一种新的基于事件面向服务的 BAM 系统设计研究 *

陈向东
(马鞍山师范高等专科学校 软件学院, 安徽 马鞍山 243041)

摘 要: 研究了面向服务的业务活动监控(BAM)系统设计问题。在分析 BAM 的几类系统模型优缺点的基础上,立足于 BAM 系统的动态感知、协同工作和实时处理数据,采用事件驱动体系架构,提出了一种基于事件、面向服务的业务活动监控系统模型。该模型以总线的形式构建在企业原有的各个信息系统之上,采用事件作为模块之间基本的通信机制,并对该系统模型中关键技术的设计进行了详细分析。最后通过原型系统的实验表明,该设计方案有良好的应用效果,可以有效地提高模型的计算效率。

关键词: 面向服务; 业务活动监控; 关键绩效指标; 发布/订阅; 事件
中图分类号: TP311 **文献标志码**: A **文章编号**: 1001-3695(2012)03-0977-04
doi:10.3969/j.issn.1001-3695.2012.03.048

New system based on event-driven and service-oriented business activity monitoring design and implementation

CHEN Xiang-dong
(*College of Software, Ma'anshan Teacher's College, Ma'anshan Anhui* 243041, *China*)

Abstract: Based on analyzing the advantages and disadvantages of several system models in BAM, established in the dynamic sense, cooperative work and real-time data processing of BAM system, this paper employed an event-driven architecture and proposed an event-based and service-oriented business activity monitoring system model. This model was constructed in the form of buses on the pre-existing various information systems of enterprises, employed events as the basic communication mechanism between modules. It also analyzed the key techniques in the system model in detail. Finally the result shows that the design has a good effect through the experiment of the prototype system.

Key words: service-oriented; business activity monitoring(BAM); KPI; pub/sub; event

当今的企业正面临着许多的压力,要适应瞬息变幻的市场环境,构建一个快速反应环境是企业在残酷的市场竞争中生存发展的关键。虽然计算机软件产品在企业各个领域中已得到广泛使用,但在信息的管理分析工作方面,仍然停留在用户以主动方式触发数据管理和分析功能,而这些信息往往滞后于现实,无法及时发现问题,使企业错失机会、蒙受不必要的损失。在如今这样一个复杂多变的市场环境下,实时性、开放性以及集成性强、可预测分析正是企业在竞争中取胜所急需的,业务活动监控系统(BAM)正是源于这些需求而产生的。BAM 的目标[1]是提供多种业务操作、业务过程和业务事务的状态和结果的实时信息,对业务活动进行管理,使企业具备了敏捷型企业所要求的素质,能够快速地响应市场变化,快速地调整业务策略,快速地实施业务流程,同时根据反馈的信息对业务流程进行快速的优化调整。

当今企业应用的规模和复杂性,使得体系结构成为 BAM 软件开发及应用能否成功的决定因素,BAM 系统与企业应用的集成方法、BAM 系统中数据的实时处理方法正成为 BAM 领域中备受关注的问题。BAM 系统模型取决于建模目标,BAM 系统作为一个服务中间件支持二次开发,如何能够与企业现有应用服务进行有效集成、降低应用之间的耦合度、降低开发难度、提高系统的开放性和扩展性以及数据处理的实时性是 BAM 系统模型需解决的问题。

1 相关工作

BAM 这个术语是在 2002 年由高德纳咨询公司(Gartner Group)提出的[1],是基于企业应用集成的一种用于监控企业运营状况的软件技术。从 BAM 术语提出至今,BAM 系统模型已经涌现出许多的研究成果,主要分为四类,即以数据为中心、以执行为中心、面向对象和面向服务的 BAM 系统模型。

1.1 以数据为中心的 BAM 系统模型

以数据为中心的系统模型是将数据库放在系统的核心层次共享[2,3],各功能部件采用统一的数据描述,各子系统完全独立;系统的整体可扩展性好,可以任意添加符合数据交换标准的应用程序。但是系统缺少实时性,因为该模型首先把数据写入数据库,BAM 再从数据库中读取数据进行处理,最终向用户展示的信息存在大量的延迟。

1.2 以执行为中心的 BAM 系统模型

以执行为中心着眼于将企业不同的应用系统通过 BAM 执行中心集成在一起[4,5]。该模型将共有的数据处理(数据获

收稿日期: 2011-08-23; **修回日期**: 2011-09-30 **基金项目**: 安徽省 2010 年高等学校省级教学质量与教学改革工程项目——省级教学研究项目"高职高专软件技术校企合作、工学结合培养服务外包人才的创新研究"(20101236)

作者简介: 陈向东(1972-),男,安徽怀宁人,副教授,硕士,主要研究方向为软件工程及应用研究(masszcxd@qq.com).

取、数据转换、数据传输)从应用程序中分离出来,放在执行中心,避免了代码的冗余;与数据库的交互都需要通过执行中心进行,有利于数据的严格管理,保证了数据的一致性。但是此种模型执行中心的功能设计复杂,同时与用户界面和应用程序保持通信,又管理 BAM 数据库,负担过重,极易形成瓶颈。

1.3　面向对象的 BAM 系统模型

随着面向对象技术的成熟,出现了更为简练的面向对象的系统模型[6,7]。该模型与前两类模型的设计思想有较大差别,BAM 内核对象中封装的是能为用户界面对象、所有应用对象、服务组件对象共享的数据及相应的操作;所有这些对象通过相互间的通信协调来完成指定的功能。面向对象的系统模型的主要优点在于:数据和功能的合理封装降低了由于数据和功能的集中管理所带来的通信上的开销和操作上的复杂性;另外,系统的无中心结构也使系统的构成变得更加灵活。从整体上看,面向对象的系统模型无论其开放性还是其有效性都要优于前两类模型。

1.4　面向服务的 BAM 系统模型

面向对象模型比以往的模型有了很大的进步,但仍有不足。对象之间的联系是一种点对点的直接联系,当系统中对象数目增加时,通信链接数将以平方级激增,同时为支持通信,每个对象实体都要维护一个包含所有对象实体功能服务信息的功能服务信息库,这部分信息不但重复,而且还要保证其一致性。这些开销都损害了系统的效率。更大的问题还在于:对象的接口没有一致的标准,造成向系统中扩充对象时的随意与不规范,不利于系统的维护以及对象的复用。随着企业中的软件系统变得日益庞大和复杂,系统之间的很多功能属于重复开发,既浪费资源又不便于管理。SOA 提供了一个具有服务可重用性的理想集成框架,支持在业务流程中进行服务组件的装配。基于 SOA 的 BAM 系统模型[8]通过企业服务总线将企业应用集成在一起,这种方法的优点是松散耦合、位置透明,具有可扩展性。然而现有的 SOA 不完全具备动态感知、协同工作、异步处理的能力,而事件驱动体系架构弥补了 SOA 的缺陷,通过感知器监测企业信息系统中数据状态的变化,然后产生可以被 BAM 处理的事件,通过事件实现系统模块之间的协同。基于此思想,本文提出了一种基于事件、面向服务的系统模型(EDABAM),动态协同是此模型的核心技术,企业信息服务系统将活动状态的变化映射成事件,通过对事件的感知、收集、发布、处理,实现业务活动状态的实时监控。

2　基于事件面向服务的 BAM 系统模型

事件驱动体系架构是一种软件架构模式,这种构件架构模式通过松散耦合的软件构件和服务来传送事件。本文所要描述的 EDABAM 系统正是基于这一思想设计的,系统体系结构如图 1 所示。整个体系架构以总线的形式构建在企业原有的各个信息系统之上,采用事件作为模块之间基本的通信机制,为企业中需要合成数据的各种应用系统提供了交互的信息流通渠道。整个系统通过适配器把信息从企业应用服务中分离出来,不需要定制和改变原先的各个信息系统就可以达到信息共享,适配器将企业信息系统中业务活动状态的变化以事件的形式发布出去,激发整个 EDABAM 系统的运行;整个系统主要由适配器(adapter)、规则引擎模块、EDABAMCore 模块、数据库模块、UI 模块、仪表盘模块组成。模块之间通过事件进行协调工作,对企业的业务活动数据进行实时监控。

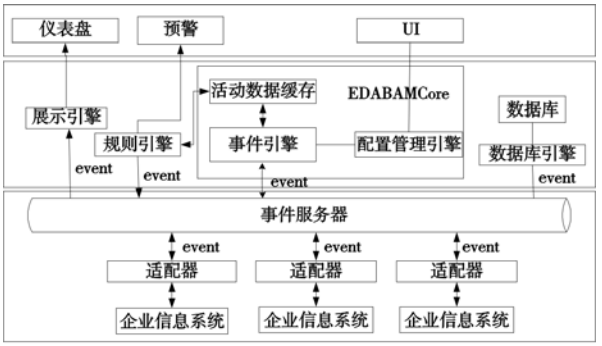

图1　EDABAM体系架构图

为了便于后文讨论,下面给出事件模型等相关概念定义:

定义 1　企业信息系统中的原始数据(RawData)。RawData 表示为一个四元组$\langle N,T,V,D\rangle$,其中,N 为数据的名称,T 为该数据的类型,V 为数据值,D 为企业信息系统名称。

定义 2　关键业务绩效指标(key performance indicator, KPI)。KPI 是对于企业中业务目标的量化规约,在 EDABAM 中每一个 KPI 对应于一个业务活动类型,通过对 KPI 的实时监测来监控企业业务活动的运行状况。KPI 表示为一个四元组$\langle K,T,V,B\rangle$,其中,K 为 KPI 的标志符,T 为 KPI 的类型,V 表示 KPI 的值,B 表示 KPI 的边界值(通常包含上边界和下边界两种)。

定义 3　业务度量(metric)。业务度量是对于业务流程实例运行过程中某一个侧面的记录,同时也是对于 KPI 的细化,它可能并不具有明确的业务含义,但是多个业务度量指标经过聚合产生一个关键绩效指标(KPI)。Metric 是从企业信息系统的 RawData 列表中选取配置获得,RawData、metric 及 KPI 关系图如图 2 所示。Metric 表示为一个四元组$\langle M,V,T,D\rangle$,其中,M 表示 metric 的标志符,V 表示 metric 的值,T 表示 metric 的类型,D 表示为 metric 所属的企业信息系统的名称。

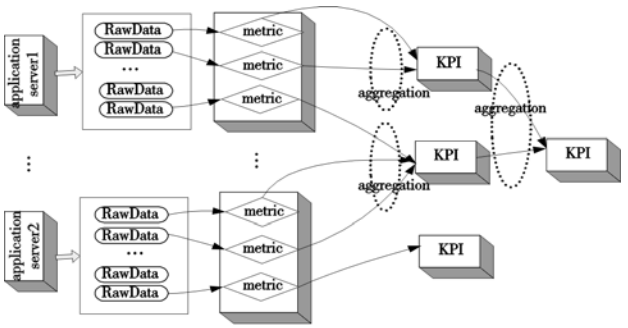

图2　RawData、metric及KPI关系图

定义 4　规则(rule)。规则是对业务的某个方面进行定义或约束的语句,在 BAM 中规则细分为 KPI 生成规则和 KPI 业务规则。KPI 生成规则描述了 KPI 由哪些 metric 通过什么样的运算计算所得;KPI 业务规则描述了 KPI 达到什么样的临界值产生预警。规则表示为一个三元组$\langle I,P,O\rangle$,其中,I 表示规则的输入,P 表示规则的断言,O 表示规则的输出。针对 KPI 生成规则,I 表示生成 KPI 的 metric 列表,p 表示 metric 是否发生更新,O 表示更新 KPI 属性值。针对 KPI 业务规则,I 表示

KPI 对象,p 表示 KPI 是否到达一定的阈值,O 表示预警。

定义 5　事件(event)。事件语义上是指发生的活动或者状态变化,事件在 EDABAM 中是信息的载体。一个事件表示为三元组 $E\langle T,S,M\rangle$,其中,T 表示事件类型,S 表示事件源,M 表示事件中包含的消息。

2.1　EDABAM 系统内部结构

适配器是整个系统的基础。适配器的作用在于:一方面,代理企业信息系统的复杂通信过程,使企业信息系统更专注于功能的实现;另一方面,它将企业信息系统的数据通过事件发布出去,激发整个 EDABAM 系统的运行。EDABAMCore 模块是这个系统的核心,管理系统的运行;通过配置管理引擎配置被监控服务(企业信息系统)、监控模型(metric 模型、KPI 模型以及规则模型);通过事件引擎和企业信息系统进行交互,它将企业数据信息从事件中解析出来,存储在活动数据缓存中;通过活动数据缓存中的数据更新驱动规则引擎的运行。规则引擎模块负责执行 KPI 生成规则和 KPI 的业务规则,KPI 生成规则执行的结果,一方面更新活动数据缓存中 KPI 的值,另一方面,将更新的 KPI 发布到消息服务器上。KPI 业务规则执行的结果通过短消息和邮件进行预警。展示引擎和数据库引擎通过事件服务器订阅 KPI 信息,展示引擎负责将 KPI 的实时信息通过仪表盘展示给用户,数据库模块将 KPI 信息进行持久化。

2.2　EDABAM 关键技术的设计与实现

EDABAM 中采用 JMS 消息中间件充当事件服务器。JMS 规范提供两种最普遍的事件处理模式:发布/订阅和点对点。在这个体系架构中,适配器将企业中业务活动状态信息通过事件发布到 JMS 服务器上,驱动整个系统的运行。系统的各个模块之间均采用事件进行相互之间的通信。

1)适配器的设计与实现

a)点对点的异步请求/应答处理机制。在 EDA-BAM 配置阶段首先要获取企业信息系统中的 RawData 数据列表,然后在 EDA-BAM 中发送 RawData 数据请求,最后接收到 RawData 数据为一个事务。因此 EDABAM 和适配器之间采用了点对点的异步请求/应答处理机制来获取 RawData 数据列表,EDABAM 向一队列中发布请求 RawData 数据列表事件,然后阻塞等待应答队列,该应答队列等待来自适配器的响应。这种交互如图 3 所示。

图3　适配器点对点的异步请求/应答处理机制

b)基于主题的发布/订阅处理机制。适配器和 EDABAM 之间采用基于主题的发布/订阅模式来发布和获取 metric 对象;当企业信息系统中 metric 产生了更新后,适配器将此更新封装到事件,该事件中包含了企业信息系统的信息、metric 的信息以及事件类型等,之后通过 JNDI 定位 JMS 服务器,并将该事件发送到特定的主题上。JMS 服务器接收到该事件后立即将该事件推送给 EDABAM 的事件引擎,采用算法 1 对事件进行处理。

算法 1　事件过滤处理算法

```
输入:企业数据更新事件 ExternalEvent e = (T,S,M)。
输出:更新企业中的 metric 信息。
for each e do{
if( e is a DataUpateEventTpye and e.m ≠ null ){
  MonitoredServiceURL serviceURL = getFromEvent(E);/*从事件中获取被监控服务的地址*/
  MonitoredService service = getFromMonitoredServiceList ( serviceURL);//从系统中获取该服务对象
  if(service ≠ null ) {
  MetricName name = getMetricNameFromEvent(e.M);/*从事件数据中获取 metric 的名字*/
  MetricValue value = getMetricValueFromEvent(e.M);/*从事件数据中获取 metric 的值*/
  if(name in service and value ≠ null){
    updateServiceMetric(name,value);//更新 metric 的值
    fireMetricUpdateEvent();//激发 metric 更新事件
  }else {return ;}
  }else {return ;}
else {return ;}
endfor
```

2)基于事件代理、多线程的活动数据处理机制

a)事件代理处理机制。该框架中的 metric 设置为 JavaBean 对象,在 JavaBean 中通过属性更新事件与其他组件进行交互;metric 作为事件源,它维护一个事件代理管理事件;规则引擎组件实现事件监听器接口来监听和处理此类事件。

在本设计中为了方便对规则的维护和管理,针对每一个 KPI 的生成规则创建一个规则引擎实例对象。在该规则引擎的工作内存中添加相应的 metric 和 KPI 对象,按照传统的 JavaBean 事件处理方法需要调用 addPropertyChangeListener()方法来注册该监听器。在实际的应用中可能包含多个 metric 对象,因此利用这种方法比较繁琐,也不安全,任何一个对象中没有注册该监听器将会导致程序出错。本文通过 Java 反射机制注册事件监听器来避免这种错误的出现,即在规则引擎插入事实对象这一方法中,将规则引擎作为事件监听器添加到对象的事件代理中,示例代码如下所示:

```
addPropertyChangeListener (Object object)//object 为 metric 对象
{
  Method method = object.getClass( ).getMethod( "addPropertyChangeListener",PropertyChangeListener.class );
  method.invoke( object, this );
}
```

b)多线程事件处理。JDK 中 JavaBean 事件代理为了防止数据冲突,对注册的事件处理器进行了串行化处理。在本文设计中,metric 值的更新将激活与之相关的 KPI 生成规则的执行,在规则的执行过程中针对 metric 只有读操作。如果采用串行化执行,将会导致与该 metric 相关的 KPI 生成规则依次执行,严重降低了系统的处理速度。本文采用算法 2 的处理方法重新设计了 JavaBean 的 PropertyChangeSupport 对象,使得 metric 产生更新后,与之相关的 KPI 生成规则并行执行,提高了系统性能。

算法 2　基于多线程的 KPI 并行处理算法

```
输入:发生更新的 Metric metric。
输出:更新 KPI。
MetricListener listenerList = metric.getListenerList();/*获取 metric 对象所有监听器*/
for each listener in litenerList do
start a thread{
  kpi = getFromListener();//获取监听该 metric 的 KPI 对象
  KPIRuleCondition con = getKPICondition(kpi);/*获取 KPI 生成规则的条件*/
  for each con in cons do
  if(metric matched con) //条件匹配
  { ruleEngine = getRuleEngine(kpiRule);/*利用 KPI 生成规则初始化一个规则引擎实例*/
```

```
    ruleEngine.insert(kpi);//将 KPI 插入到规则引擎中
    ruleEngine.fireRule();//激活规则更新 KPI
    endfor
endThread
endfor
```

3)基于发布/订阅的 KPI 发布机制

BAM 的核心任务是通过 KPI 值的实时监测来监控企业的业务活动运行状况。EDABAM 系统作为一个服务中间件通过将实时产生的 KPI 发布到事件服务器上支持二次开发,软件开发人员可以从事件服务器上订阅该事件,从中获取 KPI 的实时值,通过自己开发的应用程序实时展示。例如可以利用仪表盘、曲线图、柱状图、日志等方式进行展示。

在本方案的设计中,活动数据缓存中 metric 发生更新将驱动规则引擎的执行,产生新的 KPI 属性值,并将更新的 KPI 通过事件发布到事件服务器上,展示引擎和数据库引擎订阅 KPI 的更新事件,最终展示引擎将 KPI 的值通过仪表盘展示给用户,同时数据库对 KPI 进行持久化。

3　实验及结果

基于以上设计思想,采用开源消息中间件 ActiveMQ 作为事件服务器实现了一个原型系统,在该平台上进行如下实验。

实验 1　本文分别在 10 台 PC 上部署了经过编译和调试通过的应用程序,该应用程序中包含 10 个变量,这些变量周期性地利用随机数进行赋值,该应用程序模拟了企业信息系统;然后在每台 PC 上针对该应用程序开发和部署了一个适配器,部署完成以后,启动程序运行。实验表明,所有的应用程序都不做修改地在分布式环境下正常运行,通过适配器能够正常和 EDABAM 进行通信。这说明:a)本文设计的适配器能够为应用程序和 EDABAM 提供正常通信;b)分布式环境下的应用程序通过该框架屏蔽了物理位置的差异,实现了企业信息系统和 EDABAM 的松散耦合。

实验 2　为了考察系统内存开销,通过实验 1 的方法,在每一个应用程序中配置了 4 个 metric,然后每个 metric 每秒进行更新一次,针对 40 个 metric 配置了 20 个 KPI,系统稳定运行了 5 天,EDABAM 系统内存的运行图如图 4 所示。长时间运行后,EDABAM 内存使用范围在 18 M ~ 25 M,平均使用为 22 M,CPU 平均使用率为 2.7%。这对于企业服务器来讲,EDA - BAM 的内存消耗和 CPU 使用率都比较低。

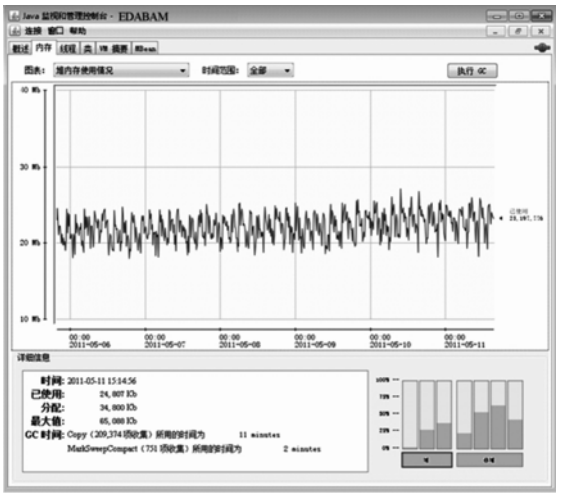

图4　EDABAM系统内存运行图

实验 3　为了考察多线程的处理机制以及测试 EDABAM 的数据处理时间,本文利用实验 2 的方法,分别配置了 10 ~ 100 个 KPI,这些 KPI 都与同一个 metric 相关。通过 metric 一次数据更新,利用 JDK 中串行化的事件处理和本方案并行化的事件处理测试所有 KPI 执行结束的时间,实验结果如图 5 所示。实验结果表明,采用多线程并行化的事件处理大大提高了系统的性能;从 metric 属性变化到 KPI 值的更新处理时间是秒级的,能够满足企业对业务活动状况的实时监控。

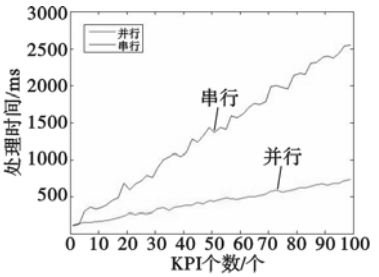

图5　并行处理和串行处理实验结果对比图

4　结束语

当今企业应用的规模和复杂性,使得体系结构成为 BAM 软件开发及应用能否成功的决定因素,BAM 系统与企业应用的集成方法、BAM 系统中数据的实时处理方法正成为 BAM 领域中备受关注的问题。本文通过对相关工作的分析,针对目前存在的问题以及企业的需求,提出了一种基于事件、面向服务的业务活动监控模型,该模型采用事件作为模块之间的通信机制,具备如下特点:a)松散耦合,通过事件服务器将应用程序进行集成,交互两边某一方的改动不会影响到另一方;b)位置透明,通过发布/订阅机制提供位置透明性,即 EDABAM 和企业应用系统无须知道对方实际的物理位置;c)数据的并行处理,通过在 metric 事件代理中采用多线程的事件处理机制,提高了系统的处理速度;d)实时数据处理,通过事件的发布/订阅机制,将数据推送给相关模块进行处理,实现了整个系统的动态感知、协同工作和数据的实时处理。

参考文献:

[1] KANG J G, HAN K H. A business activity monitoring system supporting real-time business performance management[C]//Proc of the 3rd International Conference on Convergence and Hybrid Information Technology. 2008:473-478.

[2] SCHIEFER J, LIST B, BRUCKNER R M. Process data store: a real-time data store for monitoring business processes[C]//Proc of International Workshop on Database nd Expert Systems Application. Berlin: Springer-Verlag, 2003:760-770.

[3] JENG J, FLAXER D, KAPOOR S. RuleBAM: a rule-based framework for business activity management[C]//Proc of IEEE Conference on Services Computing. [S. l.]: IEEE Computer Society, 2004:262-270.

[4] KIM H, LEE Y H, YIM H, *et al*, Design and implementation of a personalized business activity monitoring system[C]//Lecture Notes in Computer Science, vol 4553. Berlin: Springer, 2007:581-590.

[5] BROWN A, JOHNSTON S, KELLY K. Using service-oriented architecture and component-based development to build Web service applications[R]. [S. l.]: Rational Software Corporation, 2002.

[6] KOLAR J. Business activity monitoring[D]. Praha: Faculty of Informatics, Masaryk University, 2010:1109-1115.

[7] 何浪,史维峰,董建刚. 基于事件驱动的面向服务计算模型[J]. 计算机工程,2010,36(18):56-60.

[8] 汪洋,谢江,王振宇. 基于事件的发布/订阅系统模型[J]. 计算机科学,2006,33(1):111-116.