redis is often recommended as a caching add-on to solve caching-related performance challenges. Redis is a popular data structure server. It runs in-memory and has some snapshot persistence, but is not designed to be a highly persistent database and has limitations around its partitioning model and workload isolation.

Memcached
At the other end of the spectrum, Memcached is a free, open source product that's used in thousands of web, mobile, and IoT applications around the world. It's simple to install and deploy, and it delivers reliable high performance. However, Memcached has no enterprise support available, nor does it include a management console for monitoring. Many companies that deploy Memcached find they want additional capabilities not included in Memcached, such as automatic failover to avoid downtime and automatic rebalance to avoid cold caches.

Couchbase has some shared lineage with Memcached and addresses many of its limitations while also serving as a complete document database solution.

Limitations of Redis

Redis is a key-value data structure server that is popular for in-memory caching solutions. Companies who employ Redis typically use it on top of other products such as MongoDB or MySQL to improve performance. It solves other use cases but is not generally recognized as a document database. Common concerns with Redis include:

- **Complexity** – Redis data can be sharded across several nodes, but scripts and command line utilities have to be run to redistribute data when adding/removing nodes. It also runs in a primary/secondary (historically known as master-slave architecture), where the secondaries are read-only. Couchbase uses a "masterless" approach. Couchbase tasks such as rebalance, adding and failing over nodes, and more can all be done automatically.

- **Lacks built-in features** – As Redis is optimized for key-value lookups, the concept of querying is different than most database users expect. Ad hoc query and indexing is not possible with the core product If applications need a change to the data model, then rehashing of data may be required. Couchbase provides an array of built-in query and indexing services and allows them to run on different nodes – providing powerful workload isolation.

Couchbase Ephemeral Mode

Couchbase automatically persists to disk, enabling larger-than-memory data. However, Couchbase also has a memory-only Ephemeral mode, for situations where you do not ever want to invoke the overhead of disk access.

Couchbase Memcached support

Memcached still appears as an option in Couchbase, for purposes of backwards compatibility. However, it is deprecated and not recommended for any new development.

- **Persistence** – While Redis has the ability to persist data, it is still primarily an in-memory focused layer. The persistence capabilities are designed to back up data and speed up the "cold" restart process, but this impacts performance as it saves its snapshots to disk. It is not designed for real-time storage and swapping of disk or in-memory datasets. Couchbase is a complete database solution, able to efficiently load and persist data from/to disk as expected from a database.

- **Memory limitations** – Redis datasets must fit into memory. This makes it very challenging for larger datasets as they must scale up the machine or scale out the cluster of Redis nodes to shard the data across nodes. Since Redis requires all data to be in memory, Redis does not efficiently support rotating through a hot working set as requests shift over the course of a day. This requires more hardware and increased licensing costs when data volumes start to exceed memory. In contrast, Couchbase can load data that is larger than memory. Memory quotas can be set to determine how much of the dataset is kept in RAM, with most used data being read as needed into the cache for quick access.

Limitations of Memcached

Memcached is a simple, open source cache used by many companies, including YouTube, Reddit, Craigslist, Facebook, Twitter, Tumblr, and Wikipedia. It's an in-memory, key-value store for small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering.

Key advantages of Memcached include:

- **High performance** – Memcached was engineered as a low latency, high throughput, scalable cache. It is capable of delivering the throughput required by very large, internet-scale applications.
- **Simplicity** – Intentionally designed as a pure, bare-bones cache, Memcached is very simple to install and deploy.
- **Low initial cost** – Licensed under the Revised BSD license, Memcached is free open source software.

Lack of enterprise support, built-in management, and advanced features

If you encounter an issue with Memcached, you need to rely on your own resources or the Memcached community. There is no built-in management console, so users need to build their own tools or find separate tooling to monitor its performance.

Memcached does not include advanced features that many enterprises require, such as automatic failover, load rebalancing to add capacity without downtime, and cross data center replication.

Couchbase builds on and extends the strengths of Memcached – including high performance and simplicity – to deliver a more powerful replacement for Memcached, and a less complex, and more powerful alternative to Redis.

Beyond caching with a complete NoSQL solution

Modern applications must run in distributed environments and support millions of users globally with submillisecond response times. Applications employ multiple technologies to meet these requirements in their data layer. Technology choices are influenced by maturity, performance, flexibility needs, and storage requirements. It's not uncommon to have various systems of record (the authoritative data source), wrapped in layers of caches (temporary data storage for high performance). Databases serving as sources of truth, aggregating data from various microservices for a single view, often require their own caches as well.

With its many integrated features, including a built-in managed cache, disk persistence, high availability, geographic replication, structured query language, real-time analytics, full-text search, eventing, and mobile synchronization, Couchbase consolidates multiple layers into a single platform that otherwise would require separate solutions to work together.

How and where you deploy Couchbase is entirely up to you. Some use Couchbase just as a cache or just as a system of record. Others start with Couchbase as a cache and eventually evolve it to become a source of truth and system of record. Regardless of your strategy, Couchbase gives you the flexibility to choose any starting point and easily evolve over time.

Couchbase is able to provide the performance of a caching layer, the flexibility of a source of truth, and the reliability of a system of record. This reduces the need to manage data models and consistency between multiple systems, learn different languages and APIs, and manage independent technologies and the integrations between them.

Many leading enterprises have extended their Couchbase deployment to be the primary data store, across a growing number of solutions including:

- Customer 360
- Catalog & inventory management
- Field service (mobile and edge)
- IoT data management
- Content management
- Mobile data management
- Operational dashboarding
- Product catalog & pricing
- Session store
- Shopping cart
- User profile store

Once you have implemented Couchbase as a cache – and start to experience lower costs, higher performance, improved manageability, and easy scalability – you can start to consider how to leverage the other benefits of the database, including SQL++, full-text search, real-time analytics, server-side eventing, and more.

At Couchbase, we believe data is at the heart of the enterprise. We empower developers and architects to build, deploy, and run their mission-critical applications. Couchbase delivers a high-performance, flexible and scalable modern database that runs across the data center and any cloud. Many of the world's largest enterprises rely on Couchbase to power the core applications their businesses depend on.

For more information, visit www.couchbase.com.