



ارائه‌ی ابزارهای به‌منظور معیار سنجی سامانه‌های پردازش رویداد پیچیده

پایان نامه کارشناسی
مهندسی کامپیوتر (گرایش نرم‌افزار)

ارائه دهنده: محمدرضا برازش
استاد راهنما: دکتر محسن شریفی

شهریور ۱۳۹۵

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

گرایش روز افزون به معماری‌های مبتنی بر رویداد از یک سو و فراگیر شدن سامانه‌های محاسباتی از سوی دیگر، منجر به رشد فزاینده‌ی منابع تولید کننده‌ی رویداد و به تبع آن افزایش نرخ تولید رویداد شده است. در بسیاری از حوزه‌های کاربرد، لازم است سامانه‌های محاسباتی، با پردازش رویدادها و شناسایی الگوهای از پیش تعریف شده و انتزاع آنها به سطوح بالاتر لایه‌های کاربرد، ادراک لازم را نسبت به محیط خود کسب کرده و پس از اتخاذ تصمیم مناسب، واکنش تعریف شده‌ای را داشته باشد. به این نوع پردازش، پردازش رویدادهای پیچیده می‌گویند. افزایش و گسترده‌ی کاربردهای سامانه‌های پردازش رویداد پیچیده، پدید آورنده‌ی نیاز به ابزاری به منظور معیار سنجی کارایی این سامانه‌ها شده است. با این حال، عدم وجود یک استاندارد مشخص، و همینطور چالش‌های بسیاری که در زمینه‌ی پردازش جریان‌های داده وجود دارد، این امر را بسیار دشوار نموده است. در این پژوهش سعی شده تا با ارائه‌ی مجموعه‌ای از نرم افزارها، ابزاری به منظور معیار سنجی کارایی سامانه‌های پردازش رویداد پیچیده ارائه نماییم که بتواند بر این چالش‌ها فائق برآمده، و در عین استقلال از سامانه‌ی پردازش رویداد، به بررسی عملکرد و کارایی آن بپردازد. این ابزار، با مطرح کردن معیارهایی مانند نرخ ارسال و دریافت، و رویدادهای از دست رفته در حین انتقال، معیار مناسبی برای مقایسه و بررسی کارایی سامانه‌های پردازش رویداد پیچیده ارائه داده و در عین حال بر چالش‌های موجود در پردازش جریان‌های رویدادها غلبه می‌نماید.

واژه‌های کلیدی: رویداد پیچیده، پردازش رویدادهای پیچیده، معیار سنجی سامانه‌های پردازش رویداد پیچیده، معیار سنجی جریان‌های رویداد

فهرست مطالب

عنوان	شماره‌ی صفحه
فصل اول: مقدمه	۱
۱-۱- مقدمه	۲
۱-۲- انگیزه پژوهش	۳
۱-۳- چالش‌ها	۴
۱-۳-۱- چالش استقلال از سامانه‌ی پردازش رویداد پیچیده	۴
۱-۳-۲- چالش استقلال از معماری شبکه و قابلیت گسترش پذیری	۴
۱-۳-۳- چالش استقلال از سیستم عامل	۴
۱-۳-۴- چالش قابلیت تکرار آزمایش	۵
۱-۳-۵- چالش معیارهای اندازه‌گیری بازدهی	۵
۱-۴- اهداف پژوهش	۵
۱-۵- ساختار پایان نامه	۶
فصل دوم: مفاهیم بنیادی	۷
۲-۱- مقدمه	۸
۲-۲- پردازش جریان اطلاعات	۸
۲-۳- پایگاه داده‌های فعال	۹
۲-۴- پردازش جریان داده‌ها	۹
۲-۵- پردازش رویدادهای پیچیده	۱۰
۲-۶- رویداد	۱۱
۲-۶-۱- رویدادهای ساده	۱۲
۲-۶-۲- رویدادهای پیچیده	۱۲
۲-۷- الگو و قاعده	۱۳
۲-۸- عملگر	۱۴
۲-۹- مدل زمانی	۱۵
۲-۱۰- سامانه‌های پردازش رویدادهای پیچیده و موجودیت‌های آن	۱۶
۲-۱۱- خلاصه	۱۷

عنوان	شماره‌ی صفحه
فصل سوم: شرح سامانه‌ی پیشنهادی	۱۸
۳-۱- مقدمه	۲۰
۳-۲- تولید کننده‌ی فایل‌های رویداد	۲۱
۳-۲-۱- ورودی‌ها	۲۲
۳-۲-۱-۱- نوع رویداد	۲۲
۳-۲-۱-۲- تنظیمات کلی	۲۴
۳-۲-۲- فایل‌های خروجی	۲۴
۳-۲-۳- بررسی فرآیند‌ها	۲۷
۳-۳- ارسال کننده‌ی رویداد‌ها	۳۳
۳-۳-۱- الگوریتم سطل نشت دار	۳۵
۳-۳-۲- فرآیند ارسال رویداد‌ها	۳۶
۳-۴- دریافت کننده رویدادها	۳۸
۳-۴-۱- فرآیند دریافت رویداد‌ها	۳۸
۳-۵- مقایسه کننده‌ی فایل‌های رویداد	۴۰
۳-۵-۱- فرآیند مقایسه‌ی فایل‌ها	۴۰
۳-۶- خلاصه	۴۱
فصل چهارم: آزمایش‌ها و مشاهدات	۴۲
۴-۱- آزمایش‌ها	۴۳
۴-۱-۱- مقدمه	۴۳
۴-۱-۲- محیط آزمایش	۴۳
۴-۱-۳- شرح آزمایش	۴۳
۴-۱-۴- مشاهدات و نتایج	۴۴
فصل پنجم: نتیجه‌گیری و کارهای آینده	۴۶
مراجعه	۴۹
پیوست‌ها	۵۱
پیوست الف: واژه‌نامه فارسی به انگلیسی	۵۲

شماره‌ی صفحه

عنوان

پیوست ب: واژه‌نامه انگلیسی به فارسی ۵۵

فهرست شکل‌ها

عنوان	شماره‌ی صفحه
شکل ۱-۱ نحوه‌ی کار یک سامانه پردازش رویداد ساده	۳
شکل ۲-۱ ساختار قواعد در پایگاه داده‌های فعال	۹
شکل ۲-۲ ساختار پرس و جوها در سامانه‌های پردازش جریان داده	۱۰
شکل ۲-۳ رویدادی پیچیده که حاصل ترکیب دو رویداد ساده است	۱۳
شکل ۲-۴ الگوی بررسی موفقیت آمیز شکایت مشتریان در یک فروشگاه	۱۴
شکل ۲-۵ ساختار عمومی قواعد در سامانه‌های پردازش رویدادهای پیچیده	۱۴
شکل ۲-۶ مفهوم پنجره‌ی زمانی	۱۶
شکل ۲-۷ سامانه‌ی پردازش رویدادهای پیچیده [۸]	۱۷
شکل ۳-۱ نمایی از صفحه‌ی اصلی تولید کننده‌ی فایل رویداد	۲۱
شکل ۳-۲ نمایی از پنجره‌ی تعریف یک نوع رویداد	۲۲
شکل ۳-۳ نمایی از پنجره‌ی افزودن ماشین مقصد	۲۴
شکل ۳-۴ فایل xml شامل اطلاعات انواع رویدادها	۲۵
شکل ۳-۵ فایل xml شامل اطلاعات ماشین‌های مقصد	۲۵
شکل ۳-۶ نمایی از یک فایل رویداد	۲۶
شکل ۳-۷ جدول عناصر مختلف موجود در سرنوشته‌ی فایل	۲۶
شکل ۳-۸ کلاس‌های تولید کننده‌ی فایل رویداد	۲۷
شکل ۳-۹ نمایی از کلاس ConfigManager	۲۸
شکل ۳-۱۰ نمودار مراحل تولید یک رویداد	۲۹
شکل ۳-۱۱ تعریف کلاس JEvent	۲۹
شکل ۳-۱۲ کد مربوط به حالت اختصاص گردشی	۳۰
شکل ۳-۱۳ کلاس Dispatcher	۳۰
شکل ۳-۱۴ رابطه‌ی کلاس‌های دخیل در تولید فایل رویداد	۳۱
شکل ۳-۱۵ مراحل تولید فایل رویداد	۳۲
شکل ۳-۱۶ کد پروتوباف مربوط به رویدادها	۳۳
شکل ۳-۱۷ کلاس‌های کامپایل شده توسط پروتوباف	۳۳
شکل ۳-۱۸ نمایی از ارسال کننده رویدادها	۳۴
شکل ۳-۱۹ نمایی از عملکرد الگوریتم سطل نشت دار	۳۵
شکل ۳-۲۰ کلاس Machine	۳۶

عنوان	شماره‌ی صفحه
شکل ۳-۲۱ کلاس‌های ارسال‌کننده‌ی رویداد	۳۷
شکل ۳-۲۲ نمایی از دریافت‌کننده‌ی رویدادها	۳۸
شکل ۳-۲۳ کلاس‌های مسئول دریافت رویدادها	۳۹
شکل ۳-۲۴ نمایی از مقایسه‌کننده‌ی فایل‌های رویداد	۴۰

فصل اول:

مقدمه

۱-۱ مقدمه

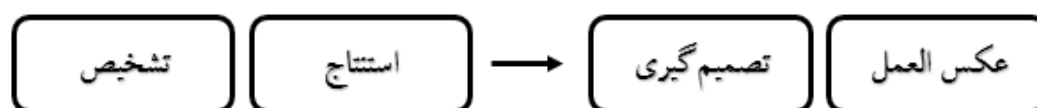
به سبب پویایی دنیای پیرامون ما، تغییرات متعدد در محیط اطرافمان با سرعت زیاد در حال وقوع است. از نوسانات ارزی گرفته تا تغییرات آب و هوایی و هزاران مثال دیگر. این تغییرات در غالب رویدادها برای ما قابل درک است. در حقیقت ما با زیر نظر گرفتن دنباله‌ای از رویدادها، آنها را درک می‌کنیم و پس از تحلیل آنها، نسبت به این تغییرات واکنش نشان می‌دهیم. به عنوان مثال ما با شنیدن صدای بلند و مشاهده‌ی افرادی که از یک نقطه‌ی خاص دور می‌شوند احساس خطر می‌کنیم و ناخودآگاه سعی می‌کنیم فاصله‌ی خود را از آن نقطه افزایش دهیم. با این توصیف، ادارک ما از محیط پیرامون مان، از طریق دنباله‌ای از رویدادهاست. اما در هر لحظه تعداد زیادی از رویداد در اطراف ما رخ می‌دهد که تنها بخشی از آنها برای ما حایز اهمیت است و واکنش ما را بر می‌انگیزد.

رشد سریع در فناوری اطلاعات، باعث عمومی شدن سامانه‌های کامپیوتری شده و امروزه در تمام جنبه‌های زندگی بشر نفوذ کرده است. مثلاً ظهور و رشد اینترنت، باعث تغییر در نحوه‌ی ارتباط ما با یکدیگر و همچنین تغییر روش تولید، توزیع و استفاده از اطلاعات شده است. امروزه با استفاده از موتورهای جستجوی قدرتمندی مانند گوگل، این امکان برای ما فراهم آمده تا از میان حجم بالایی از داده‌های موجود در اینترنت، اطلاعات مورد نظر خود را بیابیم. امروز خبرگزاری‌ها، شبکه‌های اجتماعی، متن‌های وبلاگ‌ها در کنار حسگرهای متعدد تلفن‌های هوشمند، خودروهای مجهز به کامپیوتر و بسیاری موارد دیگر، منابعی برای تولید اطلاعات جدید محسوب می‌شوند. اگر همان رویه‌ای که در بالا در مورد نحوه‌ی تعامل بشر با محیط پیرامونش گفتیم، در دنیای فناوری نیز متصور باشیم، آنگاه می‌توانیم این اطلاعات را در غالب رویدادهایی ببینیم که کاربران معمولاً علاقه دارند از وقوع برخی از آنها آگاه شوند [1]. همچنین می‌توانیم هر تعامل سامانه‌ی محاسباتی با محیط بیرونی خود را، شامل تبادل تعدادی رویداد بدانیم که سامانه لازم باشد در قبال وقوع برخی از این رویدادهایی واکنش نشان دهد.

امروزه با رشد روز افزون این سامانه‌ها، نیاز به بررسی کارایی آنها بیش از پیش محسوس است. در ادامه‌ی این فصل، ابتدا به انگیزه‌های این پژوهش می‌پردازیم. پس از آن، با معرفی و توضیح مختصر چالش‌های مرتبط با آن، خلاصه‌ای از اهدافمان را بازگو می‌کنیم. در نهایت نیز این فصل را با معرفی ساختار پایانامه، به اتمام می‌رسانیم.

۱-۲ انگیزه پژوهش

امروزه با افزایش روز افزون منابع تولید کننده‌ی رویداد و به تبع آن افزایش تعداد و گوناگونی رویدادها در حوزه‌های مختلف نرم‌افزاری مواجه هستیم. از سوی دیگر با معرفی فناوری‌های جدیدی مانند اینترنت اشیاء^۱ سامانه‌های سایبری-فیزیکی^۲ و ...، گرایش به معماری‌های مبتنی بر رویداد^۳ به دلیل دارا بودن خواصی مانند جدایی کامل^۴ و ارتباط ناهمگام^۵ افزایش چشم‌گیری داشته است..



شکل ۱-۱ نحوه ی کار یک سامانه پردازش رویداد ساده

در سال‌های اخیر، با توجه به آنچه در بالا گفتیم، پردازش رویدادها، اهمیت به سزایی پیدا کرده است و پژوهش‌های متعددی بر روی آن انجام شده است و محصولات زیادی نیز در ارتباط با آن تولید شده است. می‌تواند روند تکاملی در سامانه‌های پردازش رویداد را به سه نسل تقسیم کرد. در نسل اول، پژوهش‌ها تنها محدود به افزودن امکاناتی به موتورهای پایگاه داده^۶ بود. در نسل دوم ویژگی‌های پیشرفته‌تری از قبیل تحمل اشکال^۷ و بهبود زبان توصیف پرس و جوها^۸ به این سامانه‌ها افزوده گردید. اما در نسل سوم، تمرکز اصلی پژوهش‌ها، بهبود عملکرد این سامانه‌ها در مقیاس‌های بالا است. پیشرفت و فراوانی سامانه‌های پردازش رویداد، پدید آورنده ی نیاز به معیار سنجی و مقایسه ی آن‌ها از لحاظ بازدهی شده است. با این وجود، فقدان یک ابزار استاندارد برای این منظور حس میشود. در این پژوهش قصد داریم با معرفی یک ابزار مناسب، پاسخگوی این نیاز باشیم.

- ^۱ Internet of Things
- ^۲ Cyber-Physical System
- ^۳ Event Driven
- ^۴ Decoupling
- ^۵ Asynchronous
- ^۶ Data base Engine
- ^۷ Fault Tolerance
- ^۸ Query

۱-۲ چالش‌ها

مهمترین چالش برای معیار سنجی سامانه های پردازش رویداد پیچیده ، عدم وجود یک استاندارد مورد توافق برای بررسی این سامانه ها است. برای حصول چنین استانداردی، باید ویژگی های استقلال از سامانه ی پردازش رویداد پیچیده، استقلال از معماری شبکه و قابلیت گسترش پذیری، استقلال از سیستم عامل و قابلیت تکرار آزمایش نیز حاصل گردند.

همچنین در این راه، چالش های موجود در پردازش جریان های داده با نرخ بالا نیز، مشکل افکن اند. پردازش جریان های داده ای ، به دلیل حساسیت زمانی بالا، نیازمند ابزاری است که از پردازنده و حافظه با حداکثر بازدهی ممکن استفاده نماید.

۱-۲-۱ چالش استقلال از سامانه ی پردازش رویداد پیچیده

فراگیر شدن و افزایش کاربرد سامانه های مبتنی بر رویداد، همچنین افزایش منابع تولید کننده ی جمع ترتیب رویدادها باعث فراوانی در سامانه های پردازش رویداد پیچیده گشته است. برای معیار سنجی منصفانه و کم خطای این سامانه ها نیازمند ابزاری هستیم که بدون هیچ وابستگی به سامانه پردازش رویداد، و در لایه ای مجزا از آن عمل کند .

۱-۲-۲ چالش استقلال از معماری شبکه و قابلیت گسترش پذیری

رشد روز افزون سامانه های پردازش رویداد باعث کاربرد آنها در حیطه ی بسیار وسیعی از شبکه ها و سیستم ها شده است. این سامانه ها در شبکه هایی با وسعت جهانی مانند بازار های بورس و سازمان های تجاری بین المللی و همچنین شبکه های کوچک تر مانند ادارات خصوصی و حتی رایانه های شخصی به کار گرفته میشوند. به همین دلیل باید ابزار مورد نظر قابلیت کار در این محیط ها و تطبیق با مقیاس شبکه را داشته باشد.

۱-۲-۳ چالش استقلال از سیستم عامل

همانطور که ذکر شد، سامانه های پردازش رویداد ممکن است در طیف عظیمی از سیستم های مرتبط

مشغول به کار باشد. از آنجا که هر کدام از این سیستم ها ممکن است بر پایه ی سیستم عامل متفاوتی کار کنند، لازم است ابزار مورد نظر بر روی سیستم عامل های متفاوت قابل اجرا باشد.

۴-۲-۱ چالش قابلیت تکرار آزمایش

از مهم ترین ویژگی های هر سامانه ی معیار سنجی، قابلیت تکرار آزمایش می باشد. به این معنا که باید هر آزمایش پس از انجام ، مجددا و با حداقل تغییرات قابل اجرا در همان محیط یا محیط های دیگر باشد. این ویژگی علاوه بر بالا بردن دقت آزمایش ها توسط امکان آزمایش های متوالی، باعث میشود تا با تکرار یک آزمایش در یک شبکه یا سیستم متفاوت، به مقایسه ی کارایی آن ها بپردازیم.

۵-۲-۱ چالش معیار های اندازه گیری بازدهی

با توجه به کمیت های بسیار زیاد دخیل در یک سامانه ی پردازش رویداد ، مانند بازدهی سخت افزاری، نرم افزاری و شبکه ای ، یکی از دلایل مهم فقدان یک ابزار استاندارد برای معیار سنجی سامانه های پردازش رویداد، عدم توافق بر معیار های این اندازه گیری می باشد.

۳-۱ اهداف پژوهش

هدف از این پژوهش، ارائه ی سامانه ای است که با رفع چالش های مطرح شده، بتواند به عنوان ابزار استاندارد به منظور معیار سنجی سامانه های مختلف پردازش رویداد پیچیده به کار رود. از این ابزار باید بتوان مستقل از متغیر هایی نظیر ساختار شبکه، نوع سامانه پردازش رویداد، و سیستم عامل به منظور معیار سنجی و بررسی بازدهی سامانه های پردازش رویداد مختلف استفاده نمود. این سامانه کمیت های مختلفی از جمله نرخ ارسال و دریافت رویدادها، بررسی صحت رویداد ها و تعداد رویداد های از دست رفته در شبکه ، و مقدار کاربری پردازشگر و حافظه به هنگام ارسال و دریافت رویداد ها را به عنوان معیارهایی موثر برای اندازه گیری کیفی سامانه های پردازش رویداد ارائه میدهد.

۴-۱ ساختار پایان نامه

ارائه‌ی تاریخچه‌ی مختصری از سامانه‌های پردازش رویداد و تعریف مفاهیم بنیادی پردازش رویدادهای پیچیده، موضوع فصل دوم این گزارش است. در فصل سوم، سازوکار پیشنهادی را به طور کامل معرفی و بررسی می‌کنیم. در فصل چهارم، با ارائه‌ی نتایج آزمایش‌ها و تحلیل آنها، اقدام به ارزیابی سازوکار پیشنهادی می‌کنیم. در نهایت نیز، فصل پنجم را به نتیجه‌گیری و بررسی کارهای آتی این حوزه اختصاص داده‌ایم.

فصل دوم:

مفاهیم بنیادی

۲-۱ مقدمه

رویدادها در رده‌ی مختلفی از سامانه‌ها با رویکردهای متعدد و سازوکارهای تشخیص گوناگونی پردازش می‌شوند. در این فصل قصد داریم پس از تشریح مختصری از تاریخچه‌ی پردازش جریان اطلاعات، این سامانه‌ها را معرفی و از جنبه‌های مختلف با یکدیگر بررسی کنیم و در خلال آن با مسیری که منتهی به معرفی سامانه‌های پردازش رویدادهای پیچیده شده آشنا شویم. در این فصل همچنین قصد داریم پیش از ارائه‌ی طرح پیشنهادی، مفاهیم اولیه‌ای که در این پژوهش استفاده شده‌اند را تعریف کنیم.

۲-۲ پردازش جریان اطلاعات

افزایش همه روزه‌ی نرم‌افزارهای توزیعی که با ظهور فناوری اینترنت امکان نقش آفرینی گسترده‌تری را در زندگی ما فراهم کرده‌اند، بستری را برای تولید حجم بالایی از اطلاعات ایجاد کرده است. وجود این حجم از اطلاعات، اشتیاق زیاد کاربران را برای کشف الگوهای سودمند از میان آنها ایجاد می‌کند. لازمه کشف این الگوها، پردازش دائمی جریان اطلاعاتی است که با نرخ غیر قابل پیش‌بینی و از منابع متعدد وارد سامانه پردازشی می‌شوند. به عنوان مثال فرض کنید یک کاربر علاقه دارد اگر خبری در سه خبرگزاری مختلف در فاصله زمانی سی دقیقه منتشر شد، از آن به عنوان یک خبر مهم آگاه شود.

به صورت سنتی، اطلاعات سامانه‌های محاسباتی در پایگاه‌های داده ذخیره می‌شوند که دارای دو خصیصه‌ی عمده هستند. نخست اینکه داده‌ها می‌بایست پیش از آنکه مورد پردازش قرار بگیرند ذخیره شوند و دوم اینکه، داده‌ها به صورت ناهمگام و در زمانی مورد پردازش قرار می‌گیرند که به شکل صریح از طرف کاربر درخواست آن داده شود. هرچند استفاده از پایگاه داده‌های سنتی شیوه‌ای کارآمد برای اجرای پرس و جوها روی حجم محدودی از اطلاعات ذخیره شده محسوب می‌شود، اما به دلیل این دو خاصیت عمده شان، برای کاربردهایی که در آنها نیاز است پرس و جوها و قواعد به شکل ادامه‌داری روی جریان بی‌کرانی از اطلاعات وارد شده به سامانه اجرا شوند روش مناسبی محسوب نمی‌شوند.

این نیازمندی‌ها، سبب شده رده‌ی مختلفی از سامانه‌ها که هدف از آنها به طور خاص پردازش جریانی از

اطلاعات است توسعه یابند. هرچند هدف کلی این سامانه‌ها مشترک است، اما در جنبه‌های مختلفی از جمله معماری، مدل داده‌ای و روش‌های پردازش با یکدیگر تفاوت‌های عمده دارند. در ادامه به بررسی مختصر پایگاه داده‌های فعال^۱ و سامانه‌های پردازش جریان داده^۲ که هر کدام تلاشی برای پردازش جریان اطلاعات هست می‌پردازیم و در خلال آن، روندی که منجر به معرفی سامانه‌های پردازش رویدادهای پیچیده شده است را شرح می‌دهیم.

۲-۳ پایگاه داده‌های فعال

پایگاه داده‌های سنتی رفتاری منفعل^۳ دارند و تنها در پاسخ به درخواست‌های صریح ما اقدام به بازیابی^۴ داده‌ها می‌کنند. از همین رو، این امکان را برای ما فراهم نمی‌آورند تا در مواقعی که شرایط از پیش تعریف شده‌ای رخ داد، اعلان^۵ داشته باشند. پایگاه داده‌های فعال برای غلبه بر این محدودیت توسعه یافتند. در این پایگاه‌های داده، هنگامی که شرایط خاصی برقرار می‌شود، اقدام به اجرای خودکار یک رفتار می‌کنند [47]. در حقیقت آنها یک تعمیم روی پایگاه داده‌ی سنتی هستند که رفتار واکنشی^۶، از لایه‌ی کاربرد به لایه‌ی داده منتقل شده است [11]. ساختار قواعد در پایگاه داده‌های فعال مطابق شکل (۲-۱) می‌باشند [12].

WHEN <Event Occurs>
IF <Conditions Satisfy>
DO <Actions>

شکل ۲-۱ ساختار قواعد در پایگاه داده‌های فعال

۲-۴ پردازش جریان داده‌ها

در پایگاه داده‌های فعال، بررسی قواعد روی داده‌های ذخیره شده انجام می‌پذیرد. به همین دلیل، اگر تعداد قواعد بیش از یک آستانه‌ی مشخص باشد یا نرخ ورود رویدادها به پایگاه داده بالا باشد، کارآمدی

- ۱ Active Data Base
- ۲ Data Stream Management System
- ۳ Passive
- ۴ Retrieve
- ۵ Notification
- ۶ Reactive

آن آفت محسوسی خواهد داشت [11]. همچنین هدف اصلی در آنها، تشخیص رویدادهای ساده است. مثلاً قاعده‌ای را فرض کنید که به ازای اضافه شدن یک رکورد به جدول تراکنش، به شرطی که رقم آن بیش از یک میلیارد تومان باشد، به متصدی مربوط هشدار بدهد. در واقع پایگاه داده‌های فعال از تشخیص الگوهای پیچیده عاجز هستند [1].

برای غلبه بر این مشکل، رده‌ی جدیدی از سامانه‌ها، برای پردازش جریانی از اطلاعات در زمانی مناسب، توسعه داده شد که به آن سامانه‌های پردازش جریان داده می‌گوییم. بر خلاف رفتار عمومی در پایگاه‌های داده که پرس و جوها روی داده‌های ذخیره شده اجرا می‌شود، در پردازش جریان داده، هدف اجرای دائمی یک مجموعه از پرس و جوها و جوهایی مانا روی داده‌هایی است که وارد سامانه می‌شود [13]. ساختار پرس و جوها در سامانه‌های پردازش جریان داده، مطابق شکل (۲-۲) است [12].

```
SELECT *
FROM <Input Stream>
WHERE <Join and Other
Conditions>
WITHIN <Time Window>
```

شکل ۲-۲ ساختار پرس و جوها در سامانه‌های پردازش جریان داده

۲-۵ پردازش رویدادهای پیچیده

به هر پردازشی که واحد محاسباتی روی یک رویداد انجام می‌دهد، پردازش رویداد می‌گوییم [14]. این پردازش می‌تواند شامل اجرای اعمالی چون خواندن، ایجاد، تبدیل و پاک کردن رویداد باشد. به عنوان مثال، بیماری را فرض کنید که توسط حسگرهای اندازه‌گیری متعددی در حال مانیتور شدن است. مقادیری که توسط حسگرها به بخش مانیتورینگ ارسال می‌شوند حکم رویداد را دارند و تحلیل بر آمده از این رویدادها، ناشی از پردازش آنها است [15].

پردازش رویدادهای پیچیده، مجموعه‌ای برای شناسایی الگویی خاص در دنباله‌ای از رویدادها است [4]. برای شناخت پردازش رویدادهای پیچیده، نیاز به دانستن مفاهیمی چون رویداد هستیم که در ادامه به آنها می‌پردازیم.

۶-۲ رویداد

رویداد، وقوع یک پیش‌آمد در یک سامانه یا دامنه محسوب می‌شود [15]. یک رویداد در برگیرنده‌ی سه خصیصه‌ی عمده است. اول آنکه در دامنه‌ای که به وقوع پیوسته، معنی‌دار است. دوم آنکه پیش‌آمد آن آنی محسوب می‌شود؛ به این معنا که در یک زمان مشخص اتفاق می‌افتد. در نهایت سومین خصیصه‌ی یک رویداد، تجزیه‌ناپذیری آن است؛ به این معنا که یا به طور کامل اتفاق می‌افتد یا هرگز روی نمی‌دهد. در دنیای محاسبات، از رویداد به عنوان موجودیتی که نشان‌دهنده‌ی یک پیش‌آمد باشد یاد می‌شود. هر رویداد شامل اطلاعاتی هست که آن را توصیف می‌کند. یک نوع رویداد^۲، تعیین‌کننده‌ی شمایی^۴ از ویژگی‌های یک مجموعه از رویدادهای مشابه است. مثلاً در جدول (۱-۲)، نوع رویداد تراکنش را مشاهده می‌کنیم.

جدول ۱-۱ انواع رویداد تراکنش

Description	Type
Transaction ID	NUMBER
Amount	NUMBER
Card Data	TEXT
Terminal ID	TEXT
Merchant ID	NUMBER
Created Date	DATE
Switch Send Status	TEXT
Settlement Status	TEXT
Response Code	TEXT
RRN	TEXT

با توجه به تعریف بالا، ممکن است رویدادهای متعددی در یک دامنه‌ی مورد بررسی به وقوع بپیوندند، هرچند پردازش همه‌ی آنها اهمیتی نداشته باشد و منجر به فهم خاصی نشود. در [16] وقوع رویدادی که نیاز به واکنش داشته باشد را وضعیت تعریف کرده است. بالا رفتن دمای پردازنده‌ی یک کامپیوتر مثالی از وضعیت محسوب می‌شود.

۱ Instantaneous
۲ Atomic
۳ Event Type
۴ Schema
۵ Situation

۱-۶-۲ رویدادهای ساده

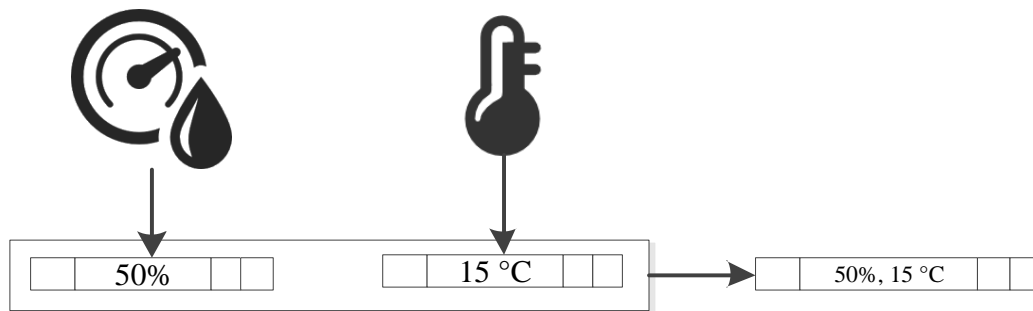
هر رویدادی که هیچ‌گونه عملیاتی از قبیل خلاصه‌سازی، نمایش، تفسیر^۱ یا ترکیب روی آن انجام نشده باشد را رویداد ساده^۲ می‌گوییم [14]. خواستگاه یک رویداد ساده که گاهی از آن با عنوان رویداد ابتدایی^۳ نیز یاد می‌شود، منبع تولید کننده‌ی رویداد است [1].

۲-۶-۲ رویدادهای پیچیده

همان‌طور که پیشتر نیز گفتیم، کشف یک الگو از رویدادها، حاوی اطلاعات با ارزش‌تری نسبت به یک رویداد ساده است [1]. اما پیش از آن لازم است بدانیم چه رویدادهایی را پیچیده خطاب می‌کنیم. یک رویداد پیچیده، رویدادی است که از یک مجموعه رویداد دیگر منتج شده باشد. حالت‌های مختلفی وجود دارد تا یک رویداد پیچیده ایجاد شود که در زیر به آنها پرداخته‌ایم [17].

رویداد پیچیده می‌تواند صرفاً حاصل الحاق^۴ چند رویداد باشد. مثلاً در یک اطلاق سرور، اگر رویدادهای دو حسگر^۵ دما و رطوبت نه به طور جدا، که در غالب یک رویداد به بخش مانیتورینگ^۶ ارسال شود، آنگاه این رویداد پیچیده محسوب می‌شود.

۱ Summarizing
۲ Representing
۳ Interpreting
۴ Simple Event
۵ Primitive Event
۶ Join



شکل ۲-۳: رویدادی پیچیده که حاصل ترکیب دو رویداد ساده است

همچنین یک رویداد پیچیده می‌تواند حاصل تجمیع^۱ یک رده از رویدادها باشد. مثلاً رویدادی را فرض کنید که میانگین معدل دانشجویان یک دانشکده در یک نیم سال تحصیلی باشد.

یک رویداد پیچیده حاصل می‌تواند تجرید^۲ مجموعه‌ای از رویدادها باشد. مثلاً سونامی سال ۲۰۰۴ در اندونزی، تجریدی از تعداد زیادی رویداد طبیعی بود [14].

در نهایت یک رویداد پیچیده می‌تواند نتیجه‌ی ترکیب تعدادی رویداد ساده یا پیچیده باشد. این ترکیب می‌تواند شامل ترکیب فصلی^۳، ترکیب عطفی^۴ یا دنباله‌ای منظم از رویدادها باشد که در غالب یک الگو یا قاعده بیان می‌شود [14]. مثلاً دنباله‌ای از سه شکایت یک مشتری در هفته‌ی جاری از فروشگاه، رویداد پیچیده‌ی مشتری عصبانی را تولید می‌کند.

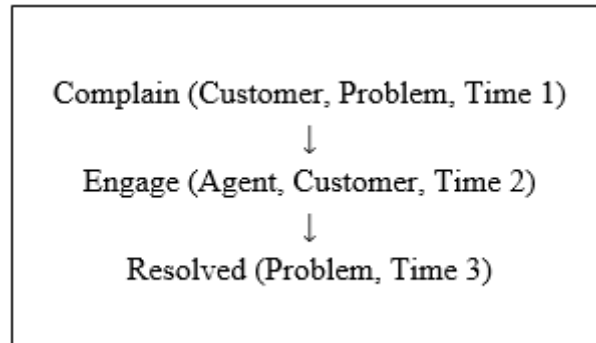
باید به این نکته توجه داشت که یک رویداد ممکن است در یک حوزه‌ی کاربرد پیچیده محسوب شود، در حالی که در حوزه‌ی کاربرد دیگری یک رویداد ساده محسوب شود.

۲-۷ الگو و قاعده

برای توصیف رویدادهای پیچیده، از الگوها^۱ استفاده می‌کنیم. یک الگوی رویداد، قالبی محسوب می‌شود که در آن نشان داده می‌شود چگونه یک مجموعه رویداد، رویدادی پیچیده را تشکیل می‌دهند [18]. به بیانی دیگر، یک الگو، ارتباط رویدادهای تشکیل دهنده‌ی یک رویداد پیچیده را مشخص می‌کند. هنگامی که یک الگو شناسایی می‌گردد، ممکن است رویدادهای جدیدی که شامل خصوصیات و داده‌هایی غیر از

^۱ Aggregation
^۲ Abstraction
^۳ Disjunction
^۴ Conjunction
^۵ Sequence

خصوصیات و داده‌های مجموعه‌ی رویدادهای تشکیل دهنده‌ی آن است تولید شود و به ابر رویداد اضافه شود. مثلاً الگو بررسی موفقیت آمیز شکایت مشتریان در یک فروشگاه مطابق شکل (۲-۴) است [14].



شکل ۲-۴ الگوی بررسی موفقیت آمیز شکایت مشتریان در یک فروشگاه

قواعد یکی از روش‌های توصیف الگوها هستند که فهم و خوانایی بیشتری را فراهم می‌کنند [46]. هر الگو می‌تواند در قالب یک یا چند قاعده برای سامانه‌های پردازش رویدادهای پیچیده تعریف شوند. یک قاعده، متشکل از گونه‌های رویدادهایی هست که توسط عملگرهایی با یکدیگر در ارتباط هستند. همچنین رویدادها شامل محدودیت‌های زمانی نیز هستند. در زیر یک ساختار عمومی قواعد در سامانه‌های پردازش رویدادهای پیچیده را مشاهده می‌کنیم [19]:

```

PATTERN <OPERATOR (List of Event Types) and
          OPERATOR (List of Event Types) and
...>
WHERE <Event Value Constraints>
WITHIN <Time Window>
RETURN <Complex Event>
  
```

شکل ۲-۵ ساختار عمومی قواعد در سامانه‌های پردازش رویدادهای پیچیده

۲-۸ عملگر

الگوها در سامانه‌های پردازش رویدادهای پیچیده از یک رده نوع رویداد تشکیل شده‌اند که با عملگرها به یکدیگر مرتبط شده‌اند. به همین منظور لازم است عملگرهای ابتدایی را که از جزئی از قواعد محسوب می‌شوند را بشناسیم.

عملگر گزینش^۱ که معمولاً با نماد σ_θ نمایش داده می‌شود، وظیفه‌ی پایش رویدادهایی را دارد که شرط θ را ارضا نمی‌کنند. شرط عملگر گزینش، بسته به سامانه‌های مختلف و امکانات آنها در توصیف قواعد می‌تواند از نوع رویداد باشد تا نتیجه‌ی یک محاسبات ریاضی.

عملگر ترکیب عطفی که آن را با نماد $\&$ نمایش می‌دهند، بیانگر الزام وقوع همه‌ی رویدادها تشکیل دهنده‌ی آن است. مثلاً برای رویدادهای e_1, e_2, \dots, e_n ، ترکیب عطفی آن به معنای الزام وقوع e_1 تا e_n است. همچنین عملگر ترکیب فصلی که عموماً با نماد \parallel نمایش داده می‌شود، به معنی وقوع یکی از رویدادهای تشکیل دهنده‌ی آن است.

عملگر دنباله که با نماد \rightarrow مشخص می‌شود، بیانگر ترتیب رویدادها در یک قاعده است. مثلاً اگر قاعده‌ای دنبال کشف الگویی باشد که در آن افزایش دمای بدن یک بیمار دقیقاً پس از بالا رفتن قند خونس اتفاق افتاد، می‌تواند با استفاده از عملگر دنباله آن را توصیف کند. اگر رویداد e_1 معرف بالا رفتن قند خون و رویداد e_2 معرف بالا رفتن دما بدن باشد و متغیر start برای هر رویداد به معنای زمان تولید آن رویداد باشد، این الگو عبارت است از:

$$e_1 \rightarrow e_2 \equiv e_1 \& e_2 \text{ where } e_1.start < e_2.start$$

عملگر نفی^۳ که با نماد \neg به تصویر کشیده می‌شود، برای تشخیص عدم وقوع یک رویداد استفاده می‌شود. مثلاً اگر در یک خط منتاژ خودرو، هیچ خطایی در طول عمل مونتاژ رخ ندهد، یک رویداد پیچیده که منتج شده از عدم وقوع رویداد خرابی در قسمت‌های مختلف است، با عنوان مونتاژ بدون خطا به بخش مانیتورینگ ارسال می‌شود.

۹-۲ مدل زمانی

در سامانه‌های پردازش رویدادهای پیچیده، زمان نقش مهمی را ایفا می‌کند. اهمیت مدل زمانی در آن است که ترتیب ورود رویدادها را مشخص می‌کند. برچسب زمان یکی از خصیصه‌های رویداد است که بر حسب منطق هر سامانه، معنی متفاوت دارد. در برخی سامانه‌ها، برچسب زمانی بیانگر زمان تولید یک رویداد است. مثلاً زمانی که حسگری علائم حیاتی یک بیمار را اندازه‌گیری می‌کند، برچسب زمان همان

^۱ Selection

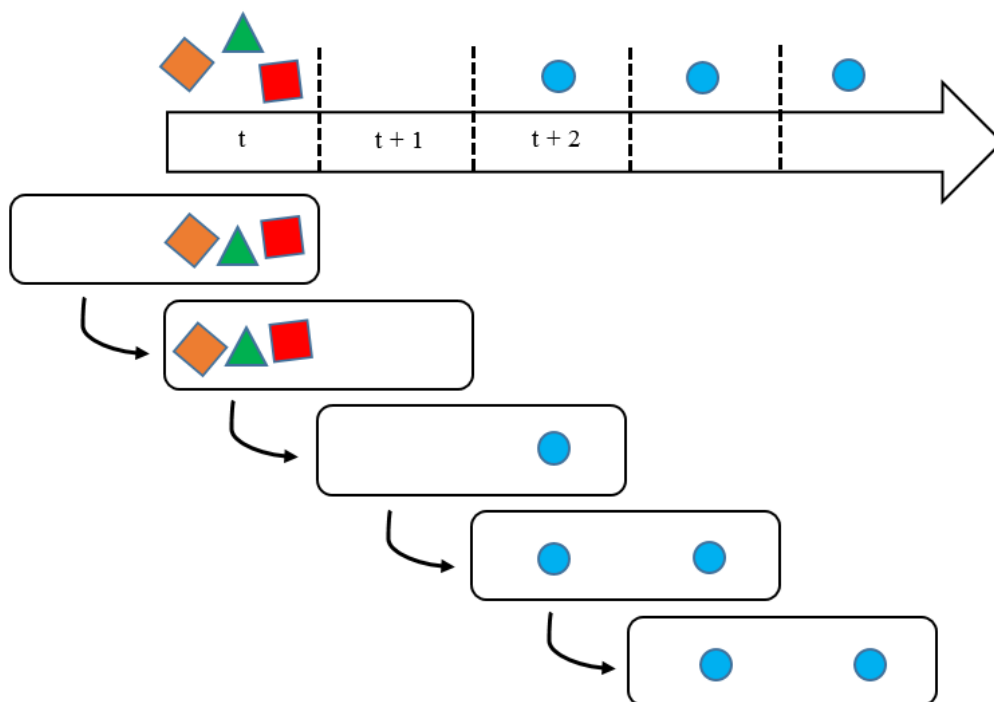
^۲ Filter

^۳ Negation

^۴ Timestamp

موقع را روی رویداد می‌زند. اما در برخی از سامانه‌ها، مقدار برچسب زمانی، معادل زمانی است که رویداد توسط سامانه مشاهده شده. باید تاکید کرد که در برخی از سامانه‌ها هر دو برچسب زمانی برای رویدادها وجود دارد.

مورد استفاده‌ی دیگر زمان در قواعد است. مدل معمولی که در قواعد برای زمان استفاده می‌شود، پنجره‌ی زمانی^۱ است. منظور از پنجره‌ی زمانی، زیر مجموعه‌ای از جریان رویدادها است که در آن محدوده‌ی زمانی قرار داشته باشد. مثلاً بررسی رویدادهای اطاق سرور در ده دقیقه‌ی گذشته. به این معنی که صحت این قاعده زمانی تشخیص داده می‌شود که الگوی آن از میان رویدادهایی باشد که در ده دقیقه‌ی گذشته وارد سامانه شده‌اند. در شکل (۶-۲) مثالی از پنجره‌ی زمانی را مشاهده می‌کنیم. هرچند در برخی سامانه‌ها طول پنجره را نه زمان، بلکه تعداد رویداد مشخص می‌کند. به این معنا که اگر مثلاً طول پنجره، ۱۰۰ رویداد باشد، الگوی مورد نظر از میان ۱۰۰ رویداد آخری که وارد سامانه شده است مورد بررسی قرار می‌گیرد.



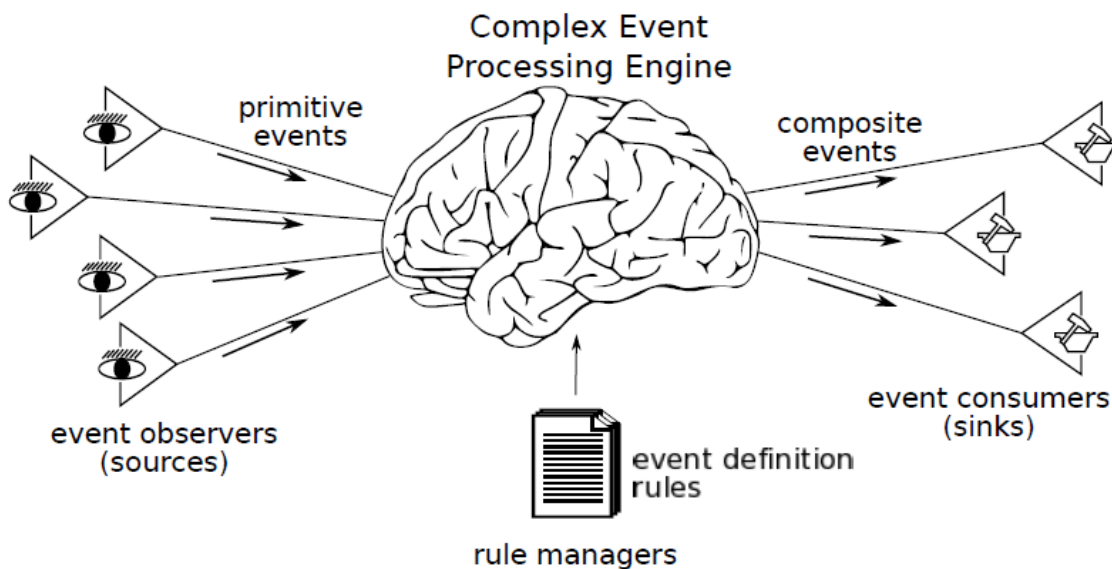
شکل ۶-۲ مفهوم پنجره‌ی زمانی

۲-۱۰ سامانه‌های پردازش رویدادهای پیچیده و موجودیت‌های آن

یک سامانه‌ی پردازش رویداد پیچیده همان‌طور که از شکل (۴-۲) نیز بر می‌آید، متشکل از موتور پردازش

^۱ Time window

رویدادهای پیچیده است که با استفاده از مخزن قواعد تعریف شده، رویدادهایی که از تولیدکنندگان رویداد وارد سامانه می‌شود را پردازش کرده و خروجی مورد نظر را به مصرف کنندگان آن ارسال می‌کند.



شکل ۲-۷ سامانه‌ی پردازش رویدادهای پیچیده [۸]

۲-۱۱ خلاصه

در این فصل، ابتدا شرح مختصری از تاریخچه‌ی سامانه‌های پردازش رویداد ارائه دادیم. در خلال این تاریخچه، پایگاه داده‌های فعال و سامانه‌های پردازش جریان داده‌ها را شناختیم و روندی که منجر به معرفی پردازش رویدادهای پیچیده شد را مطالعه کردیم. پس از آن نیز مفاهیم بنیادین پردازش رویدادهای پیچیده را تعریف کردیم.

فصل سوم:

شرح سامانه‌ی پیشنهادی

۳-۱ مقدمه

سامانه ی ارائه شده برای معیار سنجی سامانه های پردازش رویداد از ۴ قسمت مجزا تشکیل میشود. این بخش ها عبارتند از :

- تولید کننده ی فایل های رویداد
- ارسال کننده ی رویداد ها
- دریافت کننده ی رویداد ها
- مقایسه کننده ی فایل های رویداد

کاربر توسط تولید کننده ی فایل های رویداد، و با مشخص کردن انواع رویداد ها و ویژگی و نرخ ارسال آنها ، اقدام به تولید فایل های رویداد ها میکند. این فایل ها بعدا به وسیله ی ارسال کننده ی رویداد ها و بر اساس نرخ ارسالی مشخص شده توسط کاربر، میتوانند توسط شبکه به یک ماشین مقصد ارسال شود. دریافت کننده ی رویداد در ادامه، رویداد ها و زمان دریافت آنها را در حافظه ذخیره می نماید. پس از اتمام آزمایش، میتوان فایل اولیه رویداد های ارسال شده را با فایل رویداد های دریافت شده مقایسه نمود و علاوه بر مشاهده ی رویداد ها و مقادیر مرتبط به ویژگی های آن ها ، رویدادهای از دست رفته در حین ارسال و رویدادهایی که به طور اضافی دریافت شده اند را نیز مشاهده و بررسی کرد. در طول آزمایش میتوان عملکرد پردازشگر و حافظه را در هر دو طرف ارسال کننده و گیرنده را مشاهده و ذخیره نمود تا بعدا مورد بررسی قرار بگیرد. برنامه ها توسط زبان جاوا و با کمک کتاب خانه ی "protobuf" نوشته شده اند و این امر باعث می شود تا بتوان آن ها را به سهولت و بر روی اکثر سیستم عامل ها، به اجرا درآورد. این انتخاب به هدف رفع چالش استقلال از سیستم عامل صورت گرفته است.

در ادامه این بخش ها را جداگانه و به تفصیل شرح میدهم.

۳-۲ تولید کننده ی فایل های رویداد

تولید کننده ی فایل های رویداد با پذیرفتن مشخصات انواع رویداد ها و ماشین های مقصد ، اقدام به تولید فایل یا فایل هایی میکند که میتوانند در آینده توسط رشته های مجزا، به سمت مقصد خود ارسال شوند

Current Config Folder : <Not Selected>

Select Config Folder

Machines

Add Machine

Edit Machine

name	ip	port	split by type

Event Types

Add Event Type

Edit Event Type

name	rate	dispatch type	Attribute Count	Destination C...

Timestamp/Sequence

... Timestamp / Sequence ...

Timestamp Unit

... TimeStamp Unit ...

Serializer

... Serializer ...

Version

1

Serializer Version

1

Generate Config files

Generate Events

Time Limit

0

-

Count Limit

0

Load Configs

Close

STATUS :

...

شکل ۳-۱ نمایی از صفحه ی اصلی تولید کننده ی فایل رویداد

۳-۲-۱ ورودی ها

در این برنامه ورودی ها به سه بخش مهم تقسیم میشوند که به توضیح آنها میپردازیم :

- نوع رویداد ها
- ماشین های مقصد
- تنظیمات کلی

۳-۲-۱-۱ نوع رویداد

"نوع رویداد" در واقع قالبی است که رویداد ها از روی آن تولید می شود. کاربر میتواند انواع مختلفی از رویداد ها با خصوصیات مختلف تعریف کند. این خصوصیات شامل متغیر هایی مانند نرخ ارسال رویداد ها به مقصد، تعداد ویژگی های رویداد ها و توابع توضیح ریاضی تولید کننده ی مقادیر آنها ، ماشین های مقصد گیرنده ی این نوع رویداد، و نحوه ی اختصاص رویداد ها بین ماشین های مقصد هستند.

شکل ۳-۲ نمایش از پنجره ی تعریف یک نوع رویداد

نرخ، یک عدد میباشد که واحد آن میتواند رویداد بر ثانیه یا میلی ثانیه باشد. این واحد در صفحه ی اصلی

برنامه مشخص می‌گردد و بعداً در سرنوشته^۱ی فایل خروجی ذخیره می‌شود تا بتوان هنگام بازخوانی فایل از واحد زمانی نیز آگاه شد.

همچنین کاربر میتواند نوع اختصاص دهی رویداد ها به ماشین های مقصد را به صورت های مختلفی تعیین کند : عادی ، گردشی^۲ و یا با اختصاص دادن یک درصد به هر ماشین.

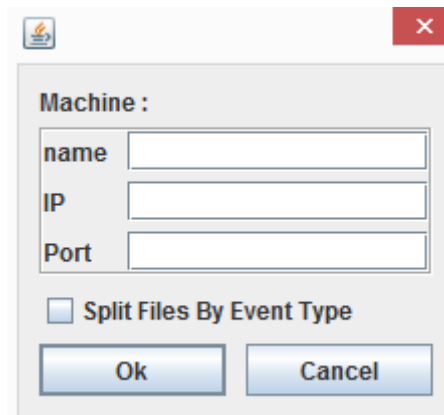
در قسمت بعدی کاربر می تواند ماشین های مقصدی که رویداد هایی از این نوع، باید به آن ها ارسال شوند را مشخص کند. در قسمت آخر نیز کاربر میتواند برای رویداد ها تعدادی ویژگی تعریف کند که مقدار آنها توسط توابع ریاضی نرمال، پواسن^۳، مثلثی ، یا نمایی^۴ و با دریافت ورودی های خاص این توابع مشخص میشود.

بدیهی است که تمامی رویداد های تولید شده از یک نوع رویداد، از خصوصیات کلی آن مانند نرخ و ماشین های مقصد تبعیت خواهند نمود.

۲-۱-۲-۳ ماشین های مقصد

در این بخش از برنامه، کاربر باید تمامی ماشین های مقصد و ویژگی های آنها را تعریف کند. این ماشین ها بعداً میتوانند در هنگام تعریف یک نوع رویداد، به عنوان ماشین مقصد آن رویداد ها انتخاب شوند. در این قسمت علاوه بر مشخص کردن آدرس شبکه و درگاه برای ارسال رویداد ها، این امکان وجود دارد که کاربر انتخاب کند رویداد های این ماشین خاص همگی در یک فایل ذخیره شده یا بر اساس نوع رویداد در فایل های جداگانه ذخیره شوند.

ذخیره سازی در فایل های جداگانه این مزیت را برای ما فراهم میکند که برای ارسال هر نوع رویداد خاص، یک رشته ی جداگانه در نظر بگیریم. این امر باعث تقسیم حجم پردازش مورد نیاز بر روی رشته های جداگانه میشود.



شکل ۳-۳-۳ نمایشی از پنجره ی افزودن ماشین مقصد

۳-۲-۱-۳ تنظیمات کلی

تنظیمات کلی شامل واحد زمانی که میتواند ثانیه یا میلی ثانیه باشد ؛ پروتکل ارسال و ذخیره ی رویداد ها که میتواند پروتکل جاوا یا پروتوباف آگول باشد، میشوند. همچنین کاربر این امکان را دارد تا رویداد ها را بر اساس نرخ های مشخص شده در قسمت تولید نوع رویداد ذخیره سازی کند، یا آنها را نادیده بگیرد. نادیده گرفتن این نرخ ها ، این امکان را می دهد تا کاربر بعدا و در هنگام ارسال رویداد ها ، نرخ ارسال را تعیین و بر اساس نیاز تغییر بدهد.

۳-۲-۲ فایل های خروجی

در نهایت و با فرمان کاربر ، فایل های رویداد به همراه دو فایل با پسوند XML. تولید میشوند. این فایل ها تنظیمات مربوط به انواع رویداد ها و همینطور تنظیمات مربوط به ماشین های مقصد را در بر دارند. در شکل (۳-۴) و (۳-۵) دو نمونه از این فایل ها مشاهده میشود.

```

<config>

<version>1</version>
<istimestamp>>false</istimestamp>
<unit>seconds</unit>
<serializer>protobuf</serializer>
<serializerVersion>1</serializerVersion>

<eventtype>
<name>e1</name>
<rate>10000.0</rate>
<dispatchtype>normal</dispatchtype>
<machines>
<machine><name>m4</name><percentage>100.0</percentage></machine>
</machines>
<attributes>
<attribute><name>1</name><distribution>normal 2 3</distribution></attribute>
</attributes>
</eventtype>

</config>

```

شکل ۳-۴ فایل xml شامل اطلاعات انواع رویداد ها

```

<machines>

<machine>
<name>m4</name>
<ip>127.0.0.1</ip>
<port>8989</port>
<eventfiles>split</eventfiles>
</machine>

</machines>

```

شکل ۳-۵ فایل xml شامل اطلاعات ماشین های مقصد

فایل های رویداد ساخته شده دارای ساختار مشخص شده ای میباشند تا علاوه بر حداقل حجم، بعداً بتوان رویداد ها و اطلاعات لازم را به راحتی از آنها بازخوانی کرد. هر فایل با سرنوشته ای شامل اطلاعاتی مانند واحد زمانی، نسخه ی برنامه و دیگر اطلاعات مهم شروع میشود. سپس رویداد ها در فایل نوشته میشوند. پیش از هر رویداد، یک برچسب زمانی از پیش تعیین شده برای ارسال آن رویداد نوشته میشود که توسط نرخ تعیین شده ی کاربر محاسبه شده است و مبدا زمانی آن * است. برنامه ی ارسال کننده ی رویداد ها پس از اجرا بر اساس این برچسب های زمانی شروع به ارسال رویداد ها خواهد کرد.

در صورتی که کاربر انتخاب کرده باشد تا نرخ ها را نادیده بگیرد، به جای این مقدار های زمانی، اعدادی مشخص کننده ی ترتیب ارسال رویداد ها در فایل نوشته میشوند تا ارسال کننده بعدا بتواند با نرخ تعیین شده، رویداد ها را ارسال نماید.

تصمیم کاربر مبنی بر ایجاد این کمیت ها در سرنوشته ی فایل و در قالب دو بیت ذخیره سازی میشود. شکل (۳-۶) نمایانگر ساختار یک فایل رویداد میباشد.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Version				Time stamp								Serializer Type			
Serializer Type				Serializer version								Is Exten.		Order T.	
Extension Size in bytes								Extension							
Extension															
3C								Event Size							
Event Size								AA							
Time Stamp/Seq number															
Time Stamp/Seq number															
Event															

شکل ۳-۶ نمایی از یک فایل رویداد

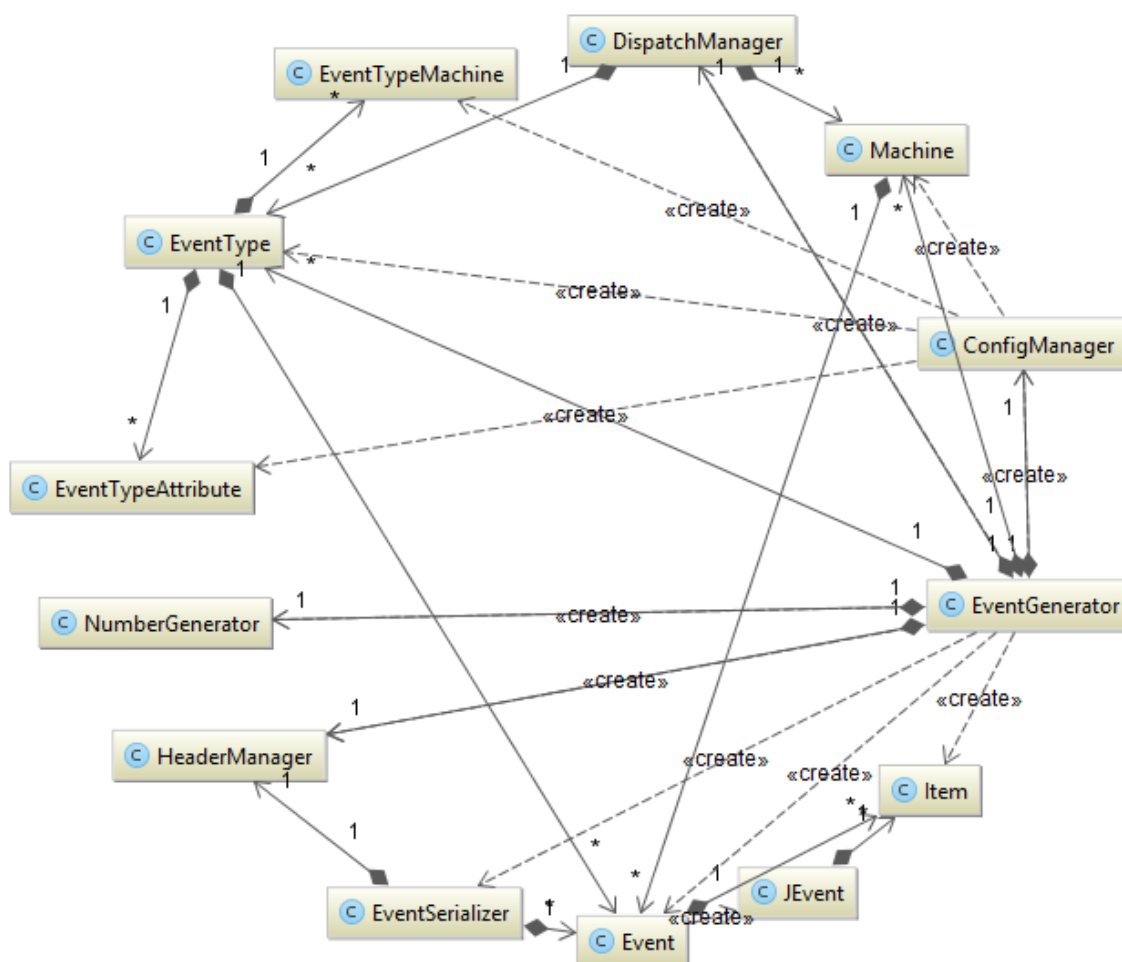
در شکل (۳-۷) کمیت های سرنوشته ی فایل ها شرح داده شده اند .

Header		
Title	Description	size
version	1.0	4 bits
Time stamp measure (mill sec or Micro Sec)	'AA' for milliseconds and '55' for seconds	1 byte
Serializer Type	'CC' for java serializer and '33' for protoBuff serializer	1 byte
Serializer version	For serializer version	1 byte
Is Extended	'01' for yes '10' for No	2bits
Order Type (Seq or time stamp)	'01' for sequence and '10' for timestamp	
Extension Size in bytes	It is optional and exist if is Extended equals AA	1 bytes
Extension if exist	It is optional and exist if is Extended equals AA	Up to 255 bytes

شکل ۳-۷ جدول عناصر مختلف موجود در سرنوشته ی فایل

فایل های ذخیره شده در نهایت با ساختار ("نام نوع رویداد" _ "نام ماشین مقصد") ذخیره سازی میشوند.

۳-۲-۳ بررسی فرآیند ها

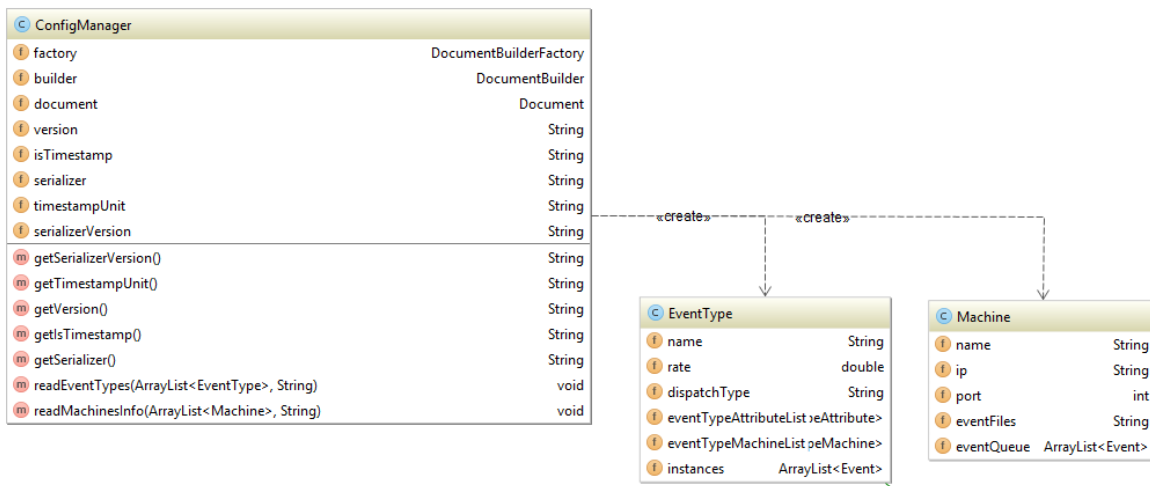


شکل ۳-۸ کلاس های تولید کننده ی فایل رویداد

در شکل (۳-۸) نمایی از تمامی کلاس های دخیل در برنامه ی تولید کننده فایل رویداد را مشاهده میکنیم. در ادامه به شرح فرآیند های رخ داده در این برنامه می پردازیم تا رابطه ی بین این کلاس ها را شفاف تر سازیم.

۳-۲-۳-۱ فرآیند خواندن تنظیمات

نوشتن و بازخوانی تنظیمات از روی فایل های XML توسط کلاس ConfigManager انجام میشود. این کلاس در واقع یک خواننده ی فایل های XML بوده و وظیفه ی تبدیل اطلاعات روی این فایل ها به اشیایی از کلاس های EventType و Machine را بر عهده دارد. این کلاس ها نمایانگر اطلاعات مربوط به "نوع رویداد" و "ماشین های مقصد" می باشند.

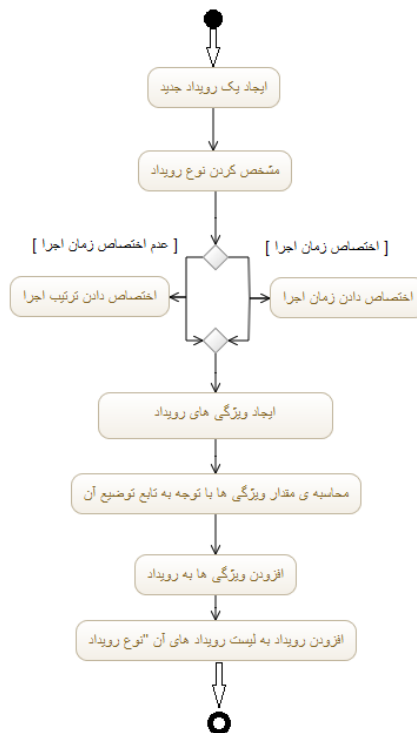


شکل ۳-۹ نمایشی از کلاس ConfigManager

همانطور که در شکل (۳-۹) مشاهده میشود، کلاس ConfigManager با فراخوانی متد های readMachineInfo و readEventType اقدام به خواندن فایل های XML کرده و سپس به ازاء هر ورودی در آن فایل ها، یک شیء از کلاس های EventType و Machine تولید میکند که در بر گیرنده ی اطلاعات مربوط به آن موجودی است. همچنین این کلاس تنظیمات کلی مانند واحد زمانی و نسخه ی نرم افزار را از فایل میخواند و در ویژگی های خود ذخیره میکند.

۳-۲-۳-۲ فرآیند تولید رویداد ها

با داشتن لیستی از انواع رویداد ها و ماشین های مقصد، میتوانیم به تولید رویداد ها بپردازیم. این فرآیند برای هر نوع رویداد یک لیست جداگانه تشکیل می دهد و بر اساس تنظیمات ، به آنها یک زمان مشخص برای اجرا، یا یک عدد برای نمایش ترتیب اختصاص می دهد.



شکل ۳-۱۰ نمودار مراحل تولید یک رویداد

در نهایت برای هر نوع رویداد یک لیست تولید میشود که شامل رویداد های تولید شده بر اساس آن می باشد. این لیست ها بر اساس زمان اجرا یا ترتیب اجرا مرتب سازی می شوند. لازم به ذکر است که در این پژوهش تمامی رویداد ها از کلاس JEvent تولید میشوند. تعریف این کلاس در شکل (۳-۱۱) مشهود است.

```

public class JEvent implements Serializable{
    private static final long serialVersionUID = 66L;
    public String name;
    public List<Item> items;
    public JEvent() { items = new ArrayList<Item>(); }
}
  
```

شکل ۳-۱۱ تعریف کلاس JEvent

۳-۲-۳-۳ فرآیند نوشتن رویداد ها بر روی فایل

با داشتن یک لیست از رویداد ها برای هر نوع رویداد، ماشین های مقصد هر نوع رویداد ، و در نظر گرفتن نحوه ی تقسیم رویداد ها بین ماشین ها ، رویداد ها به یک ماشین مقصد اختصاص داده میشوند. در نهایت بر اساس تنظیمات کاربر، برای هر ماشین مقصد یک یا چند فایل رویداد تولید میشود. کاربر

میتواند تعیین کند همه رویداد های آن ماشین در یک فایل ذخیره شوند یا اینکه هر نوع رویداد در یک فایل جدا گانه با نام "نوع رویداد"_"نام ماشین" ذخیره سازی شود.

تقسیم بندی بین ماشین ها به ۳ حالت امکان پذیر است :

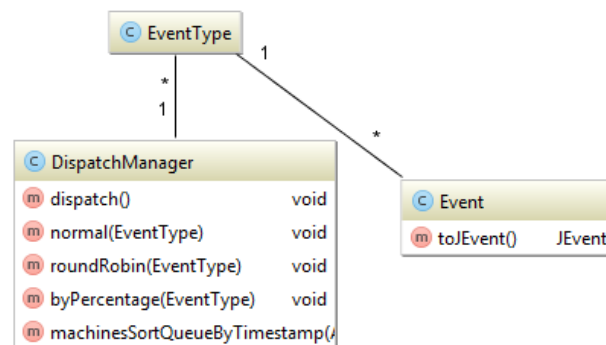
- حالت عادی: در این حالت هر رویداد به همه ی ماشین های مقصد ارسال میشود.
- حالت گردشی : در این حالت رویداد ها به ترتیب به ماشین های مقصد اختصاص داده میشوند .

```
int i = 0 ;
for(Event e: eventType.instances) {
    destMachines.get(i%destMachines.size()).eventQueue.add(e);
    i++;
}
```

شکل ۱۲-۳ کد مربوط به حالت اختصاص گردشی

- حالت درصدی : در این حالت کاربر برای هر ماشین مقصد یک درصد در نظر میگیرد و تعداد رویداد های انتصاب داده شده به هر ماشین متناسب با این درصد خواهد بود. این حالت نظیر حالت گردشی عمل میکند ولی هنگامی که یک ماشین به حداکثر تعداد رویداد خود رسید دیگر رویدادی به آن تعلق نگرفته و الگوریتم بین سایر ماشین ها گردش میکند تا زمانی که همه رویداد ها به یک مقصد اختصاص داده شده باشند. هدف از این کار این است که حداقل زمان ارسال بین رویداد ها نایل شود.

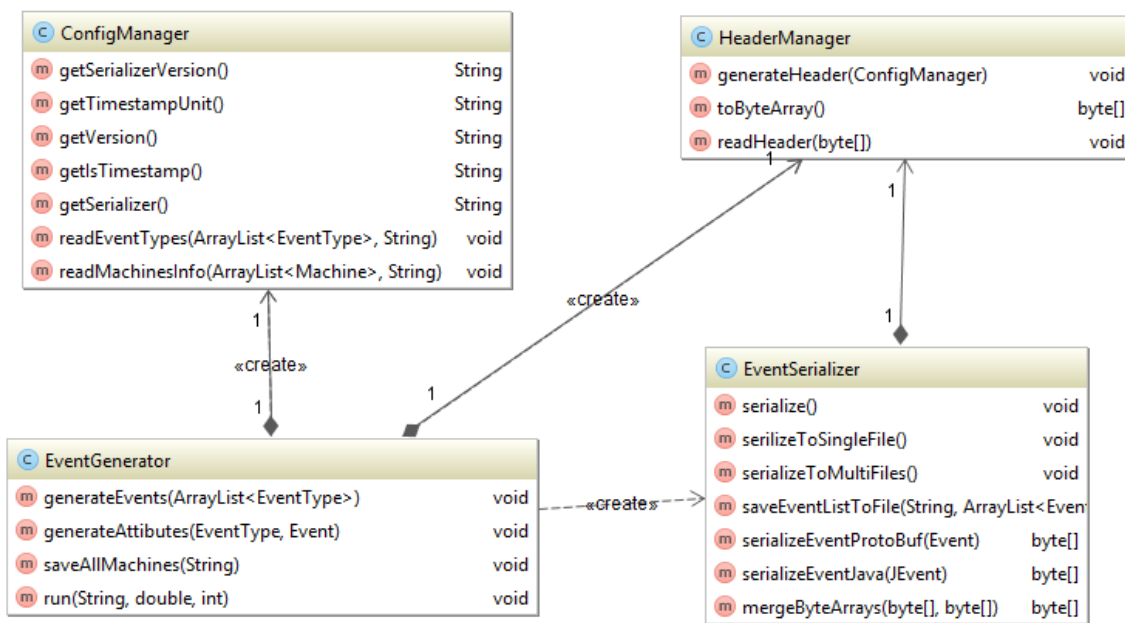
اختصاص رویداد ها به ماشین ها توسط کلاس Dispatcher صورت میگیرد. در شکل (۱۳-۳) نمایی از این کلاس نمایش داده شده است.



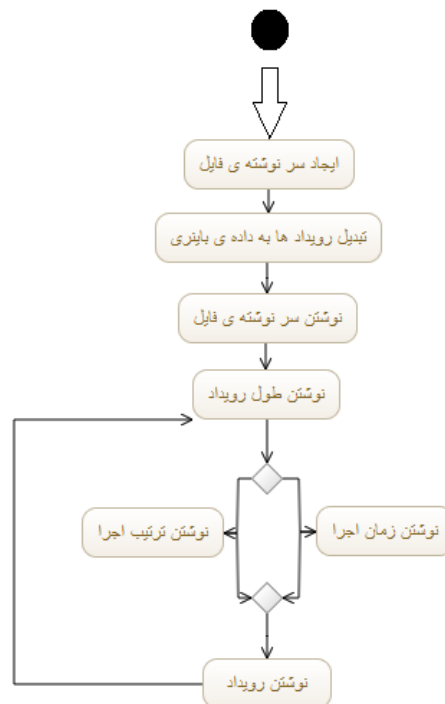
شکل ۱۳-۳ کلاس Dispatcher

در نهایت و با اختصاص دادن رویداد ها به ماشین های مقصد، عمل ایجاد فایل انجام میگیرد. ایجاد فایل ها توسط کلاس EventSerializer انجام میگیرد. این کلاس با دریافت یک ماشین مقصد، بر اساس تنظیمات آن ، اشیاء رویداد های اختصاص داده شده با آن ماشین را توسط پروتکل های پروتوباف یا جاوا (که انتخاب یکی از آنها بر عهده ی کاربر است) را به رشته های باینری قابل نوشتن بر روی فایل تبدیل می کند.

همچنین، کلاس HeaderManager ، تولید ، نوشتن و خواندن سرنوشته ی فایل را بر عهده دارد. این کلاس بوسیله ی تنظیمات خوانده شده از فایل های XML توسط کلاس ConfigManager، اقدام به تولید یک رشته ی باینری میکند که به عنوان سر نوشته در فایل ها قرار خواهد گرفت. این رشته شامل اطلاعاتی مانند واحد زمانی ، نسخه ی برنامه و غیره است که شرح کامل آن ها در شکل (۳-۶) آمده است. رابطه ی این کلاس ها با هم در شکل (۳-۱۴) مشخص است.



شکل ۳-۱۴ رابطه ی کلاس های دخیل در تولید فایل رویداد



شکل ۱۵-۳ مراحل تولید فایل رویداد

عمل تبدیل اشیاء رویداد ها به رشته های باینری، به دو صورت انجام میگیرد:

- توابع جاوا :
توابع جاوا، حالت پیشفرض تبدیل اشیاء جاوا به رشته های باینری هستند. این روش به دلیل سهولت در استفاده، و عدم نیاز به کتابخانه های خارجی پیاده سازی شده است ولی کارایی کمتری نسبت به پروتکل پروتوباف دارد.
- پروتکل پروتوباف :
پروتکل پروتوباف توسط شرکت گوگل و به صورت کد باز و برای زبان های مختلف ارائه میشود. و هدف آن انتقال اطلاعات بین برنامه های مختلف بدون وابستگی به زبان برنامه نویسی خاصی است. این ابزار دارای یک زبان ویژه است که ساختار زبانی خاص خود را دارد. کاربر با استفاده از این زبان یک پیغام را تعریف کرده، سپس بوسیله ی کامپایلر پروتوباف آن را کامپایل می کند. خروجی این کامپایلر کلاس هایی به زبان برنامه نویسی مورد نظر (مانند جاوا) می باشد که با استفاده از این کلاس ها و متد های آنها می توان اطلاعات مورد نظر را به راحتی به رشته ی باینری برای ذخیره بر روی فایل یا ارسال از طریق شبکه استفاده ، یا این اطلاعات را مجدداً از شبکه یا فایل بازخوانی کند. ویژگی مهم پروتوباف ، مستقل بودن از زبان برنامه نویسی است

بدین شکل که میتوان یک اطلاعات را با برنامه ای ارسال و بعدا به وسیله ی برنامه ای با زبان متفاوت از برنامه ی فرستنده بازخوانی کرد.

پروتوباف نسبت به توابع جاوا ، رشته های باینری با حجم بسیار کوچکتري تولید میکنند. نسبت حجم رویداد های تبدیل شده توسط پروتوباف، به رویداد های تولید شده توسط جاوا در این پژوهش ، در حدود یک هشتم است.

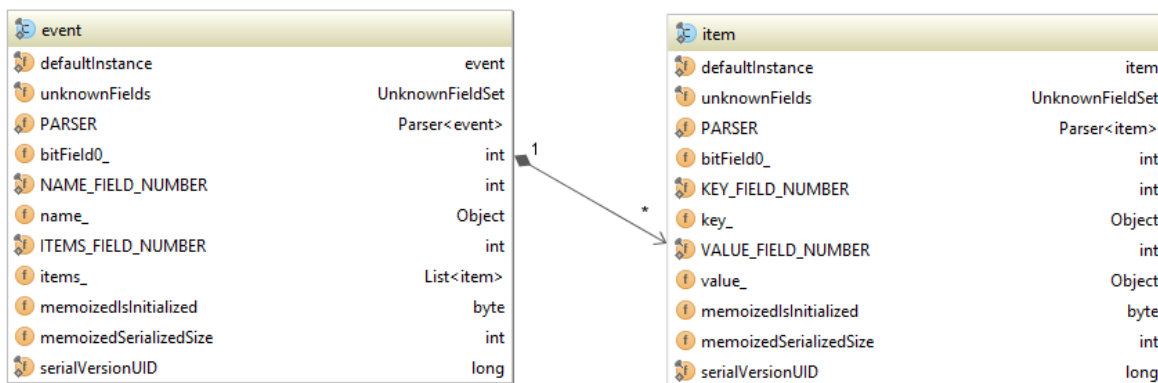
در شکل (۳-۱۶) ، کد پروتوباف مربوط به رویداد ها مشاهده میشود.

```
message event {
    required string name = 1;
    repeated item items = 2;
}

message item {
    required string key = 1;
    required string value = 2;
}
```

شکل ۳-۱۶ کد پروتوباف مربوط به رویداد ها

و در شکل (۳-۱۷) ،نمایی از دو کلاس تولید شده پس از کامپایل کد بالا مشاهده میشوند. (توجه شود که در تصور متد ها نمایش داده نشده اند).



شکل ۳-۱۷ کلاس های کامپایل شده توسط پروتوباف

برای استفاده از پروتکل پروتوباف، کافیت اشیا یی از کلاس های بالا ساخته و اطلاعات رویداد ها را به آنها منتقل کنیم. پروتوباف توابعی برای نوشتن یا خواندن این اشیاء به ما ارائه می دهد.

۳-۳ ارسال کننده ی رویداد ها

این برنامه برای کاربر این امکان را فراهم میکند تا با انتخاب یک ما شین مقصد، اقدام به ارسال فایل

های مرتبط با آن ماشین کند. لیست ماشین ها از فایل XML تولید شده در قسمت قبلی استخراج شده و فایل های مربوط به آن ماشین در پوشه ی انتخاب شده جستجو شده و انتخاب می شوند.

Config Directory: C:\Users\mohammadreza\Desktop\cep project Browse

Machines

name	ip	port	split by type
m4	127.0.0.1	8989	True

Event Files to be sent

- m4_e1
- m4_e2
- m4_e3

Rate: per Eventfile time unit ☐ Custom Rates

Send Event Files ☒ UDP ☐ TCP

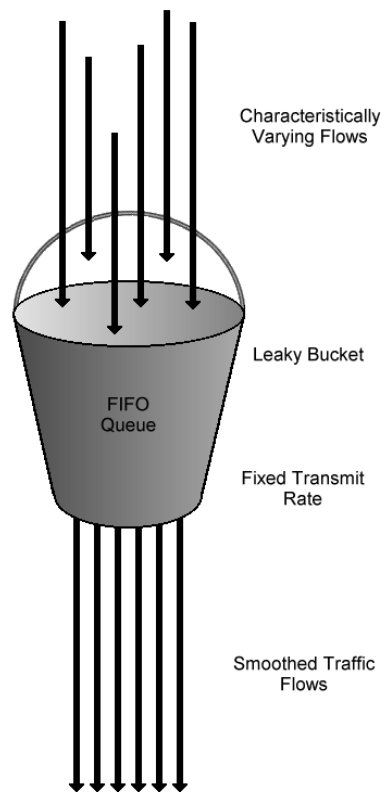
Status: ...

شکل ۳-۱۸ نمایی از ارسال کننده رویداد ها

کاربر میتواند توسط یکی از پروتکل های UDP یا TCP اقدام به ارسال رویداد ها به مقصد نماید. در صورتی که هنگام تولید فایل های رویداد، کاربر نرخ آن ها را نادید گرفته باشد، در این قسمت میتواند برای هر کدام از فایل ها یک نرخ جداگانه مشخص کند. از آنجا که هر فایل برای یک نوع از رویداد تولید شده است، کاربر در واقع برای هر نوع رویداد یک نرخ جداگانه مشخص میکند. با صدور فرمان ارسال رویداد ها، برای هر کدام از فایل ها یک رشته ی جداگانه تولید شده و به طور موازی شروع به ارسال رویداد ها می نمایند.

۳-۳-۱ الگوریتم سطل نشت دار^۱

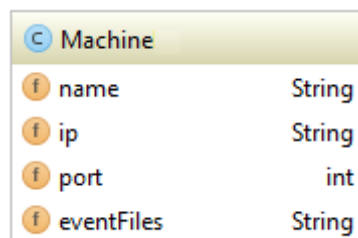
برای اطمینان از حفظ نرخ ارسال رویداد ها، از الگوریتم "سطل نشت دار" استفاده شده است. در این الگوریتم که به طور کلی برای حصول اطمینان از ثبات نرخ انجام یک فعالیت استفاده میشود، یک متغیر معرفی میشود که در هر چند ثانیه یک بار (بر اساس نرخ تعیین شده) یک واحد به آن اضافه میشود و با هر بار اجرا شدن قطعه کد یک واحد از آن کاسته میشود. اگر به هنگام اجرای کد ، متغیر به صفر رسیده بود ، رشته موقتا متوقف شده تا زمانی که متغیر مجددا افزایش یابد. مقدار این متغیر از ۱ بیشتر و از کمتر نمیتواند باشد و عمل افزایش آن، توسط رشته ی جداگانه ای صورت میگیرد. در صورتی که متغیر ۱ بود ، عمل افزایش صورت نخواهد گرفت. بدین ترتیب نرخ ارسال داده ها حفظ میشود.



شکل ۳-۱۹ نمایی از عملکرد الگوریتم سطل نشت دار

۳-۳-۲ فرآیند ارسال رویداد ها

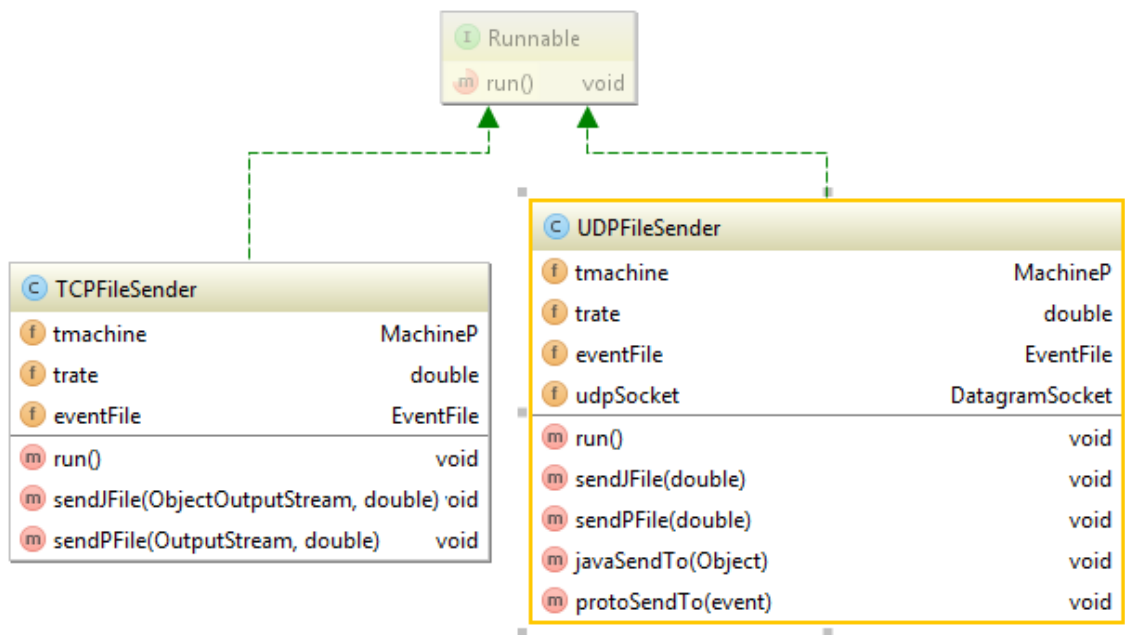
کاربر با استفاده از واسط گرافیکی یکی از پروتکل های UDP یا TCP را برای ارسال رویداد ها ، و یک پوشه ی کاری را انتخاب میکند. این پوشه شامل فایل های رویداد و همچنین فایل XML شامل اطلاعات ماشین های مقصد است که بوسیله ی برنامه ی تولید کننده ی فایل های رویداد ایجاد شدند. با انتخاب پوشه ی کاری ، لیستی از ماشین های مقصد به کاربر نشان داده میشود. کاربر از این لیست می تواند یک ماشین را انتخاب کند تا فایل های رویداد مربوط به آن توسط پروتکل انتخاب شده ارسال شوند. سپس با توجه به آدرس و درگاه ماشین مقصد که در فایل XML خوانده می شود ، یک شیء از کلاس Machine تولید میشود که شامل این اطلاعات است.



Machine	
f name	String
f ip	String
f port	int
f eventFiles	String

شکل ۳-۲۰ کلاس Machine

با انتخاب یک ماشین مقصد، تمامی فایل های رویداد مختص آن در پوشه ی کار جستجو شده و هر فایل توسط یک شیء از کلاس EventFile خوانده میشود. ارسال رویداد ها در این برنامه توسط دو کلاس TCPFileSender و UDPFileSender صورت می گیرد. به وسیله ی این کلاس ها، میتوان رشته های جداگانه ای برای ارسال رویداد ها به وسیله ی پروتکل TCP یا UDP ایجاد کرد .



شکل ۳-۲۱ کلاس های ارسال کننده ی رویداد

برنامه برای هر فایل رویداد ، یک رشته ی جداگانه به وسیله ی کلاس های TCPFileSender و UDPFileSender ایجاد می کند. هر رشته اطلاعات ماشین مقصد انتخاب شده را به عنوان ورودی دریافت کرده و پس از ایجاد سوکت مناسب ، شروع به ارسال رویداد ها به سمت آدرس و درگاه ماشین مقصد می کند.

هر رشته ، بر اساس سر نوشته ی فایل خوانده شده ، رویداد ها را بر اساس پروتکل جاوا یا پرتوباف به مقصد ارسال می کند.

۳-۴ دریافت کننده رویدادها

این برنامه با اجرا بر روی یک سیستم ، بر روی یک درگاه خاص تنظیم شده و به انتخاب کاربر به وسیله ی پروتکل های UDP یا TCP اقدام به دریافت رویداد های ارسال شده به آن سیستم میکند.

The screenshot shows a web form for configuring an event receiver. It has the following elements:

- Port:** A text input field containing the value "8989".
- Serializer:** A dropdown menu with "protobuf" selected.
- Destination Folder:** A text input field followed by a "browse" button.
- Save Events?:** A radio button.
- TCP/UDP:** Two radio buttons, with "UDP" currently selected.
- Event Count Limit:** A text input field containing the value "10000".
- Start Server:** A large blue button.
- Status:** A section at the bottom showing three red dots "...".

شکل ۳-۲۲ نمایی از دریافت کننده ی رویداد ها

کاربر همچنین این امکان را دارد تا با مشخص کردن یک پوشه، اقدام به ذخیره سازی رویداد ها ی دریافت شده بر روی فایل نماید. فایل ایجاد شده از استاندارد معرفی شده در (۳-۲-۴) پیروی میکند. در صورت استفاده از پروتکل TCP ، به ازای هر رشته ی ارسال کننده ، یک رشته ی دریافت کننده به وجود خواهد آمد و رویداد ها به طور موازی دریافت میشوند. در نهایت این رویداد ها در فایل هایی مجزا به ازای هر رشته ذخیره سازی خواهند شد.

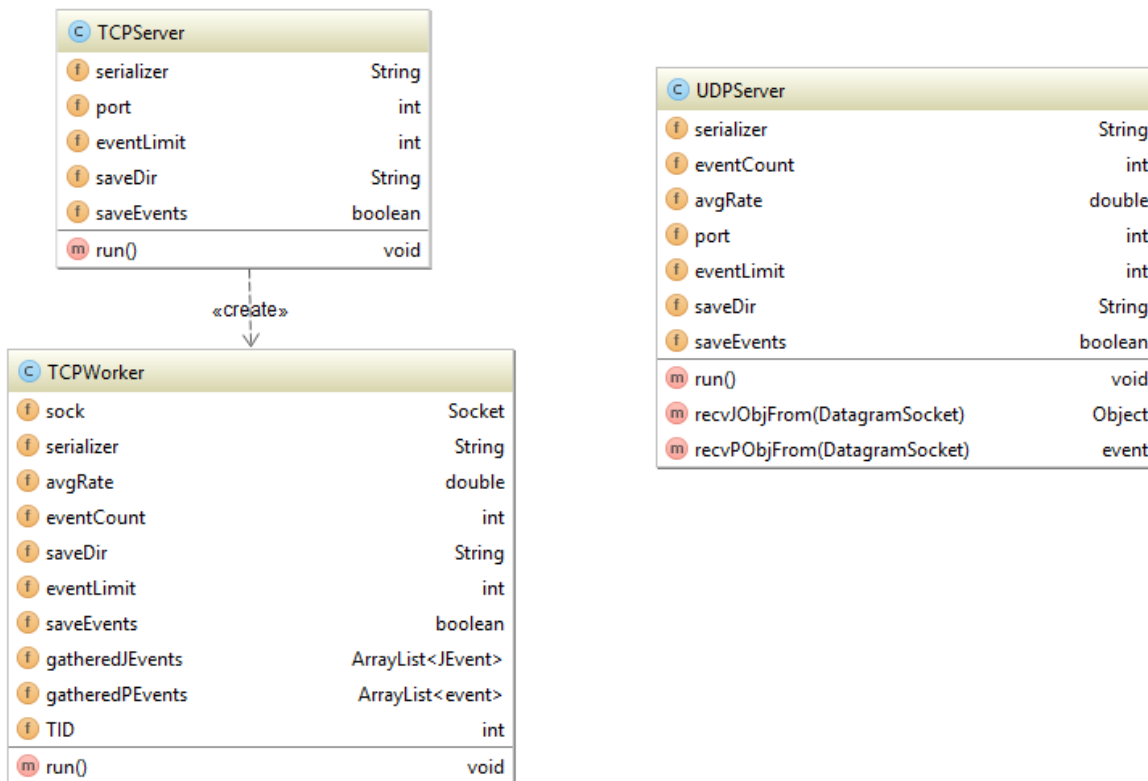
۳-۴-۱ فرآیند دریافت رویداد ها

دریافت رویداد ها، با توجه به پروتکل UDP و TCP متفاوت می باشد. در صورت انتخاب پروتکل UDP ، برنامه صرفا اقدام به ایجاد یک سوکت UDP بر روی درگاه مشخص شده کرده و شروع به خواندن رویداد های رسیده به آن می کند. این عملیات توسط یک رشته ی جدید به وسیله ی کلاس UDPServer انجام میگردد.

در صورت انتخاب پروتکل TCP به دلیل نیاز به برقراری ارتباط، برنامه با ایجاد یک شیء از کلاس TCPServer اقدام به ایجاد یک سوکت سرور TCP میکند. سپس، به ازای هر درخواست ارتباط، ارتباط را قبول کرده ، یک سوکت جدید برای آن ارتباط ایجاد کرده و سپس به وسیله ی کلاس TCPWorker یک

رشته ی مختص به آن سوکت ساخته تا شروع به خواندن رویداد های رسیده کند. این امر سبب می شود تا در صورتی که چندین رشته ی ارسال کننده درخواست ارتباط با سرور را داشتند ، به ازای هر رشته ی ارسال کننده یک رشته ی دریافت کننده ایجاد شود. در نهایت در هر دو حالت ، در صورت انتخاب کاربر، رویداد های جمع آوری شده در یک فایل به ازاء هر رشته، در آدرس داده شده به برنامه ذخیره می شوند.

فرآیند ذخیره سازی رویداد ها ، همانند فرآیند تشریح شده در (۳-۳-۲-۳-) می باشد. با این تفاوت که هر رشته ی دریافت کننده، به محض دریافت اولین رویداد یک شمارنده را آغاز کرده و زمان دریافت هر رویداد را ، به جای زمان یا ترتیب ارسال رویداد در فایل ذخیره میکند.



شکل ۳-۲۳ کلاس های مسئول دریافت رویداد ها

۳-۵ مقایسه کننده ی فایل های رویداد

کاربر با استفاده از این برنامه میتواند دو فایل رویداد را با هم مقایسه کرده و علاوه بر رویداد های درون آنها، رویداد های متفاوت بین آنها را نیز مشاهده کند. کاربرد این برنامه در بررسی دو فایل رویداد ، یکی فایل ارسال شده و دیگری فایل دریافت شده است تا بتوان رویداد هایی که در شبکه از دست رفته اند یا به نحوی اشتباه ارسال شده اند را یافت.

EventFile1 C:\Users\mohammadreza\Desktop\lcep project\m4_e3 Browse

EventFile2 C:\Users\mohammadreza\Desktop\lcep project\m4_e1 Browse

EventFile 1

e3 : 1->0.8437100624733018|
 e3 : 1->2.7541566330565934|
 e3 : 1->1.2924352975761235|
 e3 : 1->3.640713123178685|
 e3 : 1->3.903508395864407|
 e3 : 1->2.9375724222172317|
 e3 : 1->1.8363596195832113|

EventFile 2

e1 : 1->2.315498856637692|
 e1 : 1->2.412507732182266|
 e1 : 1->3.6837233210229106|
 e1 : 1->1.7247442057145552|
 e1 : 1->1.0193231559783098|
 e1 : 1->1.1354965536883381|
 e1 : 1->0.7866542961622094|

Compare

Total Events in 1	10	Total Events in 2	10
Exclusive Events in 1	10	Exclusive Events in 1	10

e3 : 1->0.8437100624733018|
 e3 : 1->2.7541566330565934|
 e3 : 1->1.2924352975761235|
 e3 : 1->3.640713123178685|
 e3 : 1->3.903508395864407|
 e3 : 1->2.9375724222172317|
 e3 : 1->1.8363596195832113|

e1 : 1->2.315498856637692|
 e1 : 1->2.412507732182266|
 e1 : 1->3.6837233210229106|
 e1 : 1->1.7247442057145552|
 e1 : 1->1.0193231559783098|
 e1 : 1->1.1354965536883381|
 e1 : 1->0.7866542961622094|

Status ***

شکل ۳-۴۴ نمایی از مقایسه کننده فایل های رویداد

۳-۵-۱ فرآیند مقایسه ی فایل ها

در این برنامه، پس از انتخاب دو فایل رویداد توسط کاربر، برنامه به وسیله ی دو شیء از کلاس EventFile شروع به استخراج رویداد ها از فایل های رویداد می کند و آنها را در دو لیست جداگانه ذخیره سازی می کند.

سپس ، با صدور دستور مقایسه ی فایل ها ، دو لیست رویداد با هم مقایسه شده و سپس رویداد های انحصاری آنها نمایش داده میشوند. با توجه به اینکه دو لیست هیچ ترتیب خاصی ندارند و تمامی رویداد ها باید با هم مقایسه شوند ، پیچیدگی محاسبات با مقیاس $O(n^2)$ خواهد بود.

۳-۶ خلاصه

در این فصل، اجزا مختلف سامانه ی پیشنهادی به ترتیب استفاده شرح داده شدند. همچنین کلاس های مهم دخیل در فعالیت آنها معرفی شده و فرآیند های آنها به همراه الگوریتم ها و کتابخانه های مهم مورد استفاده ی آنها بررسی شدند.

فصل چهارم:

آزمایشها و مشاهدات

۴-۱ آزمایش ها

۴-۱-۱ مقدمه

برای بررسی کارکرد سیستم معرفی شده ، از یک آزمایش استفاده شد. در این آزمایش اعمال ارسال و دریافت داده رویداد، در دو رایانه انجام میگیرند. هدف از آزمایش ، اطمینان از کارکرد برنامه ها و بررسی محدودیت های آنها می باشد. برنامه ها باید با موفقیت و طبق تنظیمات کاربر، به درستی اقدام به ارسال و دریافت رویداد ها نمایند.

۴-۱-۲ محیط آزمایش

در آزمایش ها ، از دو رایانه ی قابل حمل یکسان استفاده شد. مشخصات این رایانه ها در زیر ذکر می شود.

- تولید کننده : شرکت MSI
- مدل : GS60
- پردازنده : Intel Core i7 4710HQ
- تعداد هسته های پردازنده : ۴
- حجم حافظه : ۱۶ گیگابایت
- کارت شبکه : Killer e2200 Gigabit Ethernet Controller

۴-۱-۳ شرح آزمایش

در این آزمایش ها ، با ایجاد یک ماشین مقصد، تعداد ۵ نوع رویداد را به آن اختصاص دادیم. آدرس ماشین مقصد را آدرس رایانه ی مسئول گرفتن رویداد ها در نظر گرفتیم.

به هر رویداد ۵ ویژگی با مقداری تصادفی اختصاص دادیم. پروتکل باینری سازی رویداد ها را پروتکل پروتوباف در نظر گرفتیم و فایل های رویداد را به ازاء هر نوع رویداد جدا کردیم تا رشته های مجزا برای ارسال و دریافت آنها تولید شوند.

پس از تولید فایل ها ، اقدام به ارسال آنها با برنامه ی ارسال کننده ی رویداد ها نمودیم. در این برنامه از هر دو پروتکل TCP و UDP برای ارسال رویداد ها استفاده شد.

در قسمت گیرنده ی رویداد ها ، گزینه ی مربوط به ذخیره ی رویداد ها در فایل را فعال کردیم و در نهایت فایل های رویداد را توسط برنامه ی مقایسه کننده ی رویداد ها بررسی نمودیم.

در این آزمایش ها ، نرخ رویداد ها را متغیر در نظر گرفته و در طول آزمایش تغییر دادیم.

مدت زمان تمامی آزمایش ها را به دلیل جلوگیری از ایجاد فایل های رویداد حجیم ، ۱۰ ثانیه در نظر گرفتیم.

۴-۱-۴ مشاهدات و نتایج

در شروع آزمایش ، نرخ ارسال همه ی فایل های رویداد را ۵۰ رویداد در ثانیه در نظر گرفتیم. هدف از این نرخ پایین، اطمینان از کارکرد برنامه ها بدون در نظر گرفتن بازدهی پردازنده و بار شبکه بود.

کارکرد برنامه ها در این نرخ پایین مطابق انتظار بود و تمام رویداد ها بدون اتلاف به گیرنده منتقل شدند.

همچنین با بررسی زمان های دریافت در فایل های گرفته شده، نرخ دریافت را بررسی کردیم. خطای این نرخ بسیار کم و و نادید گرفتنی بود.

با افزودن نرخ ارسال به ۱۰۰۰۰ رویداد در ثانیه (مجموعاً ۵۰۰۰۰ رویداد در ثانیه توسط همه ی رشته ها) ، در پروتکل UDP ، ۶,۳٪ از رویداد های ارسال شده در حین ارسال از دست رفتند. در پروتکل TCP ، تمامی رویداد ها به درستی دریافت شدند. دلیل این تفاوت ، عدم ارائه ی اطمینان از دریافت دیتاگرم ها توسط پروتکل UDP می باشد. دیتاگرم ها به دلایل مختلف مانند پر بودن بافر ، مشغول بودن پردازنده و یا مشکلات مربوط به شبکه، ممکن است در حین ارسال از دست بروند. در بررسی نرخ دریافت رویداد ها ، تفاوتی در حدود ۹٪ با نرخ ارسال مشاهده شد. در بررسی بیشتر مشاهده شد که این تفاوت به دلیل افزایش بار پردازشی و تعداد تعویض متن ها توسط پردازنده می باشد. این خطا برای هر دو پروتکل TCP و UDP تقریباً یک سان بود.

در نهایت ، با افزودن نرخ ارسال به ۳۰۰۰۰ رویداد در ثانیه (مجموعاً ۱۵۰۰۰۰ رویداد در ثانیه) ، پردازنده ی ما به حالت مشغولیت کامل رسید. پس از این محدودیت، با افزایش بیشتر نرخ ارسال ، نرخ دریافت بسیار نامنظم شد و خطای آن بالا رفت. همچنین تعداد رویداد های از دست رفته پروتکل UDP افزایش بیشتری داشت و تا ۳۰٪ رسید. در پروتکل TCP نیز، با وجود ارسال تمامی رویداد ها، نرخ دریافت دچار آشفتگی زیادی شد تا حدی که گاهی مواقع بین دریافت دو رویداد چند ثانیه فاصله مشاهده می شد و بالعکس، برخی رویداد ها بدون فاصله و به صورت انفجاری دریافت شده بودند.

با آزمایش بیشتر مشاهده شد که پردازنده ی ما توان ارسال بیشتر از در حدود ۱۳۰۰۰۰ رویداد در ثانیه را ندارد و در صورت افزایش نرخ ارسال از این حد ، شاهد افت کارایی شدید خواهیم بود.

در نهایت، و با مشاهدات فوق ، می توان به این نتیجه رسید که علی رغم خطاهای ذکر شده، در صورت بالا بردن قدرت سخت افزاری و پهنای باند شبکه، میتوان سقف نرخ رویداد را افزایش داد و در عین حال مقادیر خطا را نیز کاهش داد.

فصل پنجم:

نتیجه‌گیری و کارهای آینده

۵-۱ نتیجه گیری

در این پژوهش ، تلاش شد تا با غلبه بر چالش هایی که به هنگام معیار سنجی یک سامانه ی پردازش رویداد پیچیده با آن ها رو برو هستیم، اقدام به ارائه ی ابزاری نماییم که بتواند به عنوان یک ابزار استاندارد و با استقلال از سامانه ی پردازش رویداد و یا سیستم عامل خاصی ، قادر به بررسی کارایی یک سامانه ی پردازش رویداد پیچیده باشد.

نتیجه ی این پژوهش، مجموعه ای از ابزار ها بود تا بتواند بر محدودیت های موجود در زمینه ی پردازش رشته ای رویداد ها غالب شده و برای ما امکان سنجش کمیت های مهمی که بر کارایی یک سامانه ی پردازش رویداد دلالت می کنند را فراهم کند. با کمک این ابزار ها قادر خواهیم بود تا در یک شرایط کاربردی، بین سامانه های پردازش رویداد مختلف ، سامانه ای با بهترین بازدهی را انتخاب نماییم. این ابزار ها شامل چهار بخش اصلی می شدند:

- نرم افزار تولید کننده ی فایل های رویداد ، که وظیفه ی تولید مجموعه ای از رویداد ها و ذخیره ی آنها بر روی فایل های مجزا را داشت. این کار این امکان را فراهم می کرد تا یک آزمایش بتواند با کم ترین تغییر در محیط های متفاوت اجرا شود.
- نرم افزار ارسال کننده ی رویدادها، که وظیفه ی خواند و ارسال فایل های رویداد به یک ماشین مقصد بر اساس نرخ های مشخص شده را بر عهده داشت.
- نرم افزار دریافت کننده ی رویدادها ، که وظیفه ی دریافت رویداد های ارسال شده به یک ماشین ، و ثبت زمان دریافت آنها را بر عهده داشت.
- نرم افزار مقایسه کننده ی فایل های رویداد، که وظیفه ی مقایسه ی دو فایل رویداد را ، به منظور بررسی میزان تفاوت بین آنها را بر عهده داشت.

بوسیله ی این ابزار ها قادر شدیم تا کمیت هایی مانند نرخ دریافت و ارسال ، و رویداد های از دست رفته در حین فرآیند ارسال را با دقت اندازه گیری نماییم. این کمیت ها به ما معیار مناسبی به منظور معیارسنجی و مقایسه سامانه های پردازش رویداد پیچیده ارائه میدهند و میتوانند به عنوان یک چهارچوب استاندارد برای این منظور مورد استفاده قرار گیرند.

۵-۲ کارهای آینده

در آینده در نظر داریم تا علاوه بر افزودن امکانات بیشتر به این مجموعه ابزار، سعی در افزایش کارایی آن نماییم.

در این راستا چند هدف را مطرح می کنیم:

- ۱- افزایش تعداد پروتکل های باینری سازی رویداد ها و امکان استفاده از پروتکل هایی مانند JSON.
- ۲- بهبود نحوه ی استفاده از پردازش گر به وسیله ی برنامه نویسی چند هسته ای به جای برنامه نویسی چند رشته ای .
- ۳- افزودن قابلیت تغییر نرخ ارسال رویداد ها در حین ارسال.
- ۴- افزودن قابلیت افزودن و تغییر رویداد ها در حین ارسال.
- ۵- استفاده از ساختار داده های بهینه تر.
- ۶- استفاده از قابلیت های پردازشی کارت گرافیک.

- [1] N. P. Schultz-Møller, "Distributed detection of event patterns," M.S Thesis, computing department, Imperial College, London, 2008.
- [2] W. Van Der Aalst, "Process mining: discovery, conformance and enhancement of business processes," published by Springer-Verlag Berlin Heidelberg, 2011.
- [3] T. Heinze, L. Aniello, et al, "Cloud-based Data Stream Processing," in ACM international conference on distributed event-based systems, 2014.
- [4] M. Hirzel, "Partition and compose: parallel complex event processing," in ACM international conference on distributed event-based systems, 2012.
- [5] A. Margara, G. Cugola, "High-performance publish-subscribe matching using parallel hardware," in IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 1, pp. 126 – 135, 2014.
- [6] C. Mutschler, M. Philippsen, "Distributed low-latency out-of-order event processing for high data rate sensor streams," in IEEE International symposium on Parallel & distributed Processing, 2013.
- [7] K. Isoyama, Y. Kobayashi, et al, "A scalable complex event processing system and evaluations of its performance," in ACM international conference on distributed event-Based systems, 2012.
- [8] G. Cugola, A. Margara, et al, "Introducing uncertainty in complex event processing: model, implementation, and validation," in Journal of Computing, vol. 97, no. 2, pp. 103 – 144, 2015.
- [9] B. Ottenwälder, B. Koldehofe, et al, "MCEP: a mobility-aware complex event processing system," in ACM Transactions on Internet Technology, vol. 14, no. 1, pp. 1 – 6, 2014.
- [10] M. D. Assunção, R. N. Calheiros, "Big data computing and clouds: trends and future directions," in journal of Parallel and Distributed Computing, vol. 79, pp. 3 – 15, 2015.
- [11] G. Cugola and A Margara, "Processing flows of information: From data stream to complex event processing," in ACM computing surveys, vol. 44, no. 3, pp. 1 – 70, 2012.
- [12] A. Thakkar, "On scaling complex event processing," M.S Thesis, Department of Computer Science, Indian Institute of Technology, Bombay, 2013.
- [13] F. Nguyen, D. Tovarňák, and T. Pitner, "Semantically partitioned peer to peer complex event processing," in intelligent distributed computing VII, Springer International Publishing, pp. 55 – 65, 2014.
- [14] D. Luckam and W. R. Schulte, "EPTS event processing glossary – version 2.0," 2011.

-
- [15] O. Etzion, P. Niblett, "Event processing in action," Manning Publications Company, 2010.
 - [16] A. Adi, and O. Etzion, "Amit-the situation manager," in The International Journal on Very Large Data Bases, vol. 13, no. 2, pp. 177 – 203, 2004.
 - [17] L. Liu and M. T. Zsu, "Encyclopedia of database systems," Springer Publishing Company, 2009.
 - [18] A. Margara and G. Salvaneschi, "Ways to react: comparing reactive languages and complex event processing," in REM, 2013.
 - [19] I. Flouris, N. Giatrakos, et al, "Issues in complex event processing systems," in IEEE Big Data Conference, 2015.
 - [20] K. Sun, Y. Wang, Y., and S. Peng, "Distributed complex event processing using rule deployment," in international conference on education, management and computing technology, 2015.
 - [21] D. McCarthy and U. Dayal, "The architecture of an active database management system," in ACM Sigmod Record, vol. 18, no. 2, pp. 215 – 224, 1989.

پوست

پیوست الف: واژه‌نامه فارسی به انگلیسی

کلمه انگلیسی	معادل به فارسی
Abstraction	انتزاع
Event Cloud	ابر رویداد
Subscribe	اشتراک
Join	الحاق
Pattern	الگو
Publish	انتشار
Internet of Things	اینترنت اشیاء
Retrieve	بازیابی
Partitioning	بخش‌بندی
Timestamp	برچسب زمانی
Bin Packing	بسته‌بندی اقلام
Filter	پالایش
Observation	پایش
Active Database	پایگاه داده فعال
Event Dispatcher	پخش‌کننده‌ی رویداد
Information Flow Processing	پردازش جریان اطلاعات
Data Stream Processing	پردازش جریان داده‌ها
Complex Event Processing	پردازش رویدادهای پیچیده
Time Window	پنجره‌ی زمانی
Rule Decomposition	تجزیه‌ی قاعده
Aggregation	تجمع
Fault Tolerance	تحمل‌پذیری خطا
Partial Order	ترتیب جزئی
Linear Order	ترتیب خطی
Conjunction	ترکیب عطفی
Disjunction	ترکیب فصلی
Partial Match	تشابه جزئی
Adaptability	تطبیق‌پذیری
Load Balancing	تعادل بار
Interaction	تعامل
Variety	تنوع
Throughput	توان عملیاتی، گذردهی

معادل به فارسی	کلمه انگلیسی
توزیع شده	Distributed
توزیع قواعد	Rules Distributing
توسعه پذیری	Scalability
تولید کننده ی رویداد	Event Producer
جایابی	Placement
جدایی کامل	Decoupling
جریان	Stream
حوزه ی کاربرد	Application Domain
داده های عظیم	Big Data
دنباله	Sequence
رویداد	Event
رویداد پیچیده	Complex Event
رویداد ساده	Simple Event
زمان پاسخ	Response Time
سازگاری	Consistency
سامانه های سایبری-فیزیکی	Cyber-Physical System
سرعت	Velocity
صحت	Veracity
علیت	Causality
قاعده	Rule
کاذب مثبت	False Positive
کاذب منفی	False Negative
کارآمدی	Performance
کشیدن	Pull
گزینش	Selection
گلوگاه	Bottleneck
مبتنی بر رویداد	Event Driven
متمرکز	Centralized
مصرف کننده ی رویداد	Event Consumer
موتور پردازش	Processing Engine
نامتجانس	Heterogeneous
ناهمگام	Asynchronous
نرخ	Rate
نفی	Negation
نوبت گردشی	Round Robin

کلمه انگلیسی	معادل به فارسی
Situation	وضعیت

پیوست ب: واژه‌نامه انگلیسی به فارسی

کلمه انگلیسی	معادل به فارسی
Abstraction	انتزاع
Active Database	پایگاه داده فعال
Adaptability	تطبیق‌پذیری
Aggregation	تجمع
Application Domain	حوزه‌ی کاربرد
Asynchronous	ناهمگام
Big Data	داده‌های عظیم
Bin Packing	بسته‌بندی اقلام
Bottleneck	گلوگاه
Causality	علیت
Centralized	متمرکز
Complex Event	رویداد پیچیده
Complex Event Processing	پردازش رویدادهای پیچیده
Conjunction	ترکیب عطفی
Consistency	سازگاری
Cyber-Physical Systems	سامانه‌های سایبری-فیزیکی
Data Stream Processing	پردازش جریان داده‌ها
Decoupling	جدایی کامل
Disjunction	ترکیب فصلی
Distributed	توزیع شده
Event	رویداد
Event Cloud	ابر رویداد
Event Consumer	مصرف کننده‌ی رویداد
Event Dispatcher	پخش کننده‌ی رویداد
Event Producer	تولید کننده‌ی رویداد
False Negative	کاذب منفی
False Positive	کاذب مثبت
Fault Tolerance	تحمل‌پذیری خطا
Event Driven	مبتنی بر رویداد
Filter	پالایش
Heterogeneous	نامتجانس
Information Flow Processing	پردازش جریان اطلاعات

کلمه انگلیسی	معادل به فارسی
Interaction	تعامل
Internet of Things	اینترنت اشیاء
Join	الحاق
Linear Order	ترتیب خطی
Load Balancing	تعادل بار
Negation	نفی
Observation	پایش
Partial Match	تشابه جزئی
Partial Order	ترتیب جزئی
Partitioning	بخش‌بندی
Pattern	الگو
Performance	کارآمدی
Placement	جایابی
Processing Engine	موتور پردازش
Publish	انتشار
Pull	کشیدن
Push	هل دادن
Rate	نرخ
Response Time	زمان پاسخ
Retrieve	بازیابی
Round Robin	نوبت گردشی
Rule	قاعده
Rule Decomposition	تجزیه‌ی قاعده
Rules Distributing	توزیع قواعد
Scalability	توسعه‌پذیری
Selection	گزینش
Sequence	دنباله
Simple Event	رویداد ساده
Situation	وضعیت
Stream	جریان
Subscribe	اشتراک
Throughput	توان عملیاتی، گذردهی
Time window	پنجره‌ی زمانی
Timestamp	برچسب زمانی
Variety	تنوع

کلمه انگلیسی	معادل به فارسی
Velocity	سرعت
Veracity	صحت

Abstract

The growing tendency to event-based architectures on the one hand and the prevalence of computing systems on the other hand, has caused the increasing growth of event-producing resources and consequently the rise in event production rate. In many domains of application, it is necessary for the computing systems to acquire the required understanding of their environment by processing events, recognizing the predefined patterns and taking them to higher levels of abstraction, and to show a certain reaction after taking the right decision. This type of processing is called complex event processing. The escalation and spread of application of such systems, in addition to causing high event processing rate, has engendered the need for a standardized tool to perform benchmarking on such systems, with the end of comparing and assessing their performance.

In this research, we have developed an independent series of softwares, which enable us to determine several variables such as event loss, and delivery and arrival rate, which in turn allows us to scale the complex event processing system itself. This toolset is presented as a standard candidate mean for scaling and benchmarking CEP systems.

Keywords: Complex event, Complex event processing, Events stream, Benchmarking Complex Event Processing systems



School of Computer Engineering

A Toolset for Benchmarking Complex Event Processing Systems

**A dissertation submitted in partial fulfillment of the
requirements for the Degree of Bachelor of Science in
Computer Engineering (Software Engineering)**

By:

Mohammad Reza Barazesh

Supervisor:

Dr. Mohsen Sharifi

August 2016

