HIGH PERFORMANCE APPLICATIONS WITH DISTRIBUTED CACHING

Get integrated caching from a complete NoSQL solution
Table of Contents

Executive summary 3
Importance of a cache in enterprise architectures 4
    Common use cases 4
Key requirements 5
Distributed caching with Couchbase Server 5
    Architectural advantages 6
        Perform at any scale 6
        Manage with ease 6
        Develop with agility 7
    Caching and document performance benchmarking 8
Why companies choose Couchbase 9
    Combined technical advantage 9
Couchbase alternatives 10
    Limitations of Redis 10
    Limitations of Memcached 11
Executive Summary

For many web, mobile, and Internet of Things (IoT) applications that run in clustered or cloud environments, distributed caching is a key requirement, for reasons of both performance and cost. By caching frequently accessed data in memory - rather than making round trips to the backend database - applications can deliver highly responsive experiences that today’s users expect. And by reducing workloads on backend resources and network calls to the backend, caching can significantly lower capital and operating costs.

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

Caching has become a de facto technology to boost application performance as well as reduce costs.

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 graphical user interfaces 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 wide area cluster networks. Cross datacenter replication is a feature that is either 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
Importance of a cache in enterprise architectures

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

Caching has become a de facto 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 – a distributed, scale-out caching solution that runs on low-cost commodity servers can reduce the need to add expensive resources.

Common use cases

Due to clear performance and cost benefits, 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 were not designed to operate at Internet 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 widely used, cost-effective technique to speed up the backend RDBMS.

- **Managing usage spikes** – Web and mobile applications often experience spikes in usage (for example, seasonal surges like Black Friday and Cyber Monday, or when a video goes viral, etc.). In these and others cases, 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, among others. 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 or real-time recommendation engine on an e-commerce site, or player history in a game.

### Key requirements

The requirements for an effective caching solution are fairly straightforward. Enterprises generally factor six key criteria into their evaluation. How you weight them depends on your specific situation.

1. **Performance**: Performance is the No. 1 requirement. 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? Obviously, less is better.

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 24x365 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 – set it and forget it.

5. **Simplicity**: 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.

6. **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.

### Distributed caching with Couchbase Server

Couchbase Server has become an attractive alternative to Redis and Memcached, as well as other caching products. 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, handle large amounts of data, and provide highly responsive experiences on any device.

For many enterprises, Couchbase Server hits the sweet spot by delivering performance, scalability, and availability, while being easy to deploy and manage. And because it’s open source, Couchbase Server 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 MySpace and Facebook. LiveJournal 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 datacenter replication (XDCR) were not built into the product.

Architectural advantages

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

Perform at any scale

- **Memory and network-centric:** Couchbase Server’s memory-first architecture, with integrated document cache, was purposely designed to deliver sustained high-throughput rates in distributed computing environments while providing consistent submillisecond latency and exceptional resource efficiency. The network-centric architecture with a high-performance replication backbone allows new workloads to be easily added while maintaining performance at scale.

- **Always-on, edge-to-cloud:** Couchbase Server 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 Server 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.

- **Workload isolation and optimization:** Adding or removing nodes can be done in minutes with push-button simplicity, without any downtime or code changes. Couchbase Server’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:** A number of successful enterprises have selected Couchbase Server specifically because of its uniquely simple and powerful active-active cross datacenter replication (XDCR) capabilities that support varying types of replication topologies.

- **Flexible deployment options:** Multiple methods of deployment are supported including hybrid-cloud and Docker containers with the Couchbase Kubernetes/OpenShift Operator.

- **Consistent performance when adding microservices:** Couchbase Server eases management with auto-sharding, replication, and failover for easy scale out and high availability. Autonomously maintain application availability 24x365 across upgrades, node failures, network failures, or even cloud provider failures. All functionality is made available across physical, virtualized, public cloud, container, and Couchbase Managed Service environments.

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.
• **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:** As a competitive open source product, 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. Also, Couchbase Server 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 Redis. And because Couchbase Server is far less complex to deploy and manage, it takes significantly 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 query and analytics to allow developers to use common database skills with Couchbase.

• **Versatile data access patterns:** Couchbase’s comprehensive set of data access methods include key-value retrieval, structured querying, full-text search, real-time analytics and server-side triggers – available across cloud, mobile, and edge devices.

• **No hassle scale out:** Applications built on Couchbase do not change when a cluster grows in size from development laptop to a multi-node production deployment. No manual re-sharding 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:** Couchbase Server’s code is streamlined, compact, and highly efficient with a small footprint. It can be installed and configured in minutes and requires very little time or training to learn. It’s easy for developers to work with through the officially supported SDKs that are available for all common 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 .NET Core.
Caching and document performance benchmarking

Couchbase Server 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 DataStax) 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 DataStax (Cassandra) performance with a cached key-value lookup and active read/write workload](image)

In addition to caching, there are three 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.

Source of truth
At its core, caching requires high performance, scalability, and ease of management. But other types of systems have additional requirements. For example, a database as a source of truth system depends greatly on aggregating information from multiple sources and making it readily available in a single view to the user. Couchbase’s flexible schema and integrated query environment make it possible to go beyond just being a cache to help provide an enterprise system view of all available data. Often this means consolidating data from traditional SQL databases to improve performance or as a mainframe data offload to reduce costs.

System of record
Operating as a system of record for enterprise data is another distinct role that Couchbase serves. In this case, Couchbase operates as 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.

Key features such as SQL-like queries, 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 Server is a great fit for many caching scenarios. More than 400 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 on the other hand is architected 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 search combine to empower enterprises beyond simple caching.

**Combined technical advantage**

When you combine all the architectural advantages of Couchbase you have the most comprehensive, high-performance NoSQL database platform to build on top of. While caching is an obvious use case to get started, once your data is in Couchbase there is so much more that is possible.

Other features include SQL-like querying using the Couchbase NoSQL query language (N1QL) – 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.

More 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, and automatic application failover between clusters.
These are all features that the most demanding enterprises 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 commercial 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, Oracle Coherence, and more. They attempt to solve for similar problems but do not particularly aim to be a comprehensive database solution to service caching or other use cases that a NoSQL offering needs to have. 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 open source data structure server. It runs in-memory and has some 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. And many companies that deploy Memcached find they want additional capabilities not included in Memcached, such as failover to avoid stampeding herds on the system of record or dynamic 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. The main complaints about 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/remove nodes. It also runs in a master-slave architecture, where the slaves are read-only, so it requires you to understand which master nodes are available for read/write operations. Couchbase uses a less complex approach. Developers can easily install without sysadmin help, Couchbase cluster can be up and running in minutes using a web-enabled management console, and tasks such as rebalance, adding and failing over nodes, and more can all be done swiftly with a couple of clicks.

- **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 is not possible with the core product, yet the model used to store the data is crucial to what kind of query patterns are possible. 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 methods and allows different kinds of services to run on different nodes – providing powerful workload isolation.
• **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 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 across geographies. This requires increased hardware and 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 yet powerful, 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. Users can set up Memcached quickly and easily.

• **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**

No commercial entity provides technical support for Memcached. So 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 to monitor its performance.

Memcached does not include advanced features that many enterprises require, such as auto failover to avoid stampeding herds against the system of record, load rebalancing to add capacity without downtime, and cross datacenter replication.

As described below, Couchbase Server 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 places 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. Others still use Couchbase differently for different applications. Regardless of your strategy, Couchbase gives you the flexibility to choose any starting point and easily evolve over time.

Couchbase is uniquely 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 eliminates the need to manage data models and consistency between multiple systems, learn different languages and APIs, and manage independent technologies.

In fact, 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
- IoT data management

For use cases like:
- 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 – with lower costs, high performance, manageability, and elastic scalability – it may be a good time to consider how to leverage the other benefits of the database, including query, text search, real-time analytics, server-side scripting, and more.

---

**About Couchbase**

Couchbase’s mission is to be the data platform that revolutionizes digital innovation. To make this possible, Couchbase created the world’s first Engagement Database. Built on the most powerful NoSQL technology, the Couchbase Data Platform offering includes Couchbase Server and Couchbase Mobile and is open source. The platform provides unmatched agility and manageability – as well as unparalleled performance at any scale – to deliver ever-richer and ever-more-personalized customer experiences.