زبان‌های Orchestration و Choreography

علیرضا سلطانی نشان

۱۲ آذر ۱۴۰۳

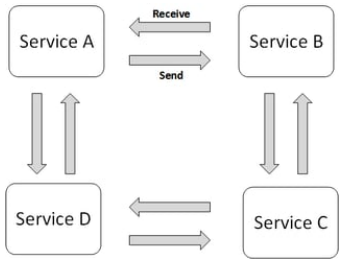
۱ Service Choreography

در این مدل از مدیریت سرویس‌ها، هر سرور از وظیفه خود آگاه است و براساس قوانین و ارتباطاتی که دارد به صورت مستقیم به سرور و سرویس‌های دیگر متصل است و با یکدیگر کار می‌کنند. دقیقاً همانند اعضای بدن که هر کدام وظیفه مشخصی دارند ولی با همکاری با یکدیگر می‌توانند به یک هدف مشترک برسند و در صورت عدم حضور صحیح یکی از اعضا عملکرد تمام اعضا تحت تاثیر قرار می‌گیرد. در این مدل هیچ Central control حضور ندارد و سرویس‌ها براساس رخدادها یا Event با یکدیگر تعامل دارند [۱]. از مزیت استفاده از این مدل می‌توان به موارد زیر اشاره کرد:

- عدم کنترل متمرکز: این کار باعث می‌شود تا مقیاس‌پذیری و تحمل خطا افزایش پیدا کند چرا که هیچ گونه Single point of failure در آن وجود ندارد که با خراب شدن سیستم مرکزی عملکرد کل سیستم مختل شود و سیستم در حالت بلاتکلیف قرار گیرد.
- این روش برای معماری‌های مبتنی بر رخداد بهطور مناسب کار می‌کند. این مدل کاملاً برای سیستم‌هایی که نیاز به پردازش Real-time دارند مناسب است.

معایب استفاده از این مدل عبارت‌اند از:

- پیچیدگی در مدیریت، زمانی که اعضای حاضر در سیستم افزایش یابد مدیریت تعاملات سیستم‌ها با یکدیگر دشوارتر می‌شود.
- این مدل همواره با چالش‌های پایش و debugging رو به رو است.



شکل ۱: عدم وجود سرویس مرکزی و تعامل سرویس‌ها به صورت Peer to peer [۲]

مثال کاربردی

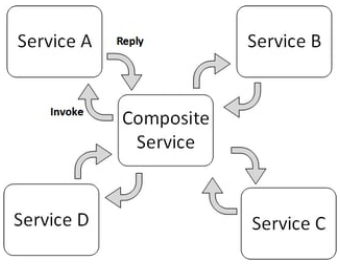
در فروشگاه‌های هوشمند که تمام سیستم‌ها به صورت مستقل وظایف مشخصی دارند. هر سرویسی به سرویس دیگر به صورت مرحله‌ای نیازمند است. تصور کنید که مشتری محصولی را انتخاب کرده است، سفارش محصول به سمت سرویس انبارداری منتقل می‌شود تا مشخص شود از این محصول به تعداد درخواست شده موجودی کالا بررسی شود. بعد از بررسی سرویس انبارداری به سرویس بررسی سبد محصولات هدایت می‌شود تا خرید و سفارش خود را نهایی کند. سپس بعد از نهایی سازی سفارشات به سرویس پرداخت منتقل می‌شود و در صورتی که پرداخت موفقیت‌آمیزی داشته باشد سرویس پیک برای ارسال محصول به مشتری صدا زده می‌شود. این سرویس‌ها در سیستم فروشگاه هوشمند تماماً براساس رخدادهایی کار می‌کنند که توسط سرویس‌های پیشین صدا زده می‌شوند.

۲ Orchestration Service

در این مدل یک Central orchestrator وجود دارد که وظیفه مدیریت تعاملات بین سرویس‌ها را به طور مرکزی دارد. یعنی اگر سیستم مرکزی نباشد هیچ تسکی بین سرویس‌ها توزیع نمی‌شود و سرویس‌ها با هدر رفت منابع رو به رو خواهند بود [۱]. از مزیت این مدل تعامل می‌توان به موارد زیر اشاره کرد:

- کنترل مرکزی: که دائما ویژگی Single point of control را ارائه می‌دهد پیاده‌سازی راحت‌تری را نسبت به مدل تعامل قبلی ارائه می‌دهد.
- کنترل خطای بهتر: یک Ochestrator می‌تواند به صورت بهینه‌تری Retryها، Rollbackها و جریان‌های جایگزین را مدیریت کند چرا که از بالا دستورات به صورت سلسله مراتبی و درختی به نودهای (سرویس‌های) پایینی منتقل می‌شود و می‌توان وضعیت هر کدام از سرویس‌ها را از بالا پایش و Debug کرد.
- مدیریت بهتر گزارش‌ها: در مدل تعامل Orchestration لاگ‌اندازی و پایش تمام فرایندها بهتر از مدل Choreography می‌باشد و می‌توان از طریق این فرایند وضعیت بعدی سیستم را پیش‌بینی کرد.

مهم‌ترین عیب این مدل تعامل وجود سیستم مرکزی برای مدیریت سرویس‌ها می‌باشد چرا که ممکن است با هر Single point of failure کل سیستم Down شود زیرا سرویس‌های زیرین تماماً تحت تاثیر دستورات سیستم مرکزی هستند و اینکار باعث می‌شود سرویس‌ها تسک‌های خود را دریافت نکنند و عملاً فرایندی شروع نشود. یا اگر در حین کار سیستم مرکزی از کار افتد سرویس‌های زیرین در وضعیت بلاتکلیف باقی می‌مانند و نمی‌توانند ادامه فرایند را پیش روند.



شکل ۲: تعامل سیستم‌ها ابتدا توسط سیستم مرکزی اتفاق می‌افتد زیرا از سیستم مرکزی تسک خود را دریافت می‌کنند [۲].

همانطور که آقای Karel Husa [۳] در استک اورفلو گفته است، ما نیازی به تفاوت قائل شدن بین دو مدل تعامل و ارتباطات Orchestration و Choreography نداریم، نیازمندی اصلی ما تمرکز روی منطق کسب و کار

سیستم است. به عبارت دیگر، انتخاب میان این دو مدل به نوع مسئله و نیازمندی‌های آن بستگی دارد نه به یک تعریف تئوریک. مدل Orchestration به دلیل ماهیت تمرکز، در دنیای IT کاربرد بیشتری دارد، در حالی که Choreography معمولاً به عنوان یک موضوع تحقیقاتی باقی می‌ماند اما ممکن است به صورت غیرآگاهانه در برخی سیستم‌ها مورد استفاده قرار گیرد.

چرا Choreography از نظر توسعه‌دهندگان تنها در آکادمیک باقی می‌ماند؟

دلیل اصلی آن این است که برای ایجاد هماهنگی در سیستم‌هایی که به صورت غیرمتمرکز هستند تلاش بسیار زیادی باید صورت گیرد تا به نتیجه مورد نظر برسد و در بسیاری از سرویس‌های امروزی، ما بدون اینکه بدانیم مدل تعامل Choreogprahy چه چیزی می‌باشد به صورت غیرمتمرکز سرویس‌های خود را با یکدیگر هماهنگ‌سازی می‌کنیم.

۳ Architecture Definition Languages

ADLها نقش اساسی در طراحی و مشخص کردن رفتار سیستم‌های توزیع شده دارند، علل خصوص در مدل‌های Orchestration و Choreography:

۱.۳ زبان‌های Orchestration

۱. زبان WS-BPEL یا Web Services Business Process Execution Language: یک زبان رسمی برای مشخص کردن المان‌ها و روابط آن‌ها در سرویس‌های وب در مدل Orchestration بکار می‌رود. در حین پیاده‌سازی ممکن است با پیچیدگی‌هایی رو به رو شویم.

۲. UML یا Unified Modeling Language: با استفاده از این زبان می‌توان برای مدل‌های Orchestration با استفاده از Activity diagram روابط بین المان‌ها را مشخص کرد.

۲.۳ زبان‌های Choreography

۱. WS-CDL یا Web Services Choreography Description Language: به صورت مشخص روی تعریف مدل Choreography وب سرویس‌ها مورد استفاده قرار می‌گیرد.

۲. Pi-Calculus: از یک زبان ریاضی برای تعریف سیستم‌های موبایل و روابط داخلی آن‌ها استفاده می‌کند. یک زبان رسمی برای استدلال در مورد سیستم‌های همروند ارائه می‌دهد.

مراجع

[1] Ghosh, Siladitya. Choreography vs orchestration in microservices. https://medium.com/@siladityaghosh/choreography-vs-orchestration-in-microservices-6ff4b857cf9a, Jul 28, 2024.

[2] Anderi, Benny Bottema &. Choreography vs orchestration. https://stackoverflow.com/questions/4127241/orchestration-vs-choreography, Apr 22, 2015.

[3] Husa, Karel. Why we do not need to distinguish between orchestration and choreography? https://stackoverflow.com/a/39364553/12158825, Sep 7, 2016.