POP-PHP:支持PHP应用的在线集成开发环境

杨 楠 吴 凌 王千祥

(北京大学信息科学技术学院软件研究所 北京 100871) (软件工程国家工程研究中心 北京 100871)
(高可信软件技术教育部重点实验室 北京 100871)

摘 要 随着云计算的发展,随时随地开发程序成为许多人的一种新的愿景。因此在线集成开发环境受到了软件开发人员的广泛关注。POP-PHP(Peking University Online Programming-PHP version)是一个支持PHP应用的在线集成开发环境,具有基本的集成开发环境功能,支持多用户同时使用,并提供了一种轻量级调试方法。其采用服务组合实现了较为完善的语法检查功能,并实现了编程行为回放以进行监测。

关键词 PHP应用,在线集成开发环境,程序调试,服务组合,编程行为监测

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.09.006

POP-PHP:Online Integrated Development Environment for PHP Applications

YANG Nan WU Ling WANG Qian-xiang

(Institute of Software,School of Electronics Engineering and Computer Science,Peking University,Beijing 100871,China)
(National Engineering Research Center for Software Engineering,Beijing 100871,China)
(Key laboratory of High Confidence Software Technologies,Ministry of Education,Beijing 100871,China)

Abstract With the development of cloud computing,developing programs anytime and anywhere becomes a new vision for many people. As a result,online integrated development environment receives wide attention from software developers. POP-PHP (Peking University Online Programming - PHP version) is an online integrated development environment for PHP Applications and has the basic function of integrated development environment and good support for large number of users simultaneously. It provides a lightweight debugging method,and uses the services composition to achieve a more perfect grammar checking and implements programming behavior playback for monitoring.

Keywords PHP application,Online integrated development environment,Debugging,Services composition,Programming behavior monitoring

1 引言

"云计算"作为一种新的计算模式,为用户提供可靠的、可定制的、服务质量有保证的新型计算环境[1]。"云应用"的出现进一步帮助用户降低IT成本,大大提高工作效率,它把传统软件"本地安装、本地运算"的使用方式变为"在线使用"的方式,通过互联网或局域网连接并操控远程服务器集群,完成业务逻辑或运算任务。目前云应用已经为许多企业级用户的核心业务提供了支持[2]。在线办公软件、云储存等应用的出现标志着传统软件正在向云应用转型。

在"云计算"发展的新形势下,如何利用Internet为程序开发者提供更便捷的开发环境成为越来越多人所关注的内容。在线集成开发环境因此受到了软件开发人员的广泛关注[3]。

在线集成开发环境相对于本地集成开发环境有很多特有的优势,主要包括以下几点。首先,由于代码的存储和运行都在云端,开发者只需要一个装有浏览器的设备即可以轻松实现随时随地开发程序;其次,目前项目的开发呈现分布式的趋势,不同成员之间开发过程的交流非常重要[4],而在线环境下同一项目组的成员可以方便地了解到其他人的代码和开发进度,便于协作推进项目;最后,在线集成开发环境可以辅助管理者通过对使用者编程行为的挖掘,指导使用者的编程过程[5],并获得相关的监测数据。

在2014年1月的TIOBE编程语言排行榜[1]中,PHP语言位居第六[6](前4名为Java、C及C的扩展语言),可见其在Web应用开发中具有重要的地位。在线集成开发环境可以对PHP应用开发提供良好的支持,这是因为作为Web应用,PHP应用通过浏览器访问,所以在浏览器中开发,即可非常方便地直接看效果;同时由于开发时程序可以直接通过网络部署到服务器上,完全避免了开发与运行环境不一致导致的系统问题。

我们调研了目前已有的一些在线集成开发环境,如Co-

1) http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

到稿日期:2013-12-25 返修日期:2014-01-16 本文受国家自然基金(61033006,61121063),国家高技术研究发展计划(863计划)(2011AA01A202,2013AA01A213)资助。

杨 楠(1991-),女,硕士生,主要研究方向为软件工程、中间件,E-mail:yangnan13@sei.pku.edu.cn;吴 凌(1985-),男,博士生,主要研究方向为程序分析、系统软件、中间件;王千祥(1970-),男,博士,教授,主要研究方向为软件工程、中间件。

deRun[1]、Cloud9 IDE[2]等，发现它们对 PHP 应用的支持并不完善：(1)缺少调试功能，即只能将项目部署到服务器上，无法提供开发所需的调试功能；(2)没有完善的语法检查功能，也没有代码提示等辅助功能；(3)不能记录和分析编程行为数据。这些为 PHP 应用开发者和管理者带来了不便。

为解决以上问题，本文设计并实现了一个支持 PHP 应用的在线集成开发环境 POP-PHP(Peking University Online Programming - PHP version)。本文主要贡献如下：

1)该系统具有基本的集成开发环境功能，可良好支持多用户同时使用；

2)系统实现了一种新的在线 PHP 调试方法，即轻量级调试；

3)系统采用服务组合[7]实现了较为完善的语法检查功能；

4)系统实现了用户编程行为的回放，可进行简单的行为监测。

本文第 2 节对相关工作进行了介绍；第 3 节对 POP-PHP 系统的总体结构和各模块功能进行了具体介绍；第 4 节给出了系统实现的 3 种关键技术；第 5 节介绍了对系统进行的相关实验，并对系统存在的问题进行了讨论；最后进行了总结和展望。

2 相关工作

现有的在线集成开发环境各有特点且都处于不断的发展完善中。

CEclipse(Cloud Eclipse)[8]将 Eclipse 的功能封装成服务，利用服务组合形成在线集成开发环境的核心功能，包括 Java 语言的代码编辑、代码提示、运行和调试等。

Java WIDE[3]是一款特定于 Java 语言的在线集成开发环境，具有代码高亮和提示功能[9]，其主要着眼于用户间的协同开发。

CodeRun 是一个基于 JavaScript 语言开发的跨平台的集成开发环境，支持 C# 代码高亮、补全、调试和部署。

Cloud9 IDE 是基于 NodeJS 构建的在线集成开发环境，支持常用开发语言的高亮，提供内置的 Vim 模式。

除了这些严格定义上的集成开发环境之外，还有 Bespin[4]、Codeanywhere[5]等在线编辑器，它们虽然不支持代码的运行或部署，但代码高亮等功能都很完善，同时也具有分享功能。

上述在线集成开发环境中，CEclipse 和 JavaWIDE 都只支持 Java 语言，后两个都支持 PHP 语言，包括 PHP 代码编辑、项目文件的管理和部署等。但相较于开发者的需求而言，这些在线集成开发环境对 PHP 语言的支持仍有不足之处，包括没有调试和语法检查功能等。事实上，CodeRun 主要着眼于 C# 语言的开发，而 Cloud9 IDE 更重视 JavaScript 语言的编辑和调试等，所以目前没有一个特定于 PHP 语言的完善的在线集成开发环境。

为了弥补这一缺陷，我们提出了 POP-PHP 系统，它包含适用于在线环境的轻量级调试功能、基于服务组合的语法检查功能，提供了对编程行为的回放，良好地解决了以上问题，并为 PHP 应用开发者和相应的课程教学提供了帮助。

3 系统介绍

POP-PHP 系统界面如图 1 所示，包括工具栏、项目管理面板、代码编辑区、大纲面板、控制台。以下对系统的结构和功能进行说明。

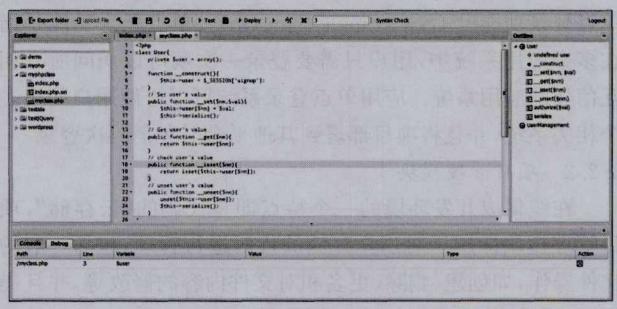

图 1 POP-PHP 用户界面

3.1 总体结构

POP-PHP 系统逻辑上采用三层架构[10]设计，划分为：表现层、业务逻辑层、数据访问层，系统结构如图 2 所示。

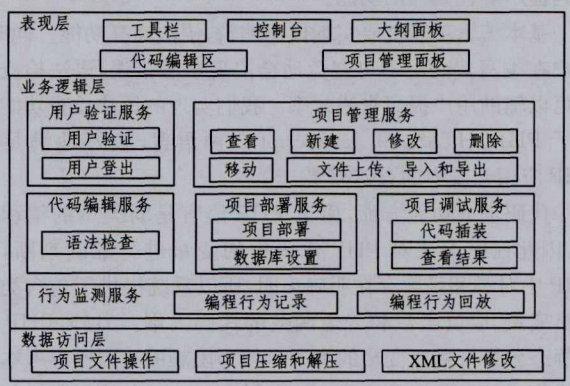

图 2 POP-PHP 的系统结构

表现层进行与项目管理、代码编辑、项目部署调试相关的用户交互，动态显示大纲和控制台内容，以及调用后台服务。

业务逻辑层使用 Java 语言开发，将各个功能封装为相对独立的服务，每个服务当中又细化为不同的操作。业务逻辑层的这种设计可以保证各个功能之间的松耦合，便于单独修改维护，也提升了系统扩展性，可较为容易地新增服务，丰富系统功能。

数据访问层主要是对文件的操作，包括项目文件的增、删、改、查等基本操作，项目的压缩和解压以及对 XML 文件的读取和修改操作。

以上 3 层有机地结合在一起，这样的架构可以降低层与层之间的依赖，利于各层逻辑的复用，很容易用新的实现来替

1) http://www.coderun.com/

2) https://c9.io/

3) http://www.javawide.org/index.php/Main_Page

4) http://bespin.mozilla.com/

5) https://codeanywhere.net/

换原有层次的实现；也使得系统的整体结构更加明确，在后期维护时，可以降低维护成本和维护时间。

3.2 各模块功能

根据系统结构中各部分对应的不同功能，可将系统划分为6个模块：用户验证模块、项目管理模块、代码编辑模块、项目部署模块、项目调试模块和行为监测模块。

3.2.1 用户验证模块

用户验证模块用于验证用户身份，只有通过系统验证的用户才能进行访问和使用，该模块主要通过过滤技术对访问的请求和响应进行拦截处理。同时我们的用户验证模块提供单点登录功能[11]（SSO，Single Sign On）。单点登录功能是指在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统。应用单点登录系统可以方便用户使用多个相关系统，并且将项目部署到其他平台时无需再次登录。

3.2.2 项目管理模块

在线集成开发环境的一个特点即是代码的“云存储”，项目管理模块即用于对用户存储的代码进行管理，包括一般的文件操作，如创建、删除、更名和对文件内容的修改等，并且提供了文件上传和项目的导入导出功能。

3.2.3 代码编辑模块

代码编辑模块用于为用户提供代码编辑基本功能，以及对PHP、HTML和JavaScript语言的语法检查和代码大纲、代码提示等代码辅助功能。

基本编辑功能主要实现代码编辑、语法高亮功能。同时，用户在编写代码时不可避免地会出现语法错误，语法检查功能可以辅助用户提高查错效率。我们实现的语法检查功能涵盖了PHP、HTML和JavaScript 3种语言，较好地满足了PHP应用开发中的查错需要。

代码辅助功能方面，由于PHP语言是弱类型的编程语言，因此程序无法对PHP代码中的变量进行准确判断，但PHP中对类和函数存在明确声明，所以系统提供了包含类中成员变量、成员函数和普通函数的代码大纲。另外，PHP通过静态分析来判断变量的类型是较为困难的。因而参考本地在线集成开发环境的解决方法，系统对PHP的库函数进行了提示，以实现自动补全。

3.2.4 项目部署模块

项目部署模块用于将项目部署到服务器上，而在部署时如果需要使用数据库，用户还可以申请数据库，系统将提供一个用户名和密码均为当前用户名的帐号，并自动建立相应的数据库提供给用户使用。用户可通过访问服务器上数据库管理工具phpMyAdmin的链接来实现对自己数据库的管理。

3.2.5 项目调试模块

项目调试模块用于对项目进行轻量级调试。由于PHP语言是解释型语言，在线环境下传统的调试方法并不适用，因此在线集成开发环境中对PHP应用的调试不易实现。在项目调试模块中，我们提出了一种新的PHP语言调试方法，称为PHP轻量级调试（PHP Lightweight Debugging），它基本上可满足开发者在项目查错过程中的需求。接下来将会对该技术的实现进行详细介绍。

3.2.6 行为监测模块

行为监测模块用于记录用户的编程行为，并以编程过程回放的形式展现给管理者。该模块可以具体应用于教学中，学生完成作业时的编程行为由系统进行记录，教师可查看学生的行为，了解学生在编程中经常遇到的问题，同时也可以根据系统的简单提示判断是否有抄袭等不良行为。

4 关键技术

POP-PHP系统主要使用了3项关键技术，即PHP轻量级调试、基于服务组合的语法检查和用户编程行为回放，弥补了已有在线集成开发环境对PHP应用支持的不足之处。

4.1 PHP轻量级调试

PHP轻量级调试方法是基于对Web应用开发者编程习惯的观察而提出：在编写PHP应用时，如果出现问题，一般开发者会在代码中插入输出语句从而输出关键变量的值，判断应用运行到该处时是否出错。

这种新的调试方法主要思想即是将开发者的这种手动插入输出语句过程自动化，其基本流程如图3（图中对每个步骤涵盖的处理块进行了标注）所示。

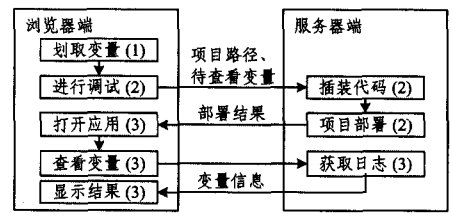

图3 PHP轻量级调试流程图

3个步骤分别为：

1. 变量选择：开发者使用鼠标在代码中划取需要查看的变量，这些变量显示在浏览器中，便于开发者查看管理；

2. 代码插装：开发者点击调试按钮，浏览器将项目路径和划取的变量发送到服务器；服务器在文件中找到这些变量，并在其下一行插入输出语句以将变量输出到日志文件中，再将修改后的代码部署到服务器上；

3. 结果显示：开发者打开应用页面时，应用运行，将变量信息写入日志文件中；开发者点击查看变量，向服务器发送指令，服务器获取日志文件，返回变量信息到浏览器端并显示。

以下对各个步骤进行详细说明。

4.1.1 变量选择

轻量级调试的最终目的是将用户想要查看的变量信息提供给用户，因而第一步需要为用户提供选择待查看变量的功能。在浏览器端用户可使用鼠标划取或双击选择编辑区中的变量，这即为用户通过该调试功能想要获取的信息变量。

实现该功能时，需要在选择单词后触发一个函数判断该词是否为变量，例如PHP语言中，若以“$”开头，则该词为变量。接下来触发选择函数，获取变量相关信息，并将其显示在浏览器中。

同时，在用户友好性方面，我们将变量存放于调试控制台中，用户可以直接点击某一项使之跳转到变量所在的行，实现快速定位，便于变量的管理。而对于循环中的变量也提供了用户自定义输出次数的功能，既方便用户的调试，也可以通过设置默认最大值应对恶意代码中的死循环情况。

4.1.2 代码插装

代码插装的流程是：变量选择完毕后，用户点击调试按钮，相关信息被发送到服务器端进行代码插装处理。代码插装即将输出语句插入到代码中的适当位置，使得应用正常运行的同时，这些输出语句得以执行，从而将用户想要查看的变量信息输出到日志文件中。

从该步骤的基本功能中可见，自动添加输出变量代码的过程需要进行语法分析，主要涉及的语法树节点有 PHP 页面、语句结束符、复合语句、条件语句、迭代语句、跳转语句和变量。遍历语法树时，需要同时进行两方面的处理：1）确定变量，对于每个变量节点，如果其行、列值与待查看变量相同，则将其存放于一个数组中等待处理；2）插入语句，如果语句结束后还有未处理的变量，则下一行需要插入输出该变量信息的语句。

例如，我们插入的输出语句为图 4 代码。

```
error_log(genstr(2, 12, "\$a", $a, gettype($a)), 3, "debug.log");
```

图 4　PHP 轻量级调试插入代码示例

该语句表示将第一个参数中的字符串发送到文件目标 debug.log。函数 genstr()生成包含文件路径、行号、列号、变量名、变量值、变量类型信息的 JSON[1]（JavaScript Object Notation，一种轻量级的数据交换格式）数据，因而需要在 PHP 文件开始处插入图 5 所示代码。

```
function genstr($line, $column, $name, $value, $type) {
    return "{\"path\":\"/index.php\",
        \"line\":$line,
        \"column\":$column,
        \"variable\":\"$name\",
        \"value\":\"$value\",
        \"type\":\"$type\"},";
```

图 5　PHP 轻量级调试文件开始处所插入代码

而代码插装过程中还应对特殊语句进行特别处理：

1. 条件语句：如果条件体内只有一条语句需要执行，那么条件表达式后面的大括号可能已被省略，若待查看的变量位于条件体唯一的这条语句中，则在条件体内语句之后插入一条输出语句将造成代码结构错误。如果用户查看的变量包含于条件表达式中，直接插入语句也会发生错误。

解决方案如下：在条件语句节点，为所有非复合语句的条件体（即未被大括号包围的语句）的前后添加大括号语句，并且将迭代语句节点中的处理延迟到复合语句节点中，即进入到条件体内，在大括号的下一行插入输出语句。具体的示例如图 6 所示。

```
if (condition)
    code to be executed if condition is true;
    // 不能在此处直接插入语句
    // 需要用大括号包围条件体
elseif (condition)
// 不能在此处直接插入语句
// 需要进入到条件体内，在大括号下一行插入
{
    code to be executed if condition is true;
}
else
    code to be executed if condition is false;
```

图 6　条件语句插入代码特殊情况示例

2. 迭代语句：同选择语句一样，迭代语句也会出现同样的两个问题，其处理方式类似。另外，迭代语句中的变量信息会被输出多次，我们对其次数进行了限制，用户也可自定义该值，从而控制输出语句的执行次数。

解决方案如下：迭代语句节点中，在插入的输出语句前后添加临时变量进行计数，达到设定值后即不再执行。例如对于一个实际次数为 10 的循环，如果设定的次数为 3，则只会在前 3 次循环中执行输出语句得到变量信息。

3. 跳转语句：如果待查看变量包含于跳转语句中，那么在这条语句结束之后插入的输出语句将不会被执行。

解决方案如下：将应返回的表达式赋值给临时变量，再插入输出语句，之后将临时变量返回。从而正确返回了表达式，也使得输出语句得以执行。

4.1.3　结果显示

插入输出语句之后的代码被部署到服务器上，并返回部署结果。用户通过链接查看 PHP 应用时，文件运行即把变量信息写入日志文件。在用户点击查看调试结果后，浏览器端即向用户显示从日志文件中获取到的变量信息，完成整个调试过程。

我们显示的信息包括变量位置、名称、值和类型，并可以进行具体的定位和管理，效果如图 7 所示。

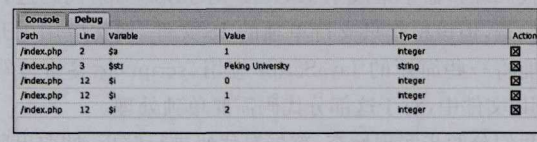

图 7　项目调试结果显示

4.2　基于服务组合的语法检查

目前已有多种本地语法检查工具，我们可以通过服务组合的方法将其应用于在线集成开发环境中。即将语法检查工具封装为服务，调用不同的服务获得返回结果。服务组合具有如下优点[12]：

1. 松耦合：各个语法检查工具之间较为独立，互相影响小，便于单独修改和维护。

2. 扩展性：系统的扩展性得以提升，可较为容易地新增服务，丰富系统功能。

以下对各语言的语法检查实现进行详细说明。

4.2.1　PHP 语法检查

目前对 PHP 的语法检查没有良好的浏览器端运行的程序可以提供支持，因而可以将本地使用的语法检查工具封装为服务，提供给浏览器端。

PHP 命令行模式[2]的“-l”参数[13]可以对指定 PHP 代码进行语法检查。我们将其封装为服务，从而实现对 PHP 语言的检查，核心实现代码如图 8 所示。

```
String[] cmd = { "php", "-l", phpFile.getAbsolutePath()};
ProcessBuilder processBuilder = new ProcessBuilder (cmd);
Process process = processBuilder.start ();
```

图 8　Java 执行 PHP 命令返回结果代码

同时我们需要对 PHP 语法检查结果进行处理，通过字符串分割抽取出每一条错误信息和其对应的行号，把错误信息直接显示在对应行前，鼠标移动到图标上即可提示详细错误

1) http://www.json.org/

2) http://www.php.net/manual/en/features.commandline.php

信息，效果如图 9 所示。

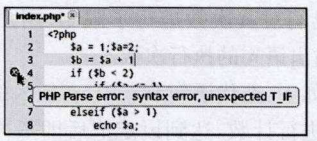

图 9　语法检查浏览器端结果显示

4.2.2　HTML 语法检查

PHP 应用开发时，服务器端使用 PHP 语言，语法检查功能可以帮助开发者较快地发现错误。同时浏览器端一般使用 HTML、JavaScript 等语言，对于这些前端开发语言，开发者很多时候难以发现代码中存在的问题，从而影响 Web 应用的正常运行。所以除了对 PHP 语言的语法检查，系统还提供了对 HTML 和 JavaScript 语言的语法检查。

对 HTML 进行语法检查的过程与 PHP 检查类似，将 HTML Tidy[1] 的检查过程封装为服务。其不同之处在于由于该工具为开源工具，我们可以获得源码，因而不需要单独创建线程，直接调用它的函数。将其封装为服务后，可以得到的数据包括警告、错误提示以及建议的符合规范的代码。

4.2.3　JavaScript 语法检查

代码编辑器中集成了 JSHint[2] 对 JavaScript 代码进行实时的检查，但该检查只支持单独的 JS 文件，现实应用中开发者可能将一些简短的 JavaScript 应用〈script〉标签直接写在 HTML 文件中，对于这部分代码需要单独处理。

我们分两步实现检查：首先对代码进行解析，抽取出所有内嵌的 JavaScript 代码；接下来使用 JSHint 对得到的 JavaScript 代码进行语法检查，结果连同 HTML 检查的结果一起显示于编辑区中。

4.3　用户编程行为回放

用户编程行为回放功能主要是基于实际教学需求实现的。在 PHP 应用开发相关的课程中，教师不仅关注学生作业的最终结果，也同样关心学生编程的过程，因而实现对编程行为的监测是必要的。

目前的系统首先实现了行为的回放，将所有记录下的编程行为按时间顺序呈现给教师，完整地展示了学生编程的整个过程。同时进行了简单的分析，辅助教师发现学生的不良行为。在之后的工作中，我们还将通过数据提出更多的分析方式，提供更进一步的监测结果。

4.3.1　编程行为记录

系统对用户编程行为的记录分为两种：1）编写代码的行为记录；2）对文件的操作记录。对于第一种记录，我们通过调用 ACE 编辑器的内置函数，获取用户的输入行为，包括插入(insertText)、插入多行(insertLines)、删除(deleteText)、删除多行(deleteLines)4 种类型。而对于第二种记录，通过操作触发的函数进行记录，系统目前记录了保存文件、关闭文件两个操作。

这些记录既包括相应的操作，也包括操作对应的文件路径、时间和其他信息。尽可能完善地记录编程行为，保证之后能够完整地回放编程过程并进行具体的分析。获取到的记录暂时保存在浏览器端，当用户对文件进行保存时，这些记录被传到服务器端，并写入数据库。

4.3.2　编程行为回放

查看回放时，用户界面与编写代码基本相同，工具栏的显示根据功能有所变化(见图 10)。用户通过项目管理面板可以选择需要查看的文件。服务器端从数据库读取该文件相关的记录，由浏览器端按时间顺序依次还原这些操作。主要显示的为第一种记录，即用户对代码文件的编辑，而第二种记录作为回放时的辅助信息。

图 10　行为回放工具栏显示

回放过程中，滑动条中的滑块随时间向前移动，当抵达用户进行过一次操作的时间点时，编辑器中插入或删除一段内容。在此过程中，可以暂停和停止；可以向前拖动滑块，取消时间控制，直接还原期间的所有操作，实现快进；也可以向后拖动滑块，将已经回放过的操作进行逆操作，如之前插入了一段代码则将该段代码删除，从而实现快退。

回放的同时，系统还会进行简单的代码分析。当出现大段插入代码的情况(插入过长的一行代码或插入过多行代码)时，系统将在对应行前出现提示(见图 11)，提醒观看回放的用户注意此处可能存在大段代码的复制。这有助于教师判断学生在完成作业过程中是否进行复制抄袭。

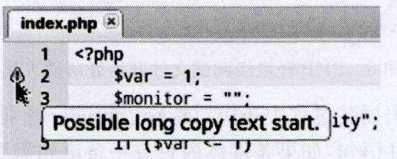

图 11　回放时的抄袭提示

5　实验和讨论

为检查系统的并发量和事务响应时间，我们进行了 3 次实验。

5.1　实验过程

3 次实验的过程如下：

实验 1　基本覆盖了所有用户需要向服务器发送请求的事务，包括载入页面，项目管理列表初始化，新建工程，新建、查看和保存文件，语法检查，部署配置和部署项目，调试开始、部署、结束和查看结果，以及重命名、上传和删除文件共 16 个事务，并发量设置为 100 个用户。

实验 2　与实验 1 基本相同，但将并发量增加到 150 个用户。

实验 3　测试了用户最为常用的两项操作——查看文件和编辑后保存文件的性能，并发量为 200。

通过以上 3 次实验的比较，我们从结果中对系统在并发量方面的性能进行评估，并分析各个事务的响应时间，从而掌握未来系统可改进的方向。

5.2　实验结果

5.2.1　系统并发量

实验 1 中 100 个用户正常运行，没有出现错误，每秒吞吐

[1] http://tidybatchfiles.info/

[2] http://www.jshint.com/

量和每秒点击数处于可满足用户需求的值范围内(见图 12)。

Statistics Summary

Maximum Running Vusers:	100
Total Throughput (bytes):	410,106,930
Average Throughput (bytes/second):	2,908,560
Total Hits:	6,000
Average Hits per Second:	42.553

图 12　实验 1 统计信息摘要

实验 2 中的并发量达到 150,每秒点击数和每秒吞吐量未发生较大改变,但运行过程出现了 17 个同样的错误(见图 13)。从服务器下载文件过程中出错,说明并发量增大后,出现超时。

Statistics Summary

Maximum Running Vusers:	150
Total Throughput (bytes):	611,244,098
Average Throughput (bytes/second):	3,167,068
Total Hits:	8,462
Average Hits per Second:	43.845
Total Errors:	17

图 13　实验 2 统计信息摘要

实验 3 通过了 200 个并发量的测试(见图 14)。所以服务器对并发量的支持取决于各个用户进行的操作,当操作较为易处理时,并发量较高;而当处理过程复杂时,并发量略低。而实际应用中,各个用户在不同时间的操作是不同的,当一部分用户编辑文件时,可能另一部分文件在进行保存、语法检查等操作,因而综合 3 个实验来看,系统应可正常支持 100 到 200 个用户同时访问。

Statistics Summary

Maximum Running Vusers:	200
Total Throughput (bytes):	39,009,562
Average Throughput (bytes/second):	1,393,199
Total Hits:	800
Average Hits per Second:	28.571

图 14　实验 3 统计信息摘要

可见,POP-PHP 系统除实现了在线集成开发环境的基本功能及其他辅助功能之外,还可以良好地支持多用户使用,体现出在线环境的特有优势。而在今后的应用中,系统可以设计负载均衡策略,以提升响应时间和处理效率。可采取的一种方式是使用分发机制,达到负载均衡的目的。当服务器负载过大时,分发中心会自动到较小的虚拟机上;或者通过添加新的虚拟机,对整个平台进行动态扩展[8]。

5.2.2　事务响应时间

而在事务响应时间方面,以实验 1 为例,事务数为 16、并发量为 100 时,事务信息摘要如图 15 所示。表中"90 Percent Time"较为准确衡量了测试过程中各个事务的真实情况,表示对于 90%的事务服务器的响应都维持在某个值附近。多数事务可以在很短的时间内进行响应,而有一些事务较为耗时。

响应时间最长的是页面载入过程,并发量为 150 时错误也出现在该事务中。这是因为目前系统处于开发阶段,所有浏览器端使用的 JavaScript 文件都是完整未压缩的,当用户第一次载入页面时,这些文件都需要从服务器端下载到本地,所以需要较长时间。因而,当系统投入使用时,浏览器端的 JavaScript 文件需要经过压缩再发布,同时当用户浏览器中已有页面缓存时,载入时间将大大缩短。

Transaction Summary

Transactions: Total Passed: 1,900 Total Failed: 0 Total Stopped: 0　　Average Response Time

Transaction Name	SLA Status	Minimum	Average	Maximum	Std. Deviation	90 Percent	Pass	Fail	Stop
Action_Transaction		88.821	116.64	131.645	11.525	128.543	100	0	0
createfile		0.003	0.054	3.003	0.303	0.029	100	0	0
createproject		0.004	0.039	0.362	0.078	0.048	100	0	0
debugdeploy		1.814	27.209	35.922	6.688	33.618	100	0	0
debugend		0.003	0.018	1.274	0.126	0.01	100	0	0
debugresult		0.008	15.921	29.286	8.18	24.921	100	0	0
debugstart		0.006	0.028	0.067	0.006	0.031	100	0	0
deploy		13.681	25.315	35.832	4.5	32.156	100	0	0
initexplorer		0.003	0.299	26.112	2.596	0.042	100	0	0
loadpage		7.879	32.505	88.555	14.741	43.459	100	0	0
removeFile		0.004	0.082	3.013	0.422	0.02	100	0	0
renameFile		0.003	0.193	3.211	0.666	0.232	100	0	0
retrievefile		0.275	2.243	6.736	1.196	3.68	100	0	0
savefile		0.003	0.029	0.686	0.092	0.03	100	0	0
setconfig		0.024	7.123	29.282	10.433	25.005	100	0	0
syntaxcheck		0.025	0.037	0.269	0.028	0.052	100	0	0
uploadFile		0.29	3.46	13.033	2.998	7.055	100	0	0
vuser_end_Transaction		0	0	0	0	0	100	0	0
vuser_init_Transaction		0	0	0.002	0	0	100	0	0

Service Level Agreement Legend: Pass　Fail　No Data

图 15　实验 1 实验事务信息摘要

结束语　本文首先介绍了在线集成开发环境的发展情况,在此基础上,分析了目前存在的不足,提出了支持 PHP 应用的在线集成开发环境 POP-PHP。在对系统的总体结构和各模块功能进行介绍后,详细说明了关键技术的实现,包括 PHP 轻量级调试方法、基于服务组合的语法检查和用户编程行为回放。最后对系统的测试和未来系统性能的改进进行了讨论。

POP-PHP 是一个较新的在线集成开发环境,仍存在需要提升和改进之处。主要包括如下几方面:寻找良好的方法提升系统稳定性和安全性;支持协同开发;增强在线集成开发环境的特有优势。

参 考 文 献

[1] Wang L, Von Laszewski G, Younge A, et al. Cloud computing: a perspective study[J]. New Generation Computing, 2010, 28(2): 137-146

[2] Buyya R, Yeo C S, Venugopal S, et al. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility[J]. Future Generation computer systems, 2009, 25(6): 599-616

[3] Goldman M, Little G, Miller R C. Real-time collaborative coding in a web IDE[C]//Proceedings of the 24th annual ACM symposium on User interface software and technology. ACM, 2011: 155-164

[4] Nordio M, Estler H, Furia C A, et al. Collaborative software development on the Web[OL]. http://arxiv.org/abs/1105.0768

[5] van Deursen A, Mesbah A, Cornelissen B, et al. Adinda: a knowledgeable, browser-based IDE[C]// Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering. Volume 2, ACM, 2010: 203-206

[6] TIOBE Programming Community Index for July 2013 [EB/OL]. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[7] Milanovic N, Malek M. Current solutions for web service composition[J]. Internet Computing, IEEE, 2004, 8(6): 51-59

[8] Wu L, Liang G, Kui S, et al. CEclipse: An online IDE for programing in the cloud[C]//2011 IEEE World Congress on Services (SERVICES). IEEE, 2011: 45-52

[9] Jenkins J, Brannock E, Dekhane S. JavaWIDE: innovation in an online IDE: tutorial presentation[J]. Journal of Computing Sciences in Colleges, 2010, 25(5): 102-104

[10] Fan Y, He H. The Research of 2-Tier and 3-Tier Structure Based on the Client/Server Architecture [J]. Application Research of Computers, 2001, 12: 23-24

[11] De Clercq J. Single sign-on architectures[M]. Infrastructure Security. Springer Berlin Heidelberg, 2002: 40-58

[12] Srivastava B, Koehler J. Web service composition-current solutions and open problems[C]// ICAPS 2003 workshop on Planning for Web Services. 2003, 35: 28-35

[13] Using PHP from the command line [EB/OL]. http://www.php.net/manual/en/features.commandline.php

(上接第 31 页)

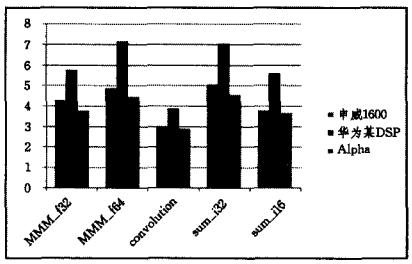

图 5 各平台 SIMD 部件向量化膨胀率

以 int 型为基准，申威 1600、DSP 和 Alpha 的向量化因子分别为 8、4 和 4，由于 DSP 的应用为大量密集的数字计算，其指令集相对复杂，一些指令的设计更为高效，这种高效会带来更高的代码膨胀率。以卷积为例进行分析，convolution 进行外层向量化后，DPS 生成的循环寻址指令能够高效地获取下一个操作数，并通过软件流水获得了时间上的并行性，具有明显的加速效果。

通过实验结果可以看出：

(1)该架构能够对 SIMD 部件的不同指令集自动变换出高效的向量化代码；

(2)指令集越复杂，向量化后的加速效果越好，代码膨胀率越高；

(3)DSP 具有较高的加速比，加速性能高于申威 1600 平均 48.7%，高于 Alpha 平均 173.6%；

(4) 向量化后 Alpha 的膨胀率较小，可读性较好，膨胀率优于申威 1600 平均 8.8%，优于 DSP 平均 51.3%。

结束语 多媒体应用的普及和高性能应用的需求使得越来越多的处理器集成了 SIMD 扩展，本文针对 SIMD 部件的自动向量化提出了一种统一架构，其主要贡献有以下 3 点：

(1) 统一架构能够自动变换为面向 SIMD 扩展部件的向量化代码，让程序员从繁冗复杂的手工向量化编码中解脱出来；

(2) 统一架构适用于不同向量长度和不同向量指令集的 SIMD 扩展部件，避免了针对不同 SIMD 扩展开发不同向量化工具的尴尬局面，有效提高了向量化编译器的开发效率；

(3) 设计的虚拟向量指令集能够支持非对齐访存、控制流、跨步访问、数据重组等向量化中的难点问题，对多媒体应用的自动向量化具有一定的普适性。

当前的虚拟向量指令集仅为特定 SIMD 指令集的抽象和精简，由于诸如 Intel 等平台向量指令集的复杂性，统一架构很难保证面向多种平台均能生成高效的向量化代码。下一步工作中，我们计划将更多的向量化指令特征融入虚拟向量指令集，以提高架构的向量化能力和加速效果。

参 考 文 献

[1] Peleg A, Weiser U. MMX Technology Extension to the Intel Architecture[J]. IEEE/ACM International Symposium on Microarchitecture, 1996, 16(4): 42-50

[2] Intel 64-ia-32-architectures-software-developer-manual [EB/OL]. http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-manual-325462.pdf, September 2013

[3] Stewart J. An Investigation of SIMD instruction sets. School of Information Technology and Mathematical Sciences[R]. University of Ballarat, 2005

[4] Franchetti F, Kral S, Lorenz J, et al. Efficient utilization of SIMD extensions[J]. Proceedings of the IEEE, 2005, 93(2): 409-425

[5] SC140 DSP Core Reference Manual. Freescale Semiconductor [EB/OL]. http:// catch.freescale.com/files/dsp/doc/ref_manual/MNSC140CORE.pdf, 2004

[6] Fridman J, Greenfield Z. The Tiger SHARC DSP Architecture [J]. IEEE Micro, 2000, 20(1): 66-76

[7] TMS320C6000 CPU and Instruction Set Reference Guild(Rev. F)[R]. Texas Instruments Inc. 2000

[8] Allen R, Kennedy K. Optimizing Compilers for Modern Architectures——A Dependence-based Approach[M]. US: Morgan Kaufmann Publishers, 2001

[9] Larsen S, Amarasinghe S. Exploiting superword level parallelism with multimedia instruction sets[C]// Proc of the ACM SIGPLAN Conference on Programming Language Design and Implementation. June 2000: 145-156

[10] Kudriavtsev A, Kogge P. Generation of Permutations for SIMD Processors[C]//PLDI, 2006. Ottawa, Canada, 2006

[11] Eichenberger A E, Wu Peng, O'brein K. Vectorization for simd architectures with alignment constraints[C]//PLDI. June 2004

[12] Wu Peng, Eichenberger A E, Wang A. Efficient simd code generation for runtime alignment[C]//CGO. March 2005

[13] Hiroaki T, Chi Y T, Sakanushi K, et al. Pack Instruction Generation for Media Processors Using Multi-valued Decision Diagram [C]//CODES+ISSS. Seoul, Korea, October 2006: 154-159

[14] Karrenberg R. Whole Function Vectorization[C]//2011 9th Annual IEEE/ACM International Symposium on Code Generation and Optimization. 2011: 141-150

[15] Zhu Jia-feng, Zhao Rong-cai, Han Lin, et al. A Vectorization Method of Export Branch for SIMD Extension [C] // 2011 IEEE/ACIS 10th International Conference on Computer and Information Science (ICIS). 2011: 265-269

[16] Nuzman D, Rosen I, Zaks A. Auto-Vectorization of Interleaved Data for SIMD[C]//PLDI. June 2006: 132-143

[17] 魏帅，魏然，侯永生. 面向科学计算程序的向量化[J]. 信息工程大学学报，2011(6): 759-763, 768

[18] Barik R, Zhao Ji-sheng, Sarkar V. Efficient Selection of Vector Instructions using Dynamic Programming[C]//2010 43 rd Annual IEEE/ACM International Symposium on Microarchitectures. 2010