浙 江 大 学

硕士学位论文开题报告

（专业学位）

论文题目： 复杂事件处理在应用性能监控中的应用

姓　名： 常文龙

学　号： 21451136

专　业： 软件工程

院　别： 软件学院

导　师： 孙建伶

二零一六 年 9 月

目　录

1. 课题来源及类型

应用性能监控 APM(application performance monitoring)是近年来企业级应用中比较受到关注的一个领域。互联网时代快速发展，各种应用层出不穷，用户对应用使用体验、应用的可用性、应用响应速度等应用性能指标的要求也随之提高。为了尽量减少客户流失率，增强用户体验，企业在应用性能监控上进行了大规模的投资。IDC 的统计数据显示，40%的应用性能问题可归因于应用本身或应用相关的原因，所以对应用进行优化是提高应用稳定性和应用服务持续运行能力的有效途径。

应用性能监控系统对应用的基础设施和关键业务进行监控，能够帮助开发人员及运维人员快速发现应用性能问题，并进行相应性能优化，以保障应用系统能够稳定、高效率地运行。

APM 主要分为应用性能数据采集、存储、可视化、分析与报警等阶段。应用性能数据分析的传统方法是将产生的性能数据存储在关系型数据库中，间隔一段时间，借用分布式框架如 Hadoop 进行批量处理，得到分析结果。这种方法不能有效地进行实时分析处理，很难满足许多商业应用希望快速获取分析结果，及时产生报警的需求。

复杂事件处理 CEP(Complex Event Processing)是一种事件驱动的实时数据处理技术，用于检测连续达到的数据之间存在特定模式，具有高吞吐量、低延时和复杂计算等特点。将复杂事件处理技术应用在 APM 的分析报警阶段，对提供实时的分析与报警。

例如，程序员如何解决生产环境上的问题，对于不提供 APM 的应用，当生

产环境出现问题时，一般的解决步骤是：1)下载生产环境的日志文件；2）在大量的日志文件中查找 bug 信息；3）分析 bug，解决 bug。

对于提供 APM 的应用，当生产环境出现问题时，开发人员只需登录 APM，根据出现 bug 的大致时间，快速定位 bug 信息，同时还提供方法调用堆栈，以及一些处理参数，帮助开发人员解决问题。

对于提供实时分析报警功能的 APM，只需在系统中定义相应的监控指标，APM 就会在相应 bug 出现的时候，产生报警信息发送给开发人员，开发人员可快速分析处理 bug，以免产生更严重的后果。

将 CEP 应用在 APM 中，借助 CEP 的实时处理、规则分析，实现了 APM 的实时分析报警功能，帮助用户快速定位应用性能问题，未解决问题争取宝贵的时间。

2. 课题的意义及国内外现状分析

2.1 应用性能监控概述及发展历程

绝大部分的应用系统都会提供对自身运行状况进行监控的功能，以方便用户更加深入和全面地了解系统的性能和当前的健康情况。

从性能监控系统测量的对象方面讲，需要监控的对象有：方法的时间响应，程序运行异常信息，业务数据的处理情况。对于系统资源的利用情况，如 CPU 利用率、内存利用率、网络带宽等。几乎所有的性能监控系统提供对上述这些监测数据的采集和可视化展示，但对这些收集过来的监控数据进一步分析，得出更有价值的监控信息的支持还很限，更不用说允许用户自主定义监控指标产生报警

信息这一功能。比如，我们采集到一条 CPU 使用率的监控数据，CPU 某一时刻的使用率达到 95%，这条数据其实不能准确说明 CPU 的负载情况，但若在过去半小时内采集到的 CPU 使用率均是大于 90%，这时我们就需要生成报警信息通知到相关负责人。目前有少部分性能监控系统开始关注对收集的原始监控数据的分析，允许用户自主定义的监控指标，并对触发监控指标的情况生成报警信息。

从 90 年代末 APM 理念出现到今天的产品方案，APM 受到技术、市场、产品的冲击与更新到现在，大致分为三个阶段：

第一阶段是以网络为中心，网络速度=应用速度。1996 年 Tivoli 与 HP 公司开发了从应用程序层面出发的应用响应管理开发包（ARM API 1.0），随后的 2.0 版本被 The Open Group 批准为开放标准。1998 年提出的面向商业的网络管理（BONM）概念，BONM 被定义为以协助网络管理者测量和提高运行网络端到端的性能，它的功能包括监控和故障发现、带宽管理、数据分析和服务水平等级协议（SLA）等，APM 这个理念正式作为这种软件技术的一部分提出。由此 2003 年 IETF 还专门为 APM 定义了管理信息库（SNMP-MIB）。

第二阶段是以 IT 部件 / 组件为中心，部件 / 组件健康监控，基础设施可用性监控。这个阶段伴随各种 IT 基础架构组件的发布，如：网络、系统、中间件、数据库。

第三阶段是以 IT 应用为中心，高度复杂，交易为核心，面向用户，面向应用生命周期管理。

经过上述这三个阶段近 15 年的发展，APM 领域已经形成了较完善的相关管理标准与解决方案。

近年来国内外也涌现出一些成熟的 APM 解决方案，国内有 OneAPM、听云，

国外有 NewRelic 、 APP Dynamic。他们提供丰富的可视化展示，托管之上的应用，可以查看系统资源的利用情况，方法的执行时间调用栈和慢查询 sql 等大多数应用共同关注的性能监测点。对于不同应用的具体业务逻辑，不能提供有效的监控。小公司一般将应用托管在 APM 服务提供商提供性能监控系统上。大型公司大多选择自主研发适用于公司业务的性能监控系统。

2.2 复杂事件处理概述及发展历程

复杂事件处理技术作为一种实时数据处理技术，近些年来逐渐走入人们的视野。它的起源与主动型数据库系统中的触发器、数据流处理系统、发布订阅系统等都有联系；它以从传统数据库技术中独立出来，成为一个新兴的研究领域。在传统的数据库系统中，数据首先被存储起来，在用户查询后返回结果;而复杂事件处理技术则是先将查询存储起来，当符合查询条件的事件发生时，实时地通知该查询的来源，这就实现了数据的实时处理。

当今事件处理技术正快速成为今天信息社会处理信息的基础。这个技术出现在 20 世纪 90 年代末期，但是事件处理并不是新的东西。在 20 世纪 90 年代，有一些研究工程指定去开发事件处理新的原则，被叫做复杂事件处理(以下简称 CEP)。CEP 的概念最早是由斯坦福大学的 David C．Luckham 提出的。主要用来研究如何从海量基本事件中抽取、组合、发现或者总结抽象出更高层、更复杂的事件。它是 1 种新兴的基于事件流的技术，将系统数据看作不同类型的事件，通过分析事件问的关系，建立不同的事件关系序列库，利用过滤、关联、聚合等技术，最终由简单事件产生高级事件或商业流程。可以为物联网、互联网计算、云计算中的上层应用提供具有复杂语义的发现、监测、预警等基础服务。

近年来 CEP 在实时数据流处理、商业活动监控、商业过程管理的自动化服务体系中扮演着越来越重要的角色。学术界和企业界对复杂事件处理领域的关注日益剧增。在 2000 年时已经有人关注基于事件的系统和对数据流的处理。但是从 2004 年开始关于复杂事件处理的研究报道才逐渐增多，也出现了一批相关开源项目和工具。现在国际上已经有了很多较成熟的复杂事件处理产品和项目。例如商业领域的产品有 BEA Systems、TIBCO Business Events、Alert CEP3 .0、IBM StreamBase 和 D EsperTech 等。

今天 CEP 主要集中应用在业务信息系统上，用来管理业务的更高等级类型的事件，但是我们能预测到 CEP 将应用于任何种类系统产生的事件类型。CEP 体现了构建应用程序的原则，嵌入 CEP 技术的 IT 系统使企业能够跟上信息的流动。CEP 的目标是让企业 IT 基础设施所有层的事件流的信息被发现，理解高层次的管理目标和业务流程的影响，并采取实时的行动。比如处理 RFID 技术所创建的事件。CEP 采用的检测技术，如很多事件的复杂模型检测，事件流处理，事件关联和抽象，事件层次结构，如因果关系，成员，和定时的事件之间的关系。CEP 还可以补充和促进某些技术 ,如面向服务的架构(SOA) ,事件驱动架构(EDA)和业务流程管理(BPM)。

事件处理软件的发展，以及对 CEP 设计问题的更深刻的理解和对更快更智能的业务处理的需求，使得 CEP 变得越来越流行。现在已经广泛应用在金融、系统自动化交易、BAM、RFID、高级检测系统、欺诈检测等领域，虽然应用的范围很广，但是他们都是独立存在的，相互之间并没有关联，而往往在一个环境下的不同系统或者同一系统下的不同环境之间存在着某种联系 ,比如业务活动和系统安全方面的关联等，在软件行业关于事件处理的发展空间很大，需要不断地

深入研究。而且随着市场空间的扩大，多种实现之间出现的竞争，需要标准化来更好的为行业提供服务。

2.3 基于复杂事件处理的应用性能监控的提出

要保证应用稳定可靠地运行，就要充分利用应用性能数据采集模块采集的数据，挖掘数据中存在的有用信息，传统的 APM 使用大数据处理技术（如 hadoop）对性能数据做离线分析得出分析结果，不支持实时处理。

对性能数据的实时处理，对系统已出现异常的检测及未来可能出现异常情况的可靠预测，都是新一代应用性能监控系统所要支持的。

复杂事件处理作为实时数据处理的一个分支，对输入的原子事件流进行分析，生成复杂事件。多用于金融领域，如反欺诈，股票走势预警。APM 的特点：1)采集的性能数据时时序的（带时间戳）；2)重点关注的是最近一段时间的数据；3)主要用来检测系统运行状况，对系统异常产生实时报警。以上几点非常适用于 CEP 技术:1)CEP 自带复杂事件规则分析引擎，能够保证分析结果的可靠性;2)CEP 将源事件置于内存，这种基于内存分析能够满足实时性需求；3)CEP 提供滑动时间窗口的设计，满足 APM 中重点关注最近一段时间数据的需求，能够自动进行能存管理，充分利用内存资源。

复杂事件处理是基于内存处理的技术，瓶颈主要在内存大小和 CPU 的处理能力。虽然复杂事件能够自动进行内存管理，自动回收位于滑动窗口之外的事件数据，但高速的事件流加上大的时间窗口，导致时间在内存中堆积，很快就会出现内存不够用的情况。

复杂事件处理基于规则引擎，将定义好的规则载入规则引擎，对不断流入事

件进行规则匹配，匹配成功则生成复杂事件。在规则定义这一块，学习成本高，且正确性难以保证，一般由专业人员根据业务人员描述的需求进行定义。

本文研究将复杂事件处理技术应用到应用性能监控系统中,借助复杂事件处理技术,为监控系统提供数据实时处理和报警能力,本文针对复杂事件处理引擎单机处理能力,内存不足的情况提出分布式的解决方案,采用集群部署,提高了处理能力，增强了扩展性。

3. 课题的研究目标、研究内容和拟解决的关键问题

3.1 课题研究目标

研究目标是：结合应用性能监控基本需求以及**公司内部应用业务的特点，探讨如何实现一个应用性能监控系统,提供应用性能数据采集、数据存储、性能数据分析报警以及数据可视化一整套服务。重点研究性能数据分析报警阶段,引入实时数据处理技术-复杂事件处理，利用基于规则的复杂事件处理引擎 JBoss Drools Fusion，为客户提供实时的性能监测和报警服务。

3.2 课题研究内容

本文针对应用性能监控和复杂事件处理的特点,主要对海量事件的复杂事件处理和监控指标自定义进行研究。提供对海量事件处理的支持,允许用户自定义监控指标插入到复杂事件处理引擎进行分析报警。

本文主要研究内容包括：

为了提供复杂事件处理对海量数据的支持,我们实现了一套基于规则分发的分布式复杂事件处理系统。Jboss Drools Fusion 是一种基于内存的复杂事件处理

引擎，输入的事件存放在内存中，基于内存的分析保证了处理的实时性，但处理能力也受到内存大小的制约。随着内存中事件的堆积，会导致处理速度变慢甚至直接崩溃。要解决这一问题，从复杂事件处理技术本身来考虑，可以定义合理的滑动时间窗口大小(复杂事件提供自动内存管理，回收处于滑动事件窗口外的事件)，多个复杂事件处理规则共享一个工作内存（work memory）。这种方法不能从根本上解决内存不足，cpu 不够用的情况。在经过对老系统中已存在规则的分析，发现大部分规则之间是相互独立的，于是我们提出了基于规则分发的分布式复杂时间处理系统。规则分发到分布式系统的各节点，事件流则采用广播技术发送到分布式系统中的所有节点，提高复杂事件并行处理能提，增强系统扩展性。

针对用户可自行定义监控指标的需求，我们实现了基于自然语言、表意明确的监控指标定义模块。想要提高系统扩展性，我们需要支持用户根据相关需求自行定义监控指标，Jboss Drools Fusion 处理的规则定义在后缀为 drl 的文件中，drl 文件支持 drools 规则语言，允许嵌套 java 代码，最终会被规则引擎解析成一个 class 文件，之前用户通过上传 drl 文件自主添加监控指标，这对用户来说，需要专门去学习 Drools 规则语言，学习成本高，正确性难以保证。同时若后续换了其他复杂事件处理框架，则之前定义的监控指标都变得无用。

通过分析监控指标，发现大多监控指标的结构大体一致，大都为监控过去一段时间内性能数据的变化。基于此，为用户提供定义监控指标的图形化界面，指引用户一步步完成监控指标定义，用户仅需填及个空，就可完成较复杂的监控指标定义，后台实现的解析工具，会将用户定义的内容解析成 drl 文件输入到复杂事件处理。

统一的事件表示也是我们要考虑的问题，复杂事件处理接收事件流的输入。

应用性能监控采集到的有多种不同类型的数据,比如 cpu 使用率、消息队列深度、方法执行时间等等,将这些数据包装成事件输入到复杂事件处理引擎之前,要考虑统一的时间表示,我们将采集数据定义成统一的数据结构 TimeData,包括四个属性(metrics、tags、value、timestamp),如一条 cpu 使用率的数据可以表示为 TimeData(CpuUsed、gce3455.statestreet.com、95、15234456678),CpuUsed 表示监控数据的分类、gce3455.statestreet.com 表示是那台机器的 cpu 使用率,95 表示 cpu 使用率为 95%,15234456678 是一个长整型表示采集数据的时间点。用了统一的时间表达,后续监控指标的定义也会变得简单、适用性更强。

3.3 拟解决的关键问题

复杂事件处理接收事件输入,我们需要为监控数据定义统一的事件表示;解决自主定义监控指标复杂度太高的问题,提供图形化界面指导用户定义监控指标;解决单机复杂事件处理能力不足,内存不足的问题,提供基于规则分发和基于事件流分发的分布式解决方案。

4. 课题的研究方法、设计及试验方案,可行性分析

4.1 课题设计方案

4.1.1 应用性能监控系统组成

本设计方案中,应用性能监控系统主要分为以下几个部分,每部分的基本功能和组成如下。

(一) 应用性能数据采集部分，用户根据需求配置定时任务采集监控数据，如方法的执行时间，方法的调用堆栈，异常信息，慢查询 sql 语句，系统资源的利用情况(CPU 利用率、内存利用率)、mq 深度等。数据采集部分采集的数据格式统一，会被封装成格式统一的事件，输入到分析报警系统的复杂事件处理引擎中。

(二) 监控指标定义模块，提供图形化界面指导用户定义监控指标，后台根据用户输入生成具体复杂事件处理引擎对应的规则。

(三) 应用性能分析报警模块，使用复杂事件处理技术 Jboss Drools Fusion 充当实时分析工具，连接数据采集模块接收连续事件，连接规则解析部分，生成 Drools 规则写入到复杂事件处理引擎。复杂事件处理引擎进行规则匹配，生成复杂事件，产生报警信息，通知到相关责任人。

(四) 应用性能数据可视化展示模块，提供可视化 web 页面，丰富的图表展示，展示被监控应用的运行状况，让用户很好的了解应用的健康状况。

4.1.2 规则解析部分

收集老系统中监控指标，分析监控指标的特点，统一监控指标定义格式，定义一套自然语言的规则解析工具，使得定义的规则的过程变得简单明了，定义的规则表意明确。对原有监控指标采用自定义规则实现，比较规则定义的复杂度，测试规则正确性以及功能完整性。

4.1.2 分析报警模块

分析报警模块使用开源的复杂事件处理引擎实现，分析处理能力依托于选用

的复杂事件处理引擎复杂事件处理的能力，对比几种开源实现的 CEP（Jboss Drools Fusion 和 Esper），处理能力都在同一个量级，单机处理能力难以提升，为了增强复杂事件处理能力、实现分析报警模块的可扩展，我们提出了横向扩展的方法，使用分布式部署，多个实例分摊监控指标，提高整个模块的并行处理能力。实现基于规则分发，基于事件流分发的分布式分析报警模块。

4.2 课题可行性分析

时间可行性分析：基于 Athena 项目团队的前期积累，目前已有一个正在运行的版本，整体架构设计已完成，具体需求明确，后台拥有大量的监控数据，已定义的监控指标，有足够的时间使用现有的复杂事件处理技术实现对监控数据的分析和报警，时间上是可行的。

技术可行性分析：CEP 技术发展至今，有很多开源实现（如 Jboss Drools Fusion、Esper），有详细的技术文档，活跃的技术论坛，使用现有开源框架，源代码开放，在遇到问题时，可以深入到源代码进行调试分析。课题依托真实的企业系统，真实的业务场景，和监控数据，在 APM 领域也有一些成熟的解决方案可以借鉴（如：听云、OneAPM）。

基于以上两点,我们认为本文的研究方法与设计方案切实可行,可操作性高,研究结果也具有很高的实用性,课题具有一定的研究价值。

5. 课题计划进度和预期成果

5.1 计划进度

开始时间	结束时间	主要工作内容
2016 年 09 月	2016 年 10 月	查阅文献资料，编写课题开题报告
2016 年 10 月	2016 年 11 月	进行需求分析 ,明确系统需求 ,搭建系统架构，进行系统设计
2016 年 11 月	2016 年 12 月	调整系统设计，基本实现系统需求
2016 年 12 月	2016 年 01 月	撰写论文正文

5.2 预期成果

总结全文，本文预期取得的主要研究成果如下：实现基于复杂事件处理的应用性能监控系统，为用户提供实时分析和报警功能。