بسم الله الرحمن الرحیم

الگوی معماری رویداد محور

استاد درس : دکتر علی کریمی

دانشجو : مهدی عبدنیا

فهرست مطالب

[مقدمه 4](#_Toc93099359)

[**الگوی معماری رویداد محور** 5](#_Toc93099360)

[تعریف معماری رویداد محور 5](#_Toc93099361)

[تعریف رویداد 5](#_Toc93099362)

[اجزای اصلی تشکیل دهنده معماری رویداد 5](#_Toc93099363)

[ویژگی های الگوی معماری رویداد محور 6](#_Toc93099364)

[معماری رویداد محور چگونه کار می‌کند ؟ 7](#_Toc93099365)

[چه زمان‌هايي از معماري رويداد محور استفاده مي‌شود؟ 8](#_Toc93099366)

[سبك‌هاي مختلف الگوي معماري رويداد محور 9](#_Toc93099367)

[مزایای معماری رویداد محور 10](#_Toc93099368)

[معایب معماری رویداد محور 12](#_Toc93099369)

[نتیجه‌گیری 13](#_Toc93099370)

[پیوست 14](#_Toc93099371)

[مراجع: 16](#_Toc93099372)

# مقدمه

امروزه در معماري نرم‌افزار الگوهاي معماري متنوعي پديد آمده‌اند كه بهبود فرآيند معماري نرم‌افزار كمك مينمايند. يكي از الگوهاي معماري رايج در معماري نرم‌افزار الگوي معماري رويداد محور [[1]](#footnote-1) که به طور خلاصه معماری رویداد محور نامیده می‌شود . در ادامه اين تحقيق ابتدا به تعريف اين معماري پرداخته مي‌شود . سپس اجزاء مختلف معماري رويداد محور مشاهده مي‌گردد . در بخش بعدي به كاربرد اين الگوي معماري پرداخته مي‌شود . سپس مزايا و معايب معماري رويداد محور توضيح داده شده است . و در بخش آخر نكاتي كه بايد درحين استفاده از اين الگوي معماري بايد مورد توجه قرار گيرد پرداخته مي‌شود .

1. فزار ‌ا

# **الگوی معماری رویداد محور**

## تعریف معماری رویداد محور

معماری رویداد محور یک رویکرد طراحی مدرن است که حول داده‌هایی متمرکز است که رویدادها را توصیف می‌کنند[8] . بسیاری از طراحی‌های اپلیکیشن‌های مدرن رویداد محور هستند ، مانند چارچوب‌های تعامل با مشتری که باید از داده‌های مشتری به صورت بلادرنگ استفاده کنند . اپلیکیشن های رویداد محور می توانند با تمام زبان‌های برنامه‌نویسی به وجود بیایند زیرا معماری رویداد محور یک رویکرد است نه یک زبان . معماری رویداد محور حداقل میزان جفت‌شدگی را ممکن می‌سازد،که آن را به گزینه خوبی برای معماری‌های کاربردی مدرن و توزیع شده تبدیل می کند[4].

## تعریف رویداد

یک رویداد را می توان به عنوان "چیزی که اتفاق افتاده است یا تغییر قابل توجهی در وضعیت" توصیف کرد. رویدادها تغییرناپذیرند، یعنی نمی‌توان آن‌ها را تغییر داد یا حذف کرد و به ترتیب ترتیب وقوعشان مرتب می‌شوند[5].

## اجزای اصلی تشکیل دهنده معماری رویداد

معماری رویداد محور دارای سه جزء اصلی است که عبارتند از :

1**-تولییدکننده رویداد**

تولییدکننده‌های رویدادها درالگوی معماری رویداد محور خالق رویدادها و منتشرکننده های آن ها هستند . هر چیزی که می‌تواند یک رویداد تولید کند، تولیدکننده در نظر گرفته می‌شود، خواه یک حسگر مبتنی بر سخت‌افزارباشد ، یا یک درخواست HTTP، یا فرآیندهای تجاری و یا هر شکل دیگری از برنامه کاربردی که می‌تواند رویداد جدیدی را در معماری ما معرفی کند. [7].

**2-‌‌ مسیریاب رویداد**

مسیریاب رویداد به عنوان صف رویداد یا گذرکاه پیام عمل می‌کند که رویدادها را قبل از ارسال به مصرف‌کنندگان رویداد ، آن‌ها را به طور موقت ذخیره ، فیلتر، پردازش اولیه و مسیریابی می كند[9,5] .

**3- مصرف‌کنندگان رویداد**

نقش مصرف‌کنندگان رویداد پردازش و اجرای رویدادهایی است که توسط تولییدکنندگان رویداد به وجود آمده‌اند.

نمونه‌های متداول مصرف‌کنندگان رویداد می‌توانند يك اعلان خودكار اسلك ،يك تابع براي تجزيه و تحليل و يا دستكاري داده‌ها ، برنامه‌اي براي توليد و ارسال ايميل‌ها براي خبرنامه‌ها و يا ورود به يك سيستم باشد [9].

## ویژگی های الگوی معماری رویداد محور

ویژگی های اصلی در معماری رویداد محور عبارتند از :

**1-‌ ارتباطات چندبخشی**

تولییدکنندگان این قابلیت را دارند که رویدادها را به مصرف‌کنندگان متعددی که در یک رویداد مشترک شده اند ، ارسال کنند . این برخلاف ارتباطات تک‌بخشی است که در آن یک فرستنده می‌تواند داده‌ها را فقط به یک گیرنده ارسال کند[10] .

**2- انتقال بلادرنگ**

تولییدکنندگان رویدادها را در صورت وقوع و به صورت بلادرنگ برای مصرف کنندگان منتشر می کنند. نحوه پردازش یا انتقال به جای پردازش دسته ای به صورت بلادرنگ است [10].

**3- ارتباطات ناهمزان**

تولییدکنده منتظر نمی ماند تا مصرف‌کننده یک رویداد را قبل از ارسال رویداد بعدی پردازش کند [10].

4- **ارتباطات ریزدانه**

تولییدکنندگان رویداد به جای اینکه منتظر یک رویداد تجمعی باشند تا آنرا منتشر کنند اقدام به انتشار رویدادهای ریزدانه تکی می‌نمایند [10] .

## معماری رویداد محور چگونه کار می‌کند ؟

همانطور که توضیح داده شد معماری رویداد محور از سه جزء تولیدکننده رویداد ، مصرف‌کننده رویداد و مسیریاب رویداد تشکیل شده است . یک تولییدکننده رویداد یک رویداد را تشخیص یا حس می‌‌كند و رخداد را به عنوان يك پيام نشان مي‌دهد . تولييد‌كننده رويداد اطلاعي از مصرف‌كننده رويداد يا نيتجه يك رويداد را تشخیص نمی‌دهد .

پس از مرحله شناسایی ، یک رویداد از تولیدکننده رویداد از طریق مسیریاب رویداد به مصرف کنندگان رویداد منتقل می شود، جایی که یک پلت فرم پردازش رویداد ، رویداد را به صورت ناهمزمان پردازش می کند. مصرف کنندگان رویداد باید در صورت وقوع یک رویداد مطلع شوند. آنها ممکن است رویداد را پردازش کنند یا فقط تحت تأثیر آن قرار گیرند.

پلتفرم پردازش رویداد پاسخ صحیح به یک رویداد را اجرا می کند و فعالیت را در یک جریان سرازیری به مصرف کنندگان مناسب ارسال می کند. این فعالیت جریان سرازیری جایی است که نتیجه یک رویداد دیده می شود.

یک از مشهورترین پلتفرم‌های جریان‌داده توزیع شده آپاچی کافکا است که یک انتخاب محبوب برای پردازش رویداد است. این پلتفرم می تواند انتشار، اشتراک، ذخیره و پردازش جریان های رویداد را به صورت بلادرنگ انجام دهد. آپاچی کافکا طیف وسیعی از موارد کاربرد را پشتیبانی می کند که در آن توان عملیاتی و مقیاس پذیری بالا حیاتی است و با به حداقل رساندن نیاز به ادغام نقطه به نقطه برای به اشتراک گذاری داده در برنامه های خاص، می تواند تأخیر را به میلی ثانیه کاهش دهد.

میان افزارهای مدیریت رویداد دیگری نیز در دسترس هستند که می‌توانند به عنوان یک پلتفرم جریان رویداد مورد استفاده قرار گیرند[3] .

## چه زمان‌هايي از معماري رويداد محور استفاده مي‌شود؟

در موارد زیر از معماری رویداد محور استفاده میشود که به بیان آن‌ها پرداخته شده است .

1. **تجمیع سیستم‌های ناهمگن بدون جفت‌شدگی**

اگر سیستم هایی موجود باشند که روی پشته‌های مختلف اجرا می‌شوند ،آن هم بدون اینکه از هم اطلاع داشته باشند . این قابلیت موجود است که از معماری رویداد محور برای به اشتراک گذاشتن اطلاعات استفاده کرد بدون اینکه نیازی به استفاده از جفت‌شدگی[[2]](#footnote-2) باشد .مسیریاب رویداد به طور غیر مستقیم قابلیت همکاری را در بین سیستم ها ایجاد می‌کند ، بنابراین آنها می‌توانند پیام‌ها و داده‌ها را در حالي كه ناشناس باقي مي‌مانند ، مبادله كنند [2].

**2-‌‌‌‌‌‌‌ ‌ گنجایش خروجی بالا**

اگر سیستم‌های زیادی وجود دارند که به عنوان مصرف‌کننده برای پاسخ‌دهی به یک رویداد کار می‌کنند ، می‌شود که از معماری رویداد محور استفاده کرد تا دیگر نیاز نباشد تا برای ارسال رویداد به هر مصرف کننده کد سفارشی و جداگانه‌ای نوشت . آن هم بدین‌صورت که مسیریاب رویداد ،رویداد را به هر یک از مصرف‌كنندگان ارسال مي‌كند و آنها مي‌توانند رویداد را به طور موازی و با اهداف خاص که برای آنها تعریف شده است ،پردازش کنند [2] .

**3-‌ نظارت بر منابع**

برای برسی منابع ، می‌شود از معماری رویداد محور برای نظارت و دریافت هشدار در مورد هرگونه ناهنجاری ،

تغییر و به‌روزرسانی استفاده کرد بدون اینکه نیازی باشد از بررسی مداوم استفاده کرد . این منابع می‌تواند شامل سطل‌های ذخیره‌سازی ، جداول پایگاه داده، توابع ‌بدون سرور،گره‌های محاسباتی و غیره باشد [2].

## سبك‌هاي مختلف الگوي معماري رويداد محور

سبك هاي مختلف الگوي معماري رويداد محور براساس نحوه عملكرد خود بر دو مدل مختلف تقسيم مي‌شوند كه در ادامه به آنها پرداخته خواهد شد[1] :

1. **مدل انتشار اشتراك**

اين يك زير ساخت پيام‌رساني مبتني بر اشتراك در جريان ‌رويداد است [3]. در این مدل هنگامی که یک رویداد منتشر می شود، مسیریاب رویداد را برای هر مشترک(مصرف‌کننده) ارسال می کند. پس از دریافت یک رویداد، نمی توان آن را دوباره پخش کرد و مشترکین جدید رویداد را نمی بینند[1].

1. **مدل جريان رويداد**

رویدادها در یک گزارش نوشته می شوند. رویدادها کاملاً سفارشی (در یک پارتیشن) و بادوام هستند. مصرف‌کنندگان در جریان مشترک نمی شوند، در عوض مصرف‌کننده می تواند از هر بخشی از جریان بخواند. مصرف‌کننده مسئول پیشبرد موقعیت خود در جریان است. این بدان معناست که یک مصرف‌کننده می تواند در هر زمان بپیوندد و می تواند رویدادها را دوباره پخش کند[1] .

مدل جريان رويداد نيز خود بر حسب نوع جريان رويداد به سه گونه ديگر تقسيم مي‌گردد كه عبارتند از:

**1-‌ پردازش رويداد گسترده**

پردازش یک جریان نامحدود از رویدادهای مرتبط، که در آن سوابق رویداد به ترتیبی ظاهر می‌شوند و با آگاهی از رویدادهای گذشته پردازش می‌شوند [4] . پردازش رویداد جریان معمولاً برای راه اندازی جریان بلادرنگ اطلاعات در داخل و اطراف سازمان استفاده می‌شود که تصمیم‌گیری به‌موقع را ممکن می‌سازد[11] .

**2-‌ پردازش رويداد ساده**

پردازش رویداد ساده زمانی است که یک رویداد بلافاصله اقدامی را در مصرف کننده رویداد ایجاد می کند [3].

پردازش رویداد ساده معمولا برای هدایت جریان بلادرنگی از کارها استفاده می‌شود که باعث کاهش زمان تاخیر و

هزینه تمام شده کسب‌وکار می‌شود [11] .

**3-‌ پردازش رويداد پيچيده**

در پردازش رویداد پیچیده یک مصرف کننده با استفاده از فناوری هایی مانند آپاچی استورم، مجموعه ای از رویدادها را پردازش می کند و به دنبال الگوهایی در داده های رویداد می گردد[1]. پردازش رویداد پیچده معمولاً برای شناسایی و پاسخ به ناهنجاری‌ها ، تهدیدها و فرصت‌های تجاری استفاده می‌شود [11].

## مزایای معماری رویداد محور

1. قابلیت تحمل خطا و بافرکردن

رویدادها ناهمزمان است، که باعث می‌شود رویدادها به محض اینکه اتفاق می‌افتند توسط تولییدکننده رویداد ارسال می‌گردند. مؤلفه مصرف کننده رویداد در این رویدادها مشترک می‌شوند و یا از جریان رویداد می‌خوانند . بنابراین اگر مصرف کننده ای از کار بیفتد، برنامه در غیاب آن به کار خود ادامه می دهد. رویدادها در مسیریاب رویداد ذخیره می‌شوند تا زمانی که مصرف کننده از موقعیت شکست بیرون بیاید و بتواند رویدادهای معلق شده را دریافت کند [5].

1. **اتصال سست یا جداسازی تولید‌کنندگان و مصرف کنندگان**

سیستم‌هایی که از معماری رویداد محوراستفاده می‌کنند دارای اجزایی هستند که به‌طور سست‌ متصل شده‌اند. این جداسازی به تفکیک منطقی تولید و مصرف رویدادها کمک می کند،بطوریکه نه تولیدکنندگان نگران نحوه مصرف رویدادها هستند، نه مصرف‌کنندگان نگران تولییدکنندگان رویدادها . به دلیل این اتصال سست، اجزا یا میکروسرویس های مختلف را می توان به زبان های دیگر توسعه داد یا از فناوری های مختلف استفاده کرد. همچنین به مقیاس بندی برنامه نیز کمک می شود. تولیدکنندگان یا مصرف‌کنندگان را می توان بر اساس الزامات بدون تغییر منطقی در سایر اجزاء اضافه یا حذف کرد [5] .

1. **مقرون به صرفه**

معماری‌های رویداد محور مبتنی بر عرضه هستند، بنابراین همه چیز بر حسب تقاضا اتفاق می‌افتد، زیرا رویداد خود را در روتر نشان می‌دهد. به این ترتیب، برای نظرسنجی مداوم برای بررسی یک رویداد هزینه ای پرداخت نمی کنید. این به معنای مصرف کمتر پهنای باند شبکه، استفاده کمتر از پردازنده است [2].

1. **قابلیت مقیاس‌پذیری زیاد**

اغلب می توان جریان‌های رویداد را به زیرجریان‌های نامرتبط تقسیم کرد و آنها را به صورت موازی پردازش کرد. همچنین این قابلیت وجود دارد در صورتیکه حجم رویدادهای پردازش نشده افزایش یافته باشد،میتوان تعداد مصرف‌کنندگان را برای پاسخگویی به این افزایش حجم مقیاس کرد. پلتفرم‌هایی مانند کافکا پردازش رویدادها را به ترتیب دقیق امکان‌پذیر می‌سازند و در عین حال امکان موازی‌سازی عظیم در سراسر جریان را فراهم می‌کنند‌ ‌[4].

**5- تجربیات کاربری بهتر**

رابطهای برنامه نویسی کاربردی رویداد محور با حذف بسیاری از مسئولیت‌هایی که قبلاً به کاربران محول شده بود، منجر به تجربه تعاملی بهتری برای کاربر نهایی با نیازهای تعاملی مدرن می‌شوند. البته، این ممکن است پیچیدگی را در تولید کننده افزایش دهد، اما خروجی کار ارزش این پیچیدگی را دارد [5].

## معایب معماری رویداد محور

الگوی معماری رویداد محور نیز مانند هر الگوی معماری دیگری علاوه بر مزایایی که دارند ، دارای معایبی نیز هستند که در این بخش به آنها پرداخته شده است .

**1-‌ محدود به پردازش ناهمزمان**

اگرچه معماری رویداد محور یک الگوی قوی برای جداسازی سیستم‌هاست ، اما کاربرد آن به پردازش‌ رویداد‌های

ناهمزمان محدود می‌شود. معماری رویداد محور در جایی که آغاز کننده باید قبل از ادامه به کار باید منتظر پاسخ باشد، نمی‌تواند به خوبی به عنوان جایگزینی برای تعاملات درخواست-پاسخ کار کند [4] .

**2-‌ ایجاد پیچیدگی اضافی**

درجایی که روش های قدیمی مشتری-سرور و درخواست-پاسخ فقط جزء اصلی را شامل می‌شود ، پذیرش معماری رویداد محور به یک کارگزار برای میانجیگری تعاملات بین تولییدکنندگان و مصرف‌کنندگان رویداد نیاز دارد [4].

**3-‌ توسعه و خطایابی پیچیده**

توسعه به دلیل ماهیت ناهمزمان الگو و همچنین ایجاد قرارداد ونیاز به شرایط مدیریت خطای پیشرفته‌‌‌‌‌‌‌‌‌‌‌‌‌تر در کد برای پردازشگرهای رویدادی که پاسخگو نیستند تا حدودی پیچیده می باشد[6] .

**4-‌ آزمون پذیری پایین**

گرچه تست واحد به طور کلی چندان سخت نیست . اما به یک سری ابزارهای تست خاص برای ایجاد رویدادها نیاز است . به علاوه بابت ماهیت آسنکرون این الگو ، آزمون‌پذیری پیچیده‌تر می‌شود [6] .

# نتیجه‌گیری

آنچه به عنوان نتیجه‌گیری از این تحقیق می‌توان بیان کرد این است که الگوی معماری رویداد محور باعث کاهش جفت‌شدگی در بین اجزای مختلف سیستمی است که تحت این الگوی معماری طراحی می‌شوند . این الگو یکی از بهترین انتخاب‌ها برای طراحی های سیستم‌هایی است که به طور ناهمزان کار می‌کنند . همچنین خاصیت مهمی دیگر باید در نظر گرفت این مسئله است که اجزای مختلف سیستمی که با این الگو طراحی شده اند قابلیت این را دارند که به طور مستقل از هم مقیاس گردند . ذکر این نکته نیز حائز اهمیت است که در بکار‌گیری هر الگوی طراحی ایجاد هزینه امری طبیعی است ،ولی در زمینه هایی که این الگوی طراحی به کار می‌اید مزایایی که به وجود می‌آورد از هزینه‌های ناشی از بکار گیری آن بیشتر است .

# پیوست

آپاچی کافکا

آپاچی کافکا (Apache Kafka )یک پلتفرم توزیع‌شده برای پردازش داده های جریانی بوده و قادر به رسیدگی و پردازش چندین تریلیون رویداد است به صورت همزمان است. کافکا در ابتدا به عنوان یک ابزار برای ارسال پیامهایی با تعداد بالا بکار برده می شد. کافکا از زمان ایجاد و منبع باز شدن توسط لینکدین در سال ۲۰۱۱ ، به سرعت از ارسال پیام به یک پلتفرم پردازش توزیعی کامل تبدیل شده است.

آپاچی کافکا، به عنوان یک پلتفرم  داده های در جریان، دارای قابلیتهایی ازقبیل زمان پاسخ کوتاه، کارایی بالاو تحمل خطا بوده و قادر به پردازش جریانهایی سریع از رویدادها می باشد. کافکا، برای پشتیبانی از برنامه های کاربردی مشتری و اتصال سیستم های پایین دست با داده های زمان واقعی پاسخ های در حد میلی ثانیه فراهم می کند.

آپاچی استورم

آپاچی استورم (Apache Storm ) یک چارچوب پردازشی توزیعی برای پردازش داده های در جریان  می باشد که به زبان برنامه نویسی clojure نوشته شده است. این برنامه در ابتدا توسط ناتان مارتس و همکارانش در موسسه بکتایپ ایجاد شده و بعد از تصاحب آن توسط تویتر به صورت نرم افزاری منبع باز درآمد.

استورم یک سیستم محاسبات زمان واقعی توزیع شده برای پردازش حجم زیاد از داده ها با نرخ تولید بالا است. استورم  بسیار سریع بوده و قادر است بیش از یک میلیون رکورد در ثانیه را برای هر گره در یک خوشه با اندازه متوسط پردازش کند. شرکت ها این سرعت را کنترل کرده و آن را با دیگر برنامه های دسترسی داده در هدوپ ترکیب می کنند تا از وقایع نامطلوب جلوگیری کرده یا نتایج مثبت را بهینه سازند.

برخی از فرصت های تجاری جدید و ویژه در استفاده از استورم عبارتند از: مدیریت خدمات مشتری در زمان واقعی، کسب درآمد از داده ها، داشبورد های عملیاتی، تجزیه و تحلیل امنیت سایبری و تشخیص تهدید.

استورم ساده بوده و توسعه دهندگان می توانند توپولوژیهای استورم را با استفاده از هر زبان برنامه نویسی ایجاد کنند.

# مراجع:

ماکروسافت 1 https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/event-driven

آمازون 2 https://aws.amazon.com/event-driven-architecture/

رد هت 3

https://www.redhat.com/en/topics/integration/what-is-event-driven-architecture

مدیوم 4

https://medium.com/microservicegeeks/introduction-to-event-driven-architecture-e94ef442d824

نیلل 5

https://www.neebal.com/blog/advantages-and-disadvantages-of-event-driven-architecture

مرجع 6 اوریلی

https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch02.html

مرجع 7 کتاب **Mastering JBoss Drools 6**

**https://hazelcast.com/glossary/event-driven-architecture/ مرجع 8**

مرجع 9

https://www.koyeb.com/blog/understanding-event-driven-architecture-and-serverless-opportunities

مرجع شماره 10

https://pradeeploganathan.com/architecture/event-driven-architecture/

مرجع شماره 11

EventDrivenArchitectureOverview\_ElementalLinks\_Feb2011\_2.pdf

1. Event Driven Architecture Pattern(EDAP) [↑](#footnote-ref-1)
2. coupling [↑](#footnote-ref-2)