به نام خدا

دانشگاه شهید بهشتی

گزارش پروژه پایانی درس آزمون نرم‌افزار

استاد درس: فائزه گوهری

ارائه دهنده: علی نوروزیان

زمستان ۱۴۰۳

عنوان پروژه: ارزیابی کارایی سیستم‌های مبتنی بر صف (RabbitMQ) در بهبود Availability

۱. مقدمه و هدف از پروژه

هدف اصلی این پروژه، بررسی تأثیر استفاده از صف پیام (RabbitMQ) در بهبود کارایی و Availability سیستم‌های مبتنی بر دیتابیس‌های MongoDB و SQL Server (دیتابیس های SQL در مقابل NoSQL) است. در این پروژه، دو سناریوی کلیدی بررسی شد:

- عملکرد سیستم **بدون استفاده از صف** و مواجهه با محدودیت‌های منابع.
- عملکرد سیستم **با استفاده از صف** برای مدیریت ترافیک بالا و جلوگیری از از دست رفتن درخواست‌ها.

با شبیه‌سازی بارکاری سنگین و تحلیل نتایج، نشان داده شد که استفاده از صف چگونه می‌تواند Availability سیستم را افزایش دهد.

۲. معرفی کامل برنامه مورد آزمون

برنامه توسعه‌یافته یک سرویس ساده مبتنی بر C# است که عملیات افزایش یک شمارنده در دیتابیس را انجام می‌دهد. (سناریو ساده در نظر گرفته شده تا از مبحث اصلی دور نشویم) این سرویس در دو معماری متفاوت پیاده‌سازی شد:

- **معماری مستقیم (بدون صف):** درخواست‌ها مستقیماً به دیتابیس (MongoDB یا SQL Server) ارسال می‌شوند.
- **معماری مبتنی بر صف (RabbitMQ):** درخواست‌ها ابتدا در صف ذخیره شده و سپس به صورت ناهمگام پردازش می‌شوند.

محدودیت منابع: با استفاده از Docker، منابع دیتابیس‌ها (CPU و RAM) کاهش داده شد تا رفتار سیستم در شرایط بحرانی شبیه‌سازی شود.

۳. حوزه مورد آزمون

حوزه تمرکز این پروژه، **آزمون کارایی (Performance Testing)** با تأکید بر موارد زیر بود:

- بررسی تأثیر استفاده از صف در کاهش خطاهای ناشی از محدودیت منابع.
- مقایسه کارایی MongoDB و SQL Server در شرایط بارکاری بالا.
- اندازه‌گیری معیارهای کلیدی مانند **تعداد درخواست‌های موفق (RPS)**، **Latency** و **خطاهای اتصال**.

۴. ابزار مورد استفاده

۴.۱. Nbomber

- **نوع ابزار:** چارچوب متن‌باز برای تست بار و کارایی در پلتفرم .NET.
- **قابلیت‌ها:**
  - شبیه‌سازی هزاران کاربر همزمان.
  - اندازه‌گیری دقیق Latency، RPS و خطاها.
  - پشتیبانی از گزارش‌های آماری و نمودارها.
- **نحوه استفاده:** سناریوهای تست با تعریف نرخ درخواست‌ها (۱۰۰۰ RPS) و مدت زمان اجرا (۶۰ ثانیه) پیاده‌سازی شدند.

۴.۲. RabbitMQ

- **نقش:** سیستم صف‌بندی پیام برای جداسازی درخواست‌ها از پردازش واقعی.
- **قابلیت‌ها:**
  - تضمین تحویل پیام‌ها (No Data Loss).
  - مدیریت ترافیک با استفاده از الگوی Producer-Consumer.

۴.۳. MongoDB و SQL Server

- **نقش:** ذخیره‌سازی داده‌ها و پردازش درخواست‌ها.
- **تنظیمات Docker:** محدودیت منابع برای شبیه‌سازی شرایط بحرانی.

۵. طرح آزمون (Test Plan)

بر اساس استاندارد **IEEE 829**، طرح آزمون شامل موارد زیر است:

۵.۱. اهداف آزمون

- ارزیابی کارایی سیستم در شرایط بارکاری بالا.
- بررسی Availability با/بدون صف.

۵.۲. استراتژی آزمون

- **سناریو ۱:** ارسال مستقیم درخواست‌ها به دیتابیس (بدون صف).
- **سناریو ۲:** استفاده از RabbitMQ به عنوان واسط.

۵.۳. محیط آزمون

- **سخت‌افزار:** سیستم با 8GB RAM و CPU 4 Core.
- **نرم‌افزار:** Docker, .NET 9, Nbomber 4.

۵.۴. معیارهای سنجش:

- نرخ موفقیت درخواست‌ها (Success Rate)
- نرخ شکست درخواست‌ها (Failure Rate)
- میانگین تأخیر پاسخگویی (Mean Latency)
- نرخ پردازش درخواست‌ها در ثانیه (Requests per Second - RPS)

۵.۵. معیارهای موفقیت

- کاهش خطاها به صفر در معماری مبتنی بر صف.
- افزایش RPS به ۱۰۰۰ درخواست بر ثانیه.

۶. سند موارد آزمون (موجود در فایل Reports.txt)

۷. سند نتایج اجرای آزمون

۷.۱. MongoDB بدون صف

- **درخواست موفق:** 34,981 (RPS: 874.5)
- **خطاها:** 5,009 (اتصال به دیتابیس قطع شد).
- **Latency میانگین:** 625ms.

۷.۲. MongoDB با صف

- **درخواست موفق:** 60,000 (RPS: 1000)
- **خطاها:** 0.
- **Latency میانگین:** 0.01ms.

۷.۳. SQL Server بدون صف

- **درخواست موفق:** 78 (RPS: 4.9)
- **خطاها:** 13,674 (Timeout اتصال).
- **Latency میانگین:** 55,823ms.

۷.۴. SQL Server با صف

- **درخواست موفق:** 59,995 (RPS: 999.9)
- **خطاها:** 0.
- **Latency میانگین:** 0,01ms.

۸. جمع‌بندی

تجربیات موفق:

- استفاده از RabbitMQ باعث حذف کامل خطاها و دستیابی به 100% Availability شد.
- MongoDB در معماری بدون صف عملکرد بهتری نسبت به SQL Server داشت (RPS بالاتر).

مشکلات و راهکارها:

- **مشکل:** تنظیمات اولیه RabbitMQ برای اطمینان از تحویل پیام‌ها زمان‌بر بود.
- **راهکار:** استفاده از سیاست‌های Retry و Acknowledgment.
- **مشکل:** Nbomber در گزارش‌دهی جزئیات خطاها محدودیت داشت.
- **راهکار:** ادغام با ابزارهای مانیتورینگ مانند Grafana.

پیشنهادات:

- انجام آزمون‌های طولانی‌مدت (Stress Testing) برای بررسی پایداری سیستم.
- استفاده از دیتابیس‌های هیبریدی (مانند Redis) برای بهبود کارایی.

ضمیمه: فایل‌های پروژه شامل کدها، تنظیمات Docker و خروجی‌های Nbomber در قالب ZIP پیوست شده است.

منابع:

- Molyneaux, I. (2009). The Art of Application Performance Testing. O’Reilly Media.
- Knott, D. (2015). Hands-On Mobile App Testing. Addison-Wesley Professional.
- IEEE Std 829-2008. IEEE Standard for Software and System Test Documentation.
- MongoDB Performance Best Practices: [docs.mongodb.com](https://www.mongodb.com/docs/manual/administration/optimization/)
- RabbitMQ Performance Measurements: [rabbitmq.com](https://www.rabbitmq.com/blog/category/performance/)
- Nbomber Documentation: nbomber.com
- Docker Documentation: docs.docker.com