基于 AOP 的性能监测系统的研究与实现

唐雅璇，余金山
（华侨大学 计算机科学与技术学院，福建 泉州 362011）

摘 要：提出一种基于动态 AOP 的监测器模型及其实现，它能够对现有的、已在运行的系统进行最灵活和能耗最小的代码级性能分析。利用 AOP 动态织入机制，可在目标系统运行过程中动态添加或删除监测器，从而提高监测的灵活性。

关键词：性能监测；AOP；IoC；织入机制
中图分类号：TP393；TP311 **文献标识码**：A **文章编号**：1674-7720(2012)12-0074-04

Research and implementation of performance monitoring system based on AOP

Tang Yaxuan, Yu Jinshan
(College of Computer Science and Technology, Huaqiao University, Quanzhou 362011, China)

Abstract: The paper introduced a monitor model based on dynamic Aspect-Oriented Programming .It proves a most flexible and minimizes energy consumption code level performance analysis for current running system.With dynamic weaving mechanism,the monitor could be added or deleted in the running process of the object system to improve the flexible character of monitoring.

Key words: performance monitoring; AOP; IoC; weaving mechanism

随着 Internet 的日益普及和电子商务的飞速发展，基于 B/S 的 Web 应用系统大量出现。在这些 Web 应用系统中，客户端只提供人机交互的接口而未参与实际运算，而服务器端将承担数据存储及运算的全部工作，在面对海量客户端数量的情况下，服务端的性能压力问题就尤为突出。为了保证 Web 应用系统的正常运行，以及系统在服务器端具有较高的性能来响应大量用户的并发访问，进行合理的性能规划和有效的性能监测就显得非常重要。性能监测的主要意义[1]是：在系统运行阶段，通过监测系统的运行情况，发现错误并对故障进行定位，从而解决开发和测试阶段中的错误。

现有的监测器存在以下不足[2-3]：监测逻辑与业务逻辑紧密耦合，可扩展性、可复用性、可维护性差；监测部件分散在目标系统中，缺乏合理的组织；监测代码写在业务代码中，不易修改；监测部件在开发阶段手动插入，不适合动态添加或删除。

本文设计并实现了一款 AOP 监测器，它围绕目前.NET 平台下主流的 Web 开发框架——Asp.net MVC，除了利用 AOP 关注点分离原则，克服了现有监测器的不足外，还针对 MVC 框架的特点，实现了注入切面，不仅实现了方法级别的监测，同时还实现了针对 Controller 及 Action 级别的监测，与其他方法[2]相比，该监测方式更适合 Web 系统的使用。

1 性能监测的作用及主要监测对象

性能监测也可称为构件交互行为监测[4]，指对构件在交互活动期间所进行的操作进行监视记录的过程。构件交互行为[5]是指在分布式构件软件中，构件以主体的身份在完成某项协作活动时，通过构件的接口在运行环境中与其他构件进行交互的活动。性能监测主要包括：软件实体间的交互行为监测、行为信息的收集、为行为诊断、预测和可信评估等提供基础数据。

对于分布式构件软件的性能监测，其各监测类型所对应的监测对象[3]如表 1 所示。

现有的监测器大都面向操作系统级别，其监测范围及粒度均大于 Web 系统的监测所需，从而导致所监测的数据不能很好地用于对系统性能的提升分析上。由于 Web 系统属于非连续性开放系统，基于 HTTP 协议的基本“请求—响应”特性，决定了其安全性及可用性均与系统的业务存在必然的联系，传统的监测器不能根据一定的业务逻辑对所采集的数据进行筛选，而只能通过海量的原始数据结合业务进行判断。这也是本课题未直接采用传统性能监测器而是进行构件设计的一个主要原因。

表 1 构件交互行为监测对象表

监测类型	监测对象
交互行为的基本信息监测	交互事件本身所包含的数据(事件类型、事件中传递的参数名、参数类型、参数值、返回类型、返回结果等),与执行交互事件相关的线程信息(线程名及 ID、线程的状态信息、创建时间、活跃时间、销毁时间等),与执行交互事件相关的性能信息(CPU 使用率、内存使用率等)
交互行为安全性监测	构件遭受恶意攻击的次数
交互行为有效性监测	交互次数、交互行为基本信息
交互行为可靠性监测	构件不能提供正常服务的失效间隔时间
交互行为可用性监测	构件不能提供正常服务的失效间隔时间、构件不能提供服务的失效恢复时间
交互行为时效性监测	交互事件开始执行时间、交互事件结束执行的时间

2 基于 AOP 的性能监测系统及其实现

2.1 系统结构

本文设计实现的 AOP 性能监测系统结构如图 1 所示,它由 3 个部分构成:目标监测系统、独立监测控制台和性能监测构件包。其中可视化部分——独立监测控制台是辅助实现动态织入的关键,也是连接目标监测系统和性能监视构件包的桥梁,它提供性能数据读写接口以实现标准化的性能数据采集,以及为目标系统扩展 IoC 功能提供织入控制容器。性能监测构件包提供了基本标准接口,监控器的类型和功能可以根据实际项目需要扩展新的监测组件。

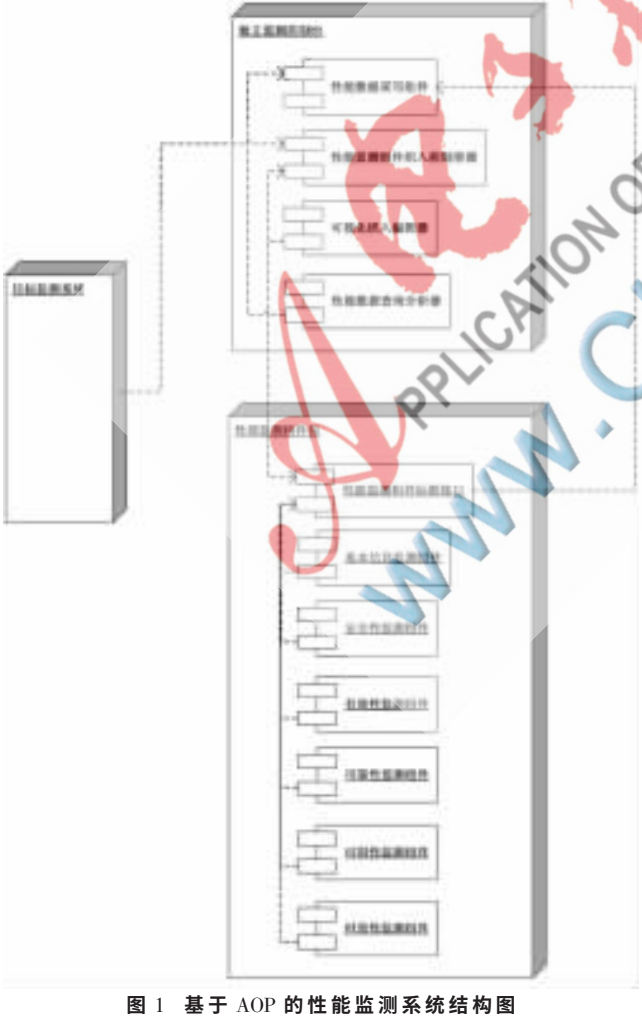

图 1 基于 AOP 的性能监测系统结构图

2.2 相关技术介绍

本系统主要采用 AOP 技术实现监测代码和业务代码松散耦合,并可在目标系统运行过程中动态添加或删除监测器。本系统主要采用的开发工具有 Asp.net MVC、NInject.WEB.MVC、WMI。

面向方面编程(AOP)是美国施乐公司帕洛阿尔托研究中心在 20 世纪 90 年代提出的一种编程思想。AOP 技术作为一种新兴的软件开发方法,提供了不同关注点的分离实现机制,具有可扩展性、可复用性、易理解性、易维护性等特性。本监测器利用 AOP 关注点分离原则,实现监测代码与业务代码松散耦合,提高目标系统的可扩展性、可复用性、易维护性;使用 AOP 的动态织入机制,允许动态添加、删除监测器,从而实现监测的可控性与动态性。

Asp.net MVC 框架将模型、视图、控制器引入开发模型,将 Web 系统跟踪从页面层次提升到了面向业务的层次。NInject 是一个超轻量级的依赖注入框架,具备灵活、自由的织入机制,其扩展框架 NInject.WEB.MVC 是专门为 Asp.net MVC 设计的扩展包。除了一般 IoC 所具备的构造函数、方法、属性注入之外,还提供了针对 Controller 生命周期的注入及扩展的 MvcInjectApplication 类。WMI 是 Windows 的核心管理技术,本系统中多处需要提取操作系统及硬件信息,而在 Windows 操作系统环境下,WMI 编程接口具有良好的可操作性,因此本系统的交互行为监测组件主要使用该技术实现。

2.3 系统的实现

本系统使用 Visual Studio.Net2010 开发,使用 Asp.net MVC2.0 和 NInject.Web.MVC 技术实现。

一般的 Web 系统由于页面级别和类级别往往不能包含一个完整的业务功能,因此在实现对业务功能级别的性能监测上存在着一定的复杂度。本系统围绕 Asp.net MVC 框架而实现,它改变 Web Form 框架依赖页面的 Web 架构模式,按照控制器和动作的方式将业务功能进行逻辑划分,使得 IoC 注入方式增加了控制器注入、动作注入、控制器异常注入、动作异常注入等几种方式,使得 Web 系统的性能监测粒度范围有了更实际的业务价值。图 2 为 Asp.net MVC 的执行流程。

AOP 技术中存在多种编译时机和织入方式,由于本

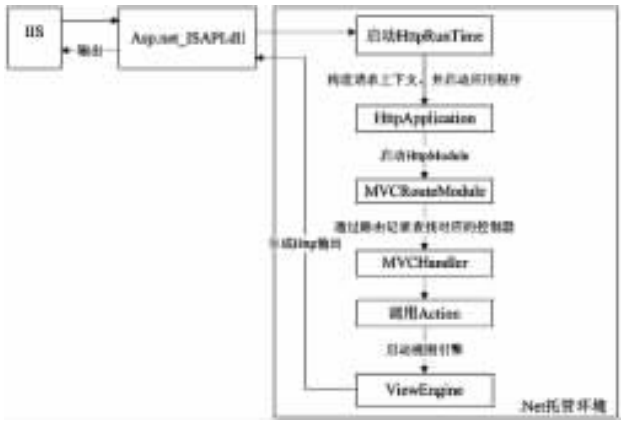

图 2 Asp.net MVC 执行流程图

系统所设计的运行环境一般不能获取被监测系统的源代码，也不能对其进行修改和重编译。因此，采用运行时的动态织入技术实现监测组件到被监测系统指定连接点的织入。

在.NET 平台下的多种 IoC 框架中，NInject 框架最符合本系统的设计要求。它提供了对 Asp.net MVC 系统的良好基础扩展，而且只提供运行时织入。本系统中，通过 NInjectMvcApplication 类对 HttpApplication 的替换，实现了由 NInject 的 IoC 容器替换原本 Web 系统应用程序的目的，从而使系统内对象的创建全部受到 NInject 容器的控制。其替换过程示意图如图 3 所示。

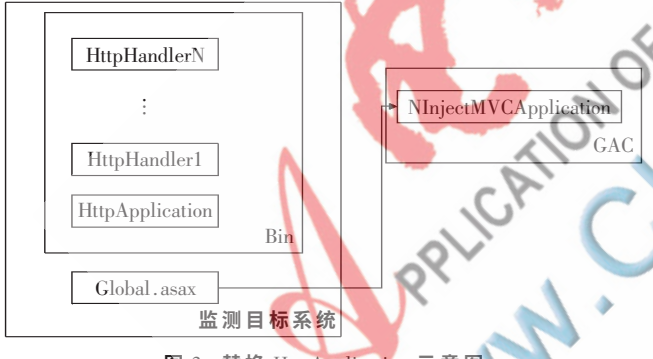

图 3 替换 HttpApplication 示意图

该替换过程首先将 NInjectMvcApplication 类注册到公共运行库(GAC)，而后搜索目标 Web 系统的全局配置文件并替换其 HttpApplication，其关键代码逻辑如下：

```
public void ReplaceIoContainer(string path)
{
    if(string.IsNullOrEmpty(path)) throw new FileNotFound-
                                        Exception(path);
    var files = Directory.GetFiles(path);
    foreach(var file in files)
    {
        if(file.ToLower().Equals("global.asax"))
        {
            using(var sr=new StreamReader(file))
            {
                var data = sr.ReadToEnd();
                Regex reg=new Regex("inherits=\"(\\w+)
\"",RegexOptions.IgnoreCase | RegexOptions.Multiline);
                var mat = reg.Match(data);
                if(mat.Length==0) throw new InvalidOpera-
                                        tionException();
                reg.Replace(data, typeof (NInjectMvcAppli-
                                        cation).FullName);
                using (var sw=new FileStream(path,
                                        FileMode.Open))
                {
                    var writeData = Encoding.UTF8.Get-
                                        Bytes(data);
                    sw.Write(writeData,0,writeData.Length);
                }
            }
        }
    }
}
```

替换 IoC 容器后可按配置将监测组件按需进行织入，织入连接点为构造函数、方法、属性、控制器、动作等，以常量进行标识，其关键代码如下：

```
                                        /// <summary>
        /// 检测器的插入位置(可以用或符号多重选择)
                                        /// </summary>
[Flags]
public enum MonitorPoint
{
    mpBeforeContructor,mpBeforeMehtod,mpBeforeProperty,
                                        mpAfterContructor,
    mpAfterMethod,mpAfterProperty,mpBeforeController,
                                        mpSurroundController,
    mpAfterController,mpBeforeAction,mpSurroundAction,
                                        mpAfterAction,mpActionException
}

public interface IMonitorContainer
{
                                        /// 注入的位置
    MonitorPoint InjectPosition { get; set; }
                                        /// 所要注入的监视器
    MonitorBase Monitor { get; set; }
                                        /// NInject 注入引擎
    IKernel Component { get; set; }
}
```

通过连接 Ikernel 和 MonitorBase，容器将监测组件织入到目标系统指定位置。由于监测组件存在多样性，本系统为使监测组件设计具有更好的扩展性，做了以下工作：

(1)标准化织入目标连接点方式：任何拓展的监测组

件，都必须支持 IoC 容器所提供的所有织入点的监测工作；

(2) 预留监测数据输出接口：由于监测组件的多样性，其数据产生量也有不同的变化，因此预留了多个监测数据输出接口(例如文本中 HTTP、数据库、XML、电子邮件等)。

图 4 列出了监测组件的可拓展接口。

本文设计并实现了一款运行在作业环境下的代码级的 Web 性能监测分析工具。它采集实时数据进行分析；使用 AOP 技术实现了 Web 应用系统的代码级监测分析，并且可自由控制监测范围；使用构件技术实现了即插即用，减少了对现有系统的修改和结构上的破坏，避免因性能监测而引入新的 BUG；通过控制台自由配置变更监测的范围，灵活调整监测的目标；利用 Windows 性能计数器显示检测和分析的结果。该工具对系统进行监测分析时，不需要对作业系统进行任何重编译的工作，尽量地降低性能监测所带来的额外损耗。下一步将对构件包进行扩展，使其能够应用在更多的系统上，实现多个被测系统的联合监测。

图 4 监测组件扩展接口图

参考文献

[1] 朱幸辉，杨树强.基于 CORBA 分布式对象监测系统的研究与实现[J].计算机应用研究，2005，22(10)：39-41.

[2] 万灿军，李长云.基于动态 AOP 的构件交互行为监测器[J].计算机应用，2011，31(2)：572-576.

[3] 黄兴华，胡飞.AOP 技术在面向用户的软件组件测试中的应用[J].计算机应用与软件，2009，26(8)：125-127.

[4] 万灿军，李长云.动态演化环境中可信软件行为监控研究与进展[J].计算机应用研究，2009，26(4)：1201-1204.

[5] 万灿军，李长云，贺宗梅.分布式软件的交互行为监测机制的研究[J].计算机工程与应用，2011，47(5)：60-64.

(收稿日期：2012-02-24)

作者简介：

唐雅璇，女，1987 年生，研究生，主要研究方向：软件工程，软构件。

余金山，男，1952 年生，教授，主要研究方向：软件工程，软件复用。