به نام خدا

دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

تحقیق درس برنامه سازی وب

مقدمه ای بر Hazelcast برای Cache کردن داده

نگارش:

ریحانه اخلاقیان

استاد:

استاد یحیی پورسلطانی

بهار ۱۴۰۳

فهرست

مقدمه

در عصر فناوری اطلاعات و وب، مدیریت کارآمد داده‌ها یکی از اساسی‌ترین نیازها برای توسعه‌دهندگان و مدیران سیستم‌ها است. Hazelcast به عنوان یکی از قدرتمندترین ابزارهای Cache کردن داده مطرح شده است که می‌تواند بهبود چشم‌گیری در عملکرد سیستم‌های وب ایجاد کند. در این تحقیق، به معرفی Hazelcast و کاربردهای آن در زمینه Cache کردن داده می‌پردازیم.

Cache کردن داده

Cache کردن داده به فرآیند ذخیره موقت داده‌های پرکاربرد در یک مکان سریع‌تر و قابل دسترس‌تر (مانند حافظه RAM) گفته می‌شود تا در صورت نیاز مجدد به آن داده‌ها، سیستم بتواند به سرعت به آن‌ها دسترسی پیدا کند و نیاز به بازیابی آن‌ها از منابع کندتر (مانند دیسک‌های سخت یا پایگاه‌های داده) نداشته باشد. این تکنیک به بهبود عملکرد و کارایی سیستم‌های نرم‌افزاری کمک می‌کند و زمان پاسخ‌دهی را کاهش می‌دهد. Cache ها می‌توانند در سطوح مختلفی مانند حافظه پنهان پردازنده (CPU cache)، حافظه پنهان سیستم عامل (OS cache) یا حافظه پنهان برنامه‌های کاربردی (Application cache) پیاده‌سازی شوند. این تکنیک به‌ویژه در سیستم‌های وب که نیاز به پاسخ‌دهی سریع دارند، بسیار مؤثر است.

معرفی Hazelcast

Hazelcast در سال 2008 توسط Talip Ozturk تأسیس شد و به مرور زمان با مشارکت تعداد زیادی از توسعه‌دهندگان به یک ابزار کلیدی در صنعت تبدیل گردید.

Hazelcast یک پلتفرم محاسبات توزیع‌شده و ذخیره‌سازی داده در حافظه (in-memory) است که به توسعه‌دهندگان امکان می‌دهد تا برنامه‌های مقیاس‌پذیر و با کارایی بالا ایجاد کنند. این پلتفرم از ساختارهای داده توزیع‌شده مانند نقشه‌ها (maps)، مجموعه‌ها (sets)، صف‌ها (queues) و موضوعات (topics) پشتیبانی می‌کند و به صورت خوشه‌ای (clustered) عمل می‌کند تا داده‌ها و پردازش‌ها را بین چندین گره (node) توزیع کند. Hazelcast برای کاربردهای متنوعی از جمله caching، مدیریت جلسات (session management)، پردازش جریان (stream processing) و هماهنگ‌سازی داده‌ها (data synchronization) قابل استفاده است. به دلیل معماری توزیع‌شده و حافظه‌محور خود، Hazelcast بهبود قابل توجهی در زمان پاسخ‌دهی و مقیاس‌پذیری برنامه‌ها فراهم می‌کند.

در نتیجه ویژگی‌های کلیدی Hazelcast عبارت اند از:

- مقیاس‌پذیری بالا
- پشتیبانی از داده‌های توزیع‌شده
- سادگی در راه‌اندازی و استفاده

معماری Hazelcast

معماری Hazelcast بر پایه یک خوشه توزیع‌شده (distributed cluster) از گره‌ها (nodes) است که به صورت همتا به همتا (peer-to-peer) با یکدیگر ارتباط دارند. هر گره در خوشه Hazelcast می‌تواند به عنوان یک عضو فعال عمل کند که داده‌ها و وظایف پردازشی را ذخیره و مدیریت می‌کند. داده‌ها به صورت خودکار بین گره‌ها تقسیم و تکثیر می‌شوند تا از دسترسی بالا و تحمل خطا اطمینان حاصل شود. Hazelcast از ساختارهای داده توزیع‌شده مانند نقشه‌ها (maps)، مجموعه‌ها (sets)، صف‌ها (queues) و موضوعات (topics) پشتیبانی می‌کند و از پروتکل‌های مختلفی برای ارتباط بین گره‌ها استفاده می‌کند. این معماری به Hazelcast اجازه می‌دهد تا به راحتی مقیاس‌پذیری افقی (horizontal scalability) را فراهم کند، به طوری که با اضافه کردن گره‌های بیشتر به خوشه، توان پردازشی و ظرفیت ذخیره‌سازی افزایش می‌یابد. مدل داده Hazelcast بر اساس نقشه توزیع‌شده طراحی شده است که امکان جستجو و دسترسی سریع به داده‌ها را فراهم می‌کند.

به صورت خلاصه مبانی و اجزای معماری Hazelcast به صورت زیر است:

- Cluster
- Member
- Client

کاربردهای Hazelcast در Cache کردن داده

یکی از مهم‌ترین کاربردهای Hazelcast در بهبود عملکرد وب‌سرورها و دیتابیس‌ها است. با استفاده از Hazelcast می‌توان به دسترسی سریع‌تر به داده‌ها و کاهش بار بر روی دیتابیس دست یافت.

a) بهبود سرور ها:

۱) session های کاربران (session replication): Hazelcast می‌تواند داده‌های مربوط به session های کاربران (HTTP sessions) را بین سرورهای مختلف به اشتراک بگذارد. این کار باعث می‌شود که حتی در صورت از کار افتادن یک سرور، داده‌های session کاربران از بین نروند و به سرور دیگری منتقل شوند.

۲)کاهش بار سرورها (load balancing): با استفاده از Hazelcast، درخواست‌ها بین چندین سرور توزیع می‌شوند. این کار باعث می‌شود که هیچ یک از سرورها بیش از حد بارگیری نشوند و از تمامی منابع بهینه استفاده شود.

b) بهبود دیتابیس ها:

۱) کاهش تعداد کوئری‌ها به دیتابیس: با cache کردن نتایج کوئری‌ها در Hazelcast، نیاز به ارسال کوئری‌های مشابه به دیتابیس کاهش می‌یابد. این کار فشار روی دیتابیس را کاهش داده و زمان پاسخ‌گویی را بهبود می‌بخشد.

۲) پایدارسازی کوئری‌های پرهزینه: کوئری‌هایی که زمان اجرای زیادی دارند یا منابع زیادی مصرف می‌کنند، می‌توانند در Hazelcast، cache شوند. بنابراین در درخواست‌های بعدی، نیازی به اجرای دوباره این کوئری‌ها نیست.

۳) جلوگیری از ازدحام در دیتابیس (throttling) : Hazelcast به کاهش ازدحام و transaction های تکراری کمک می‌کند از این رو دیتابیس‌ها کمتر در معرض فشار حجم بالا قرار می‌گیرند.

c) قابلیت‌های Hazelcast برای بهبود عملکرد:

۱) مقیاس‌پذیری افقی (horizontal scalability): با اضافه کردن گره‌های بیشتر به خوشه‌ی Hazelcast، توان محاسباتی و ظرفیت حافظه را به راحتی می‌توان افزایش داد. این کار به بهینه‌سازی عملکرد سرورها کمک می‌کند.

۲) همگام‌سازی داده‌ها و transaction های توزیع‌شده: Hazelcast با پشتیبانی از نقل و انتقالات توزیع‌شده (distributed transactions) و هماهنگ‌سازی داده‌ها، امکان مدیریت داده‌ها در محیط‌های توزیع‌شده را فراهم می‌کند.

مقایسه Hazelcast با سایر ابزارهای Cache

انتخاب ابزار مناسب برای cache کردن داده‌ها بستگی به نیازهای خاص پروژه دارد. اگر به دنبال یک راه‌حل جامع و توزیع‌شده با قابلیت‌های پردازش داده‌های پیچیده باشیم، Hazelcast یا Apache Ignite گزینه‌های مناسبی هستند. اگر سرعت و سادگی مورد نیاز است، Redis و Memcached می‌توانند انتخاب‌های بهتری باشند.

۱. مقایسه Hazelcast و Redis:

Hazelcast بیشتر برای کاربردهای پیچیده‌تر و پردازش داده‌های توزیع‌شده مناسب است، در حالی که Redis به دلیل سادگی و سرعت بالا بیشتر برای cache کردن داده‌ها و مدیریت session های کاربران استفاده می‌شود.

۲. مقایسه Hazelcast و Memcached:

Hazelcast قابلیت‌های بیشتری مانند تراکنش‌ها و پردازش داده‌های توزیع‌شده دارد، در حالی که Memcached بسیار سبک و سریع است و برای cache کردن داده‌های موقت مناسب است.

۳. مقایسه Hazelcast و Apache Ignite:

هر دو ابزار قابلیت‌های مشابهی دارند، اما Apache Ignite امکانات بیشتری برای پردازش داده‌های توزیع‌شده و پشتیبانی از SQL دارد.

نتیجه‌گیری

در این تحقیق، به بررسی استفاده از Hazelcast برای cache کردن داده‌ها پرداختیم Hazelcast .به عنوان یک ابزار توزیع‌شده و همتا به همتا (peer-to-peer) با قابلیت‌های پیشرفته‌ای مانند نقشه‌های توزیع‌شده، صف‌ها، مجموعه‌ها و موضوعات توزیع‌شده، پشتیبانی از تراکنش‌های توزیع‌شده و قابلیت تحمل خطا، انتخاب مناسبی برای بهبود عملکرد سیستم‌ها و کاهش بار روی دیتابیس‌ها می‌باشد.

استفاده از Hazelcast به ما این امکان را می‌دهد که داده‌ها را به صورت موازی پردازش کنیم و از مقیاس‌پذیری افقی برای افزایش کارایی سیستم بهره‌مند شویم. به علاوه، این ابزار با پشتیبانی از زبان برنامه‌نویسی Java وAPI های قدرتمند، توسعه و پیاده‌سازی را برای برنامه‌نویسان آسان‌تر می‌کند.

در مقایسه با سایر ابزارهای cache کردن داده‌ها مانندRedis ، Memcachedو Apache Ignite ، Hazelcastبا ارائه قابلیت‌های بیشتر و پشتیبانی از پردازش داده‌های توزیع‌شده، یک راه‌حل جامع و کارآمد برای پروژه‌های پیچیده‌تر است. با این حال، انتخاب نهایی بستگی به نیازهای خاص پروژه دارد و در برخی موارد، ابزارهای ساده‌تر و سریع‌تر مانند Redis یا Memcached می‌توانند مناسب‌تر باشند.

در نهایت، استفاده از Hazelcast برای cache کردن داده‌ها می‌تواند به طور قابل توجهی عملکرد سیستم‌ها را بهبود بخشد، زمان پاسخ‌دهی را کاهش دهد و تجربه کاربری بهتری را فراهم کند. با توجه به مستندات رسمی و منابع موجود، پیاده‌سازی و پیکربندی Hazelcast نیز به سادگی امکان‌پذیر است و می‌تواند به عنوان یک ابزار قدرتمند در معماری‌های توزیع‌شده مورد استفاده قرار گیرد.

منابع

۱. وب‌سایت رسمی Hazelcast (https://hazelcast.com/why-hazelcast/)

۲. مستندات Hazelcast در GitHub: https://github.com/hazelcast/hazelcast

۳. https://docs.hazelcast.com/hazelcast/latest/architecture/architecture