PHP、Java和C语言中Web服务引擎的性能比较

Tatsubori, Akihiko Tozawa and Tamiya Onodera，Toyotaro Suzumura，Scott Trent，Micaki 东京研究实验室，IBM研究中心，1623-L4岛村，大和市，神奈川-肯242-8502，日本

摘 要：PHP作为Web 2.0时代的编程语言而众所周知，可实现敏捷的服务器端软件开发。它从版本5开始通过基于C的内置库正式支持SOAP消息传递。在本文中，我们在定性和定量方面对PHP作为Web服务引擎的能力进行了深入研究，同时将其与Java和C中实现的其他Web服务引擎进行了比较。我们将Axis2用于此目的，因为它是一个开源Web服务引擎，其实现在Java和C中均可用。我们报告称，作为Web服务引擎的PHP与涉及小型有效负载的Web服务的Axis2 Java具有竞争性，并且大大超过5-17倍的大型有效负载。正如作者所预料的那样，Axis2 C表现最佳，但实验结果表明，PHP性能更接近Axis2 C，具有更大的有效载荷。这种性能差异来自这样一个事实：PHP运行时中的SOAP引擎是在C中用单一体系结构实现的，而Axis2使用更模块化的体系结构来灵活地插入各种WS- \*标准的处理程序，并且Axis2使用称为ADB（Axis2数据绑定）的不同数据绑定机制。本文是第一次尝试比较用PHP，Java和C实现的Web服务引擎，作者认为这可以通过让人们了解其良好的性能得分和高生产率特性来推动PHP中基于SOAP的Web服务的开发。

1 简介

Web服务技术，尤其是基于WS- \*的技术，对于面向服务的体系结构的实现起着关键作用。但是，很难说Web服务在技术首次出现时与假设一样广泛使用。可能的原因包括XML处理的性能问题，太多复杂的WS- \*标准等等。Web 2.0时代出现的最新技术是通过社区驱动的方法产生的，普通程序员在使用它时发挥着关键作用。从这个观点来看，上述不受欢迎的最大原因之一是软件生产率和性能劣势。PHP语言被广泛用作流行的服务器端语言[7]。大量的开源软件和公司的网站使用PHP，因为它可以实现高软件生产力。此外，PHP在第5版中正式提供了SOAP支持。因此，这种支持有助于促进基于SOAP / WS- \*的Web服务的广泛使用。在本文中，我们描述了PHP和Axis2的彻底比较研究，这是从定性和定量的角度来看主要的开源SOAP引擎实现。请注意，我们并不认为PHP比提供Web服务支持的其他编程语言更好，而是我们专注于中性比较。本文的结构如下。第2节描述了Axis2和PHP中SOAP扩展的概述。第3节描述了编程模型，部署模型，WS- \*标准支持和其他功能方面的定性比较。接下来，第4节描述了使用称为StockQuote Web服务和WSTest的基准测试的定量比较，在第5节中，我们总结了PHP，Java和C之间Web服务运行时的比较。我们通过介绍第6节中的相关工作以及第7节中的结论和未来工作来总结本文。

2 Web服务运行时概述

本文的主要目的是演示PHP SOAP引擎与各种编程语言中实现的Web服务引擎的不同之处。今天有许多商业产品和开源软件提供Web服务引擎，但在本文中，我们使用了Axis2项目提供的两个引擎，一个基于Java的实现和一个基于C的实现。正如本文后面详细描述的那样，在软件生产力和部署简易性方面，基于C的实现并不比基于Java和PHP等语言的运行时更直接。但是，我们主要使用它来进行性能比较，以了解最佳性能。接下来，我们将概述PHP及其Web服务运行时和Axis2。

2.1 PHP Web服务器

PHP从版本5开始正式支持SOAP，具有基于C的扩展。以前版本的SOAP引擎可以从用PHP实现的PEAR（PHP扩展和应用程序存储库）或用C实现的PECL（PHP扩展社区库）中获得。有时，PHP被定型为适度执行语言运行时，因为它被解释。但是，PHP可以缓存中间操作码，而且标准的“扩展”库集主要用C语言实现。这些功能使PHP与其他动态脚本语言（如Ruby和Python）不同。

2.2 Apache Axis2

Axis2 [5]是一个开源项目，提供Web服务引擎作为Axis1的后续项目。Axis2的架构从头开始重建，以提供更高的性能意识和灵活性，以支持（各种WS- \*标准。有两种实现，用Java编写的Apache Axis2 / Java和用C编写的Apache Axis2 / C.许多功能将Axis2与Axis1区分开来，例如支持各种WS- \*标准和异步Web服务。Axis2的另一个值得注意的特性是它的REST支持。启用REST选项后，单个操作将返回XML消息或SOAP信封。此功能非常重要，因为它减轻了希望以SOAP和REST方式发布其Web服务的程序员的负担。

3 定性比较

本节介绍PHP和Axis2从各种角度提供的Web服务支持之间的比较研究，包括软件架构，XML处理模型，WS- \*标准支持，服务器提供商的编程模型和部署模型，支持用于RESTful服务和其他功能。

3.1 软件体系结构

Axis2的体系结构基于以下概念：与Axis版本1相比，体系结构应该更灵活，可配置。Axis2体系结构具有模块化体系结构，可以使用模块安装附加功能，以支持各种WS- \*标准。这个体系结构有两种实现，Axis2 Java，顾名思义，用Java编写，Axis2 C，用C语言编写。Axis2 Java通常部署为像Apache Tomcat这样的servlet引擎中的servlet。另一方面，Axis2 C可以部署为Apache HTTP服务器中名为mod\_axis2.so的共享库。与Axis2相比，PHP中的SOAP支持提供了一个单一的体系结构，实现为用C编写的扩展（PHP运行时的内置库）。扩展由核心运行时引擎在PHP进程中通过定义的C接口直接调用。[11]中提供的PHP语言是一个能够缓存中间代码（APC缓存）的解释器，而Java有一个即时编译器。将PHP运行时与HTTP服务器链接的两种主要方法是mod\_php中的共享库方法和FastCGI服务器方法。使用mod\_php，PHP运行时在HTTP服务器进程内运行。使用FastCGI，可以将多个PHP进程作为单独的外部进程生成，并通过定义良好的FastCGI协议与HTTP服务器进行通信。在自动内存管理方面，PHP使用引用计数，而Java使用垃圾收集。因此，PHP的运行时性能已经包含了内存管理的成本，但Java中垃圾收集的一个悲剧性后果是你无法预测它何时会发生。

3.2 XML解析器和处理模型

众所周知，XML处理（尤其是序列化和反序列化）是影响Web服务引擎整体性能的关键因素。Axis2的XML处理模型基于AXIOM（轴对象模型），它为SOAP提供了一个简单的API。AXIOM基于StAX API（Streaming API for XML），Axis2 Java使用Woodstox，它是符合JSR 173标准的StAX解析器实现，而Axis2的C实现，Axis2 C，使用libxml2来处理SOAP消息。此XML解析器与PHP中的SOAP引擎相同。PHP中的SOAP引擎，实现为Cbased扩展，使用libxml2（PHP 5.2.5中的libxml-2.5.4）[11]。LibXML2是一个C库，它提供流式拉解析器和类似SAX的接口以及类似DOM的方法，但PHP中的SOAP扩展仅使用SAX接口而不是流API。[8]中的文章描述了用C编写的Libxml2 XML Parser与其他XML解析器（包括Woodstox [10]（用Java编写的StAX Pull Parser））之间的性能比较。本文的结果表明，C语言中类似SAX的LIBXML2实现在所有基准测试中表现最佳，各种文档大小从小型文档（4.5KB，13.5KB，50KB，95KB）到大型文档（455KB，905KB） ，4000KB）。对于大多数文档大小，它提供的吞吐量是其他XML解析器的三分之一到两倍。创建对象模型（如AXIOM或DOM）的处理速度也在[10]中报告，这表明LIBXML2比所有其他实现快得多。

3.3 编程模型和部署模型

在本节中，我们将比较Web服务引擎的软件生产力，例如可编程性和实现和发布Web服务的人员的易部署性。第一个Axis2 Java具有各种功能，例如处理对象的能力，如普通旧Java对象（POJO），它允许程序员轻松开发和发布他们的Web服务。与Axis1相比，实现Web服务非常简单。如果开发人员拥有POJO，则使用提供的ant脚本自动生成所有必需文件，包括WSDL，服务框架文件，客户端存根文件和AAR（Axis2 Archive）可部署存档文件。但是，在测试阶段，将重复使用此工具来定期修改，发布和调试代码。Axis2 C还具有Axis2 Java提供的大部分功能。但是，非程序员很难调试Web服务实现并从WSDL生成代码，特别是那些没有使用C调试器处理分段错误的人。最后，表2显示了用于通过PHP中的SOAP支持发布报价库Web服务的代码片段。WSDL文件的位置在SoapServer构造的参数中指定。在发布中，您只需在适当的Web目录中找到PHP脚本即可。与Java不同，不需要服务描述文件和编译，也不需要生成存根代码。开发人员可以简单地将处理函数注册到SoapServer。同时，编写客户端代码也很简单。您可以简单地获取以下PHP代码片段所示的结果。 $ client = new SoapClient（'StockQuoteService.wsdl'）; $ quote = $ client-> getStockQuote（$ symbol）;

尽管PHP中可能存在运行时错误，但由于数据是动态类型的，并且在编译时未进行检查，因此程序员可以在不编译的情况下以敏捷方式直接编辑和调试文件。

3.4 功能WS- \*标准支持

如上一节所述，Axis2旨在提供灵活的体系结构，以支持各种WS- \*标准。相比之下，当前PHP发行版提供的标准SOAP扩展支持SOAP 1.1，SOAP 1.2和WSDL 1.1规范的子集。但是，WSO2™的PHP Web服务框架[16]已经开始支持全套WS- \*。

（1）REST支持

Axis2支持SOAP 1.1和SOAP 1.2，以及REST风格的Web服务。单个服务既可以同时作为SOAP样式也可以作为REST样式服务公开。PHP被认为是实现RESTful服务时使用的语言。在PHP中实现此类服务非常简单，只需要很少的额外编码。

（2）异步支持

JAX-WS 2.0的主要贡献是支持异步Web服务。[11]中提供的PHP实现中提供的SOAP扩展目前不支持此功能，因为PHP语言本身不支持线程。可能的替代实现类似于Java 1.4中引入的Java New I / O（NIO），或者扩展PHP以支持线程。

4 定量比较

本节描述了3个SOAP引擎之间的定量比较：PHP5中的SOAP引擎，Axis2 Java和Axis2 C.特别是对于PHP，我们还比较了两个Web服务器，Lighttpd [9]和Apache 2。

4.1 测试方法

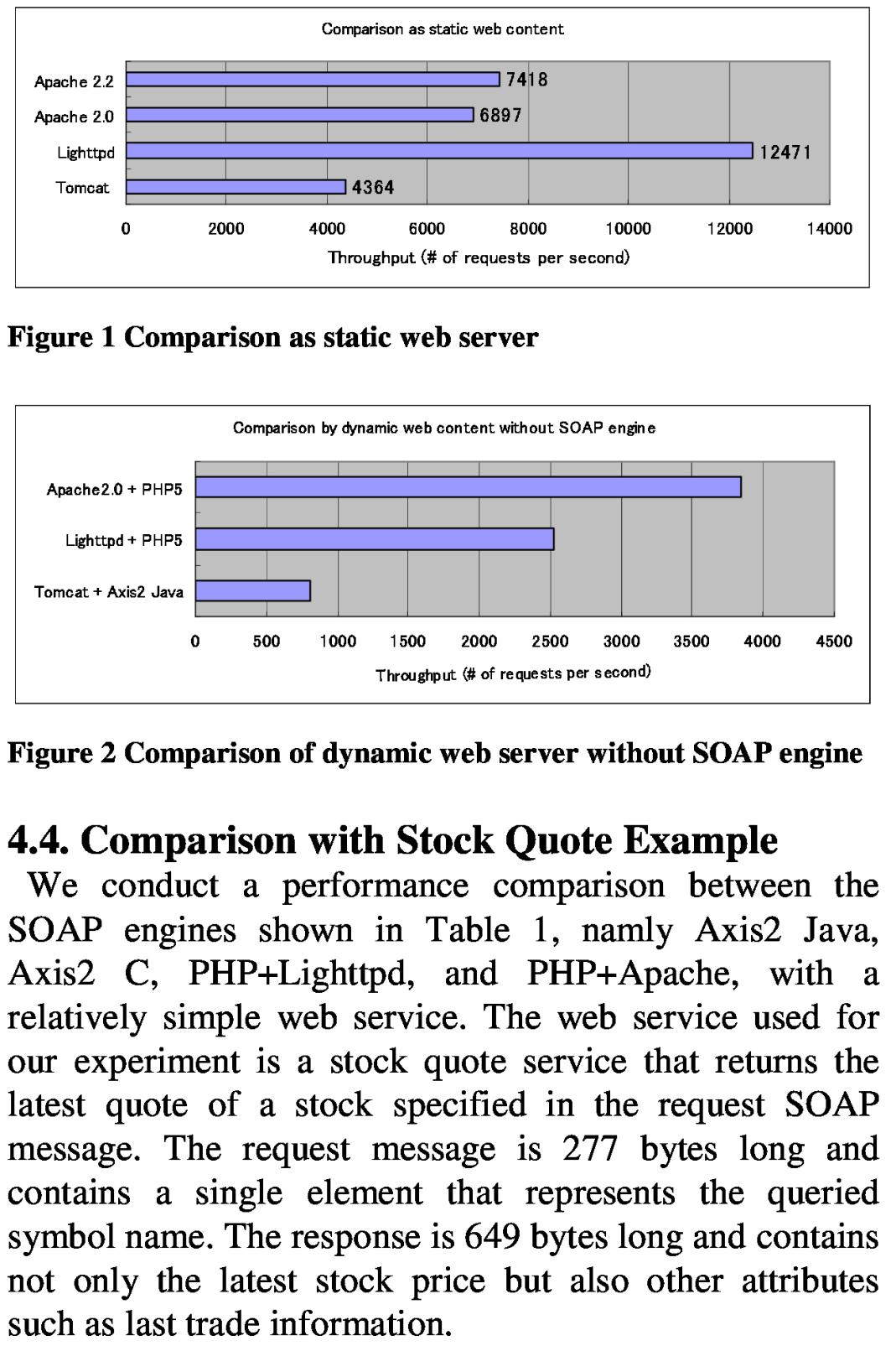
服务器端的整体架构遵循三层架构，包括Web服务器，SOAP引擎（Web服务引擎）和Web服务实现本身。我们准备了表1中所示的4种待测试变体。我们检查了当今可用的众多Web服务器和连接器，但我们尽可能地关注最流行的配置。对于PHP，我们选择了两个代表性的HTTP服务器：最广泛使用的Apache HTTP服务器（版本2.0.63）和Lighttpd（1.4.19）[10]，它已经得到社区的广泛认可（包括用于商业网站）因为它的高性能。表1中显示的连接器（通常称为SAPI）表示将PHP运行时连接到HTTP服务器的方法。在Apache中，我们选择了mod\_php方法，其中PHP运行时及其SOAP引擎在与Apache HTTP服务器相同的进程内运行。对于Lighttpd，我们选择FastCGI作为连接器，其中多个PHP进程作为外部进程运行，并通过定义良好的FastCGI协议与Apache HTTP服务器通信。我们将FastCGI进程数配置为16。接下来，对于Axis2 Java，我们使用Apache Tomcat（5.5.26）作为servlet引擎，其中Axis2 Java作为Java servlet运行。对于Axis2 C，我们将Apache 2.2.26用于HTTP服务器，其中Axis2 C引擎在与Apache http服务器相同的进程内作为mod\_axis2运行。

4.2 实验环境

我们使用IBM IntelliStation M Pro和3.4 GHz Xeon™单处理器以及运行Fedora Core 7（Linux Kernel 2.6.21）的3GB RAM作为服务器。Tomcat配置为使用IBM Java虚拟机：J9 VM 1.5.0 Build 2.3。我们使用IBM IntelliStation M Pro和2.4 GHz Xeon™双核处理器以及运行Fedora Core 7的1GB RAM作为客户端。服务器和客户端通过千兆以太网连接，网络延迟为0.14毫秒。Apache Bench 2.0.40用于测量吞吐量。Apache Bench配置为使用100个并发连接，并且设置了HTTP keep-alive选项。

4.3 比较没有SOAP引擎的

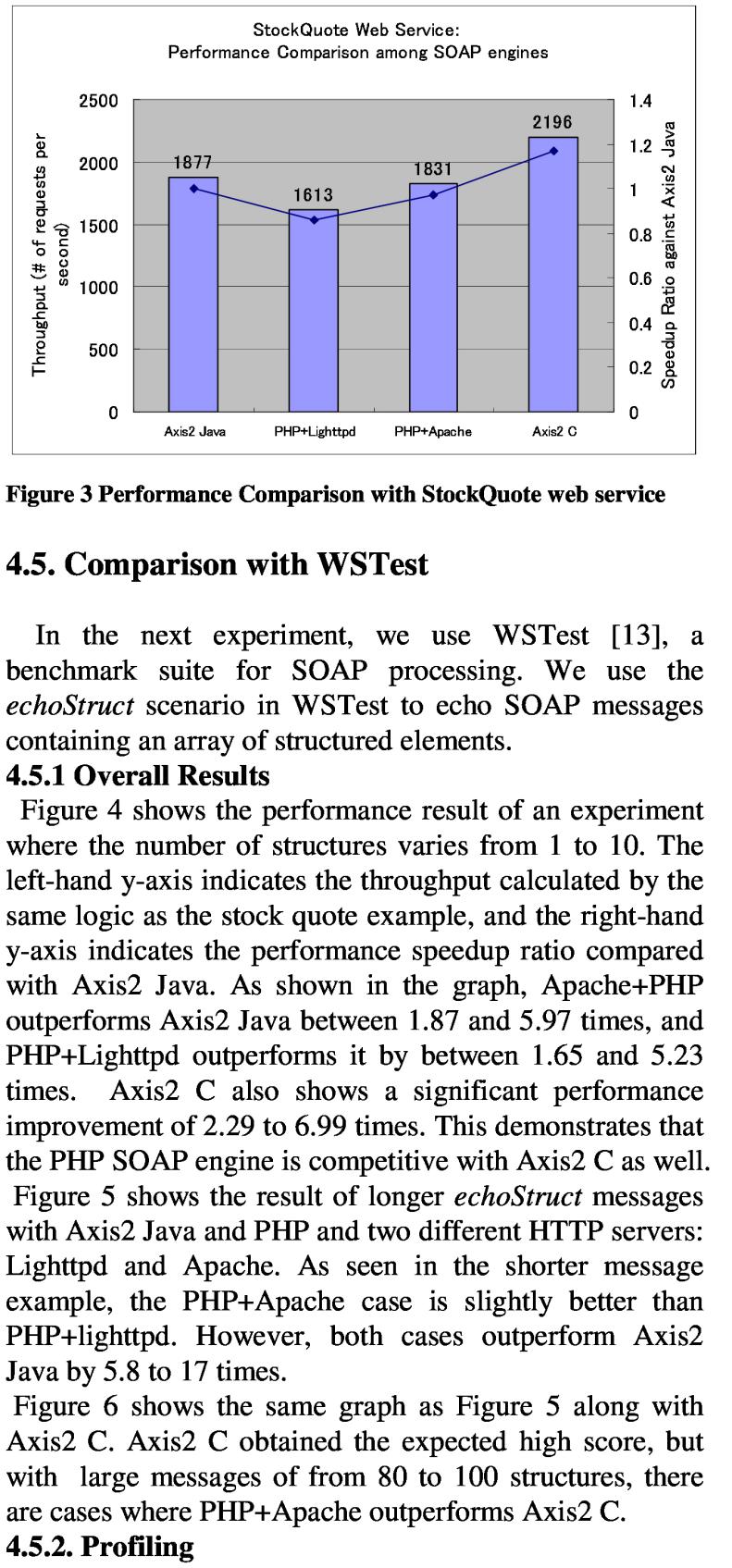
Web服务器在本实验中，为了考虑没有SOAP引擎的每个Web服务器的性能特征，我们进行了两次实验。我们比较了服务静态HTML文件时的Web服务器性能，该文件包含股票报价Web服务使用的相同SOAP消息，如下一节所述。结果如图1所示.Lighttpd表现最佳，因为Lighttpd旨在提供静态文件时提供最佳性能，例如，使用sendfile系统调用以避免用户和内核空间之间的冗余复制。Apache 2.2比Apache 2.0好大约7％，比Tomcat 5.5好大约70％。图2演示了在没有SOAP引擎的情况下在每个配置中运行动态Web应用程序的性能比较。此实验旨在显示没有SOAP引擎的每个配置的初始开销。本实验中使用的Web应用程序使用与Apache Bench客户端下一节中通过HTTP POST方法使用的Stock Quote Web服务相同的请求消息，然后返回与Web服务相同的响应消息。编写每个运行时配置的动态Web应用程序，以便可以在Web中提供容器，例如Tomcat的JSP脚本和Lighttpd和Apache的PHP脚本。



4.4.与股票报价实例的比较

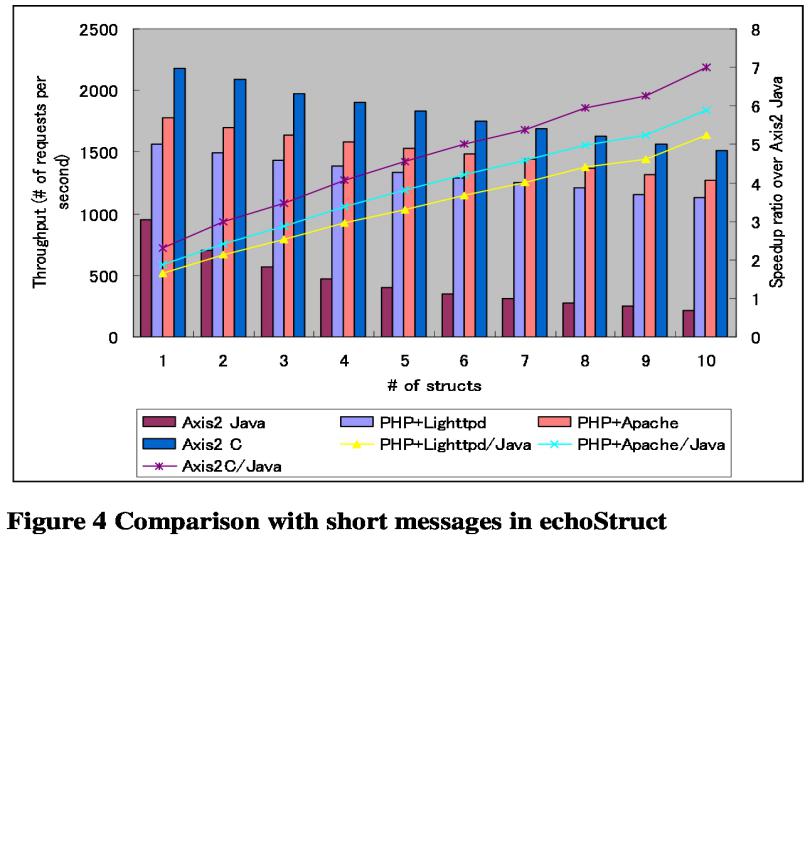
我们使用一个相对简单的Web服务，在表1、名称为Axis 2 Java、Axis 2 C、PHP+Lightttpd和PHP+Apache中的SOAP引擎之间进行性能比较。用于我们实验的Web服务是一个股票报价服务，它返回请求SOAPMessage中指定的股票的最低价。请求消息的长度为277字节，并包含一个。表示查询符号名称的单个元素。响应长度为649字节，不仅包含最新的股票价格，而且还包含其他属性，如上一次交易信息。

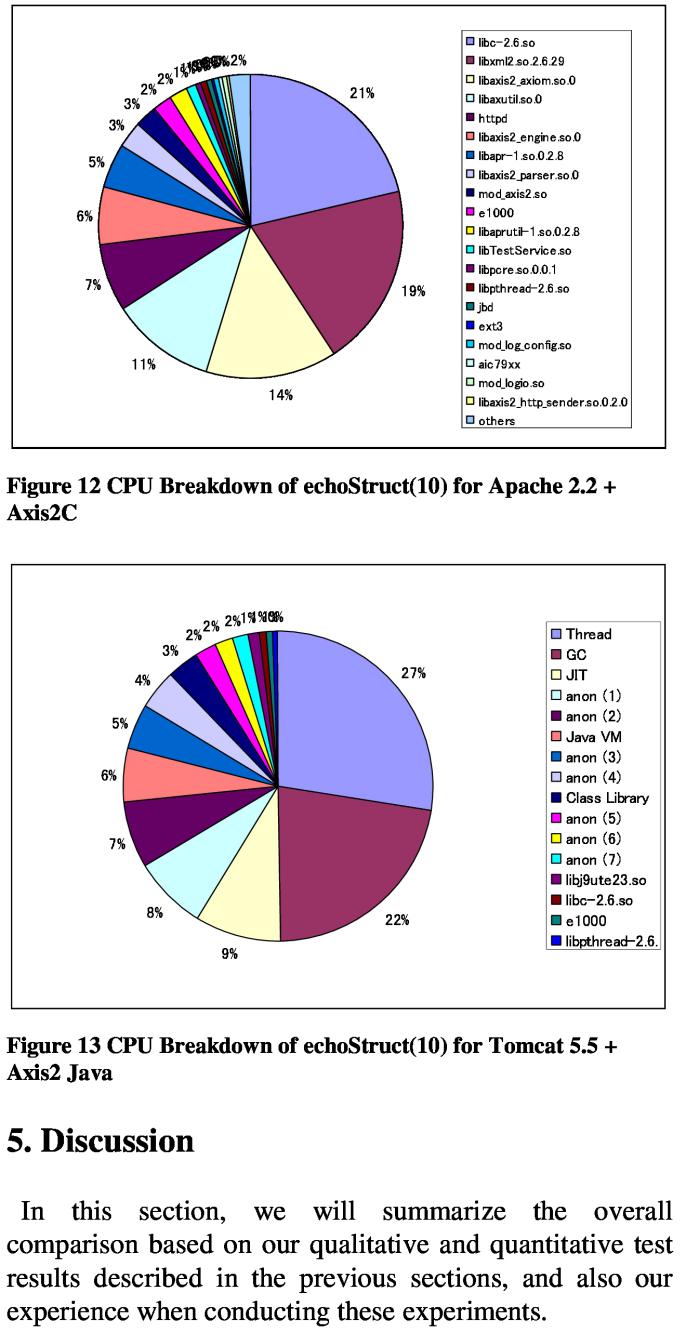
给定股票报价服务的WSDL文件，为Axis2生成每个SOAP引擎的服务框架。由于PHP目前不包含WSDL生成机制，因此我们从头开始实现它。表2显示了基于PHP的Web服务的所有代码，表3显示了基于Java的Web服务的代码片段以及从WSDL生成的许多其他代码。结果如图3所示。左侧y轴表示吞吐量，该吞吐量对应于Apache Bench计算的一秒钟内处理的请求数。右侧y轴显示了针对Axis2 Java的每个SOAP引擎的性能改进率。如图所示，PHP + Lighttpd性能是Axis2 Java的86％，PHP + Apache性能是Axis2 Java的97％，而Axis2 C比Axis2 Java高出25％。由于SOAP引擎的性能主要取决于序列化和反序列化的成本，而且由于这种类型的Web服务不会在这些操作上花费那么多时间，因此这个结果清楚地表明性能差异来自其他处理。第一个和第二个PHP实验之间的关键区别在于PHP运行时是否调用用C编写的SOAP引擎扩展。这个结果表明，Apache 2.0和PHP5的混合效率低于Lighttpd和PHP5的混合效果。在Lighttpd配置中，16个PHP进程作为单独的进程运行，而在Apache配置中，PHP引擎嵌入在数百个HTTP进程中。



为了更好地理解性能特征，我们在10结构消息echoStruct场景中对每个SOAP引擎进行了分析。图9中的CPU故障表明，使用PHP + Lighttpd，在PHP引擎中花费了84％的CPU，而HTTP服务器只消耗了13％的CPU。为了进一步研究PHP过程中的CPU使用情况，图10显示XML处理消耗了12％的CPU，而C库消耗了21％的CPU。c库中函数的进一步细分表明许多函数与内存管理有关。图11显示了PHP + Apache的CPU故障。由于mod\_php PHP引擎在Apache HTTP服务器进程内运行，因此该图包含两个组件。图12显示了Axis2C的CPU故障，图13显示了Axis2 Java的CPU故障。如图13所示，垃圾收集占CPU使用率的22％。由于PHP内存管理使用引用计数，因此内存管理成本包含在运行时引擎CPU使用中。

内存占用除了总吞吐量之外，我们还需要考虑内存使用情况。PHP + Lighttpd案例中PHP的内存占用量与生成的PHP进程数成正比。一个PHP进程消耗大约5.5 MB，16个进程消耗大约90.4 MB。另一方面，在Axis2 Java案例中，包括Web服务器，SOAP引擎和Web服务实现在内的所有组件都在一个具有多个线程的进程中运行，这些线程占用62.1 MB。使用FastCGI的优点允许通过FastCGI协议与http服务器通信，在不同的节点上生成PHP进程，但如果我们只使用单个节点，Axis2 Java可以提高内存效率。



在下一个实验中，我们使用WSTestI 13，用于SOAP处理的标记套件。我们使用WSTest中的echo Struct场景来回显包含一个结构化元素数组的SOAP消息。 后果，结果（ result的名词复数 ）图4显示了在结构数量从1到10之间变化的实验的性能结果。左侧y轴表示通过与库存报价相同的逻辑计算的吞吐量。 例如，与Axis 2 Java相比，右边的便捷轴表示性能加速比。如图所示，ApachePHP优于Axis2Java 1.87至5.97次，PHP Lightttpd 比它高出1.65和5.23倍。Axis2C还显示出显著的性能改善2.29到6.99倍。这说明PHPSOAP引擎与Axis2C竞争为W 图5显示了Axis 2、Java和PHP以及两种不同的HTTP服务器的长回声结构消息的结果：Lightttpd和Apache。从较短的消息示例中可以看到，PHPApache的情况是sli 比PHP更好。然而，这两种情况的性能都比Axis2Java高5.8至17倍。图6显示了与图5和Axis 2 C相同的图形。Axis2C获得了预期的高分，但是使用了80到100个结构的大消息，在一些情况下PHP Apache的性能优于 轴2

5 讨论

在这里面。在前几节所描述的定性和定量测试结果的基础上，我们将总结总体上的比较，以及在进行这些实验时的有效性。

PHP与Java一起使用PHP Web服务堆栈的良好性能的主要原因之一是使用最快的xml解析器libxml2，它是用C语言编写的。关于与C模块的接口，PHP的扩展API是为高效率而设计的（但不是为了健壮性或可移植性），而Java Native Interface（J NI）则是为了健壮性和可移植性而非效率而设计的。如果我们在Java WS堆栈中使用libxml，由于从C到Java的频繁昂贵的调用，它将导致显着的开销。还请注意，用Java编写的xml解析器比C语言慢，因为Java对字符串的处理效率低，而且禁止（可能是类型不安全的）指针算法。但是，为了获得更高的系统稳定性和一些专业程序员的工作效率，Java肯定处于有利地位。

PHP与C股票报价Web服务实验的性能表明，与其他SOAP引擎相比，Axis2 C表现良好。Axis2 C是用C实现的，但是这个性能得分可能是预期的，但是因为Axis2的设计是高度模块化的。目前尚不清楚Axis2 C是否仍然优于PHP SOAP引擎，后者采用单片方法设计，并且也用C语言编写。但是，从生产力，可编程性，易部署性和可调试性的角度来看，它远远落后于其他引擎。特别是在易于调试方面，调试为Axis2 C编写的Web服务并不简单，因为从分段错误中产生的核心转储与原始Web服务定义之间的关系并不明显。

PHP作为SOAP引擎考虑到最近流行的编程语言的趋势，PHP满足了生产力和性能的基本语言要求。尽管Java可以满足这些要求，但由于PHP用于各种Web应用程序，而SOAP和WS- \*尚未成功用于公共空间（某些强大的企业应用程序除外），我们声称PHP处于有利地位。

6.相关工作

Ceccet，et al。比较了EJB，Java Servlet和PHP [2] [15]。他们还报告说PHP语言表现最好。然而，本文中进行的实验使用传统的Web应用程序工作负载（如TPC-W）来对电子商务应用程序进行基准测试。因此，此性能结果不适用于需要大量XML处理的Web服务引擎的性能比较。如3.2节所述，诸如[8]之类的文章报告了各种XML解析器的性能比较，但它们只关注XML处理，因此它不包括端到端场景的性能比较。许多IT供应商提供高性能Web服务引擎，例如IBM WebSphere™Application Server，以及性能基准报告。我们的论文侧重于不同编程语言的性能含义。一些努力[4] [17]致力于XML处理的性能优化，尤其关注基于Java的Web服务引擎的优化。

7.结论和未来工作

本文描述了PHP中实现的Web服务引擎与Java和C中实现的Web服务引擎的比较。鉴于不同的体系结构和不同的编程语言，实验结果表明，与基于Java和C的实现相比，PHP具有相当高的性能，同时为用户提供了高软件生产力。我们的贡献将为普通程序员提供更多的认识，即除了当前流行的REST风格的Web服务之外，PHP是发布基于SOAP / WS- \*的Web服务的可行选择。可以进一步探讨几个主题。首先，我们可以扩展各种Web服务，例如WSTest中的其他场景或更多涉及业务逻辑和数据库访问的真实场景。此外，我们可以探索更广泛的服务器配置。在本文中，我们仅使用Axis2 Java及其默认数据绑定ADB，但Java中的其他Web服务实现可能与本文中报告的结果不同。我们还可以比较不同的动态脚本语言，如Ruby和Python。此外，我们可以在RESTful服务方面探索性能，而不是专注于基于SOAP的Web服务服务。在通过HTTP Get操作访问Web服务之后，XML或JSON被用作数据格式，但是我们可以比较PHP和Axis2，因为PHP现在是RESTful Web服务的流行，而Axis2并没有真正使用这种方式，即使Axis2同时提供RESTful和SOAP功能，用于相同的Web服务实现，无需额外的编程工作。

参考

[1] Alex Ng，陈世平，Paul Greenfield，“对当代商业SOAP实施的评估”，AWSA（澳大利亚软件与系统架构研讨会）2004

[2] Emmanuel Cecchet，Anupam Chandra，et.al，“用于生成动态Web内容的中间件架构的性能比较”，Middleware 2003

[3] Subramanyam S.，et al。，“Designing a web services

benchmark”, In ICWS 03. International Conference on Web Services, Las Vegas, USA.

1. Toyotaro Suzumura, Satoshi Makino and Naohiko Uramoto, “Optimizing Differential XML Processing by Leveraging Schema and Statistics”, ICSOC 2006, Springer, Lecture Notes in Computer Science, pp. 264p276

[5] Srinath Perasa, Chathura Herath, Jaliya Ekanayake, et. Al. “Axis2, Middleware for Next Generation Web Services”, ICWS 2006, URL: <http://apache.axis.org/axis2/>Performance Comparison of Web Service Engines in PHP, Java, and C

Toyotaro Suzumura, Scott Trent, Michiaki Tatsubori, Akihiko Tozawa and Tamiya Onodera

Tokyo Research Laboratory, IBM Research,

1623-14 Shimotsurusma, Yamato-shi, Kanagawa-ken 242-8502, Japan

{toyo, trent, mich, [atozawa,tonodera}@jp.ibm.com](mailto:atozawa,tonodera}@jp.ibm.com)

**Abstract：**PHP is well known as a programming language in the Web 2.0 era enabling agile server-side software development. It has officially supported SOAP messaging since version 5 through a C-based built-in library. In this paper we perform a thorough study of the capability of PHP as a web service engine in both qualitative and quantitative aspects while comparing it with other web service engines implemented in Java and C. We used Axis2 for this purpose as it is an open source web service engine whose implementation is available both in Java and C. We report that PHP as a web service engine performs competitively with Axis2 Java for web services involving small payloads, and greatly outperforms it for larger payloads by 5-17 times. As the authors expected, Axis2 C performs best, but the experimental results demonstrate that PHP performance is closer to Axis2 C with larger payloads. This performance difference comes from the fact that the SOAP engine within the PHP runtime is implemented in C with a monolithic architecture, whereas Axis2 uses a more modular architecture for the flexible insertation of handlers for an assorted set of WS-\* standards, and also that Axis2 uses a different data binding mechanism known as ADB (Axis2 Data binding). This paper is the first attempt to compareweb services engines implemented in PHP, Java and C, and the authors believe that this boosts the development of SOAP-based Web services in PHP by letting people know its decent performance score and high productivity Characteristics

**1 Introduction**

Web service technology, especially that based on WS-\*, plays a key role for the realization of Service Oriented Architecture. However, it is difficult to argue that web services are as widely used as assumed when the technology was first emerged. Potential reasons include the performance problem of XML processing, too many complicated WS-\* standards, and so forth. Recent technologies seen in the Web 2.0 era have arisen through the community driven approach, where average programmers are playing the key role as they use it. From this viewpoint, one of the biggest reasons for the above unpopularity is the software productivity and performance disadvantages.

The PHP language is widely used as a popular server side language [7]. A great number of open source software and company’s web sites use PHP since it can enable high software productivity. Moreover, PHP officially offered SOAP support in version 5. Thus, thissupport helped boost the wide spread use of web services based on SOAP/WS-\*.In this paper, we describe a thorough comparative study of PHP and Axis2, the major open source SOAP engine implementation from a qualitative and quantitative perspective. Please note that we do not argue that PHP is better than other programming languages which provide web services support, but rather we focus on a neutral comparison.

The structure of this paper is as follows. Section 2 describes an overview of Axis2 and the SOAP extension in PHP. Section 3 describes the qualitative comparison in erms of programming model, deployment model, WS-\* standard support, and other functionalities. Next, Section 4 describes the quantitative comparison using a benchmark called StockQuote web service and WSTest, and in Section5, we summarize the comparison in web services runtime between PHP, Java, and C. We conclude this paper by introducing related works in Section 6, along  
with conclusions and future works in Section 7.

**2 Overview of Web Services Runtimes**

The main objective of this paper is to demonstrate how the PHP SOAP engine is different from web services engines implemented in various programming languages. Today there are many commercial products and open source software that provide a web services engine, but in this paper we use two engines available from the Axis2 project, a Java-based implementation and a C-based implementation. As described detail later in this paper, in terms of software productivity and ease of deployment, the C-based implementation is not more straightforward than runtimes based on languages such as Java and PHP. However, we use it mainly for performance comparison to understand the best possible performance. Next we willprovide an overview of PHP and its web services runtime, and Axis2.

**2.1 The PHP Web Services Runtime**

PHP has officially supported SOAP since version 5 with a C-based extension. SOAP engines for previous versions were available from PEAR (PHP Extension and Application Repository) implemented in PHP or PECL (The PHP Extension Community Library) implemented in C. Sometimes PHP is stereotyped as moderately performing language runtime since it is interpreted. However, PHP can cache intermediate opcode in Alternative PHP Cache （ APC), and moreover the standard set of “extension” libraries is mostly implemented in C. These features make PHP different from other dynamic scripting languages such as Ruby and Python.

**2.2. Apache Axis2**

Axis2 [5] is an open source project to provide a web service engine as a follow-on project to Axis1. The architecture of Axis2 is rebuilt from scratch to provide more performance-awareness and flexibility to support an assortment of WS-\* standards. There are two implementations, Apache Axis2/Java written in Java, and Apache Axis2/C written in C. Many features distinguish Axis2 from Axis1 such as the support of various WS-\* standards and asynchronous web services. Another notable feature of Axis2 is its REST support. When the REST option is enabled, a single operation returns either of an XML message or a SOAP envelope. This feature is important as it alleviates the burden of programmers who wish to publish their web services both in SOAP and REST style.

**3 Qualitative Comparison**

This section presents a comparative study between web services support provided by PHP and Axis2 from variety of perspectives including software architecture, the XML processing model, WS-\* standard support, the programming model and deployment model for server providers, the support for RESTful service, and other functionalities.

**3.1. Software Architecture**

The architecture of Axis2 is based on the concept that the architecture should be more flexible and configurable in comparison to Axis version 1. Axis2 architecture has a modular architecture that enables the installation of additional functionality using modules to support an assortment of WS-\* standards. There are two implementations of this architecture, Axis2 Java which, as the name implies, is written in Java, and Axis2 C, which similarily is written in C. Axis2 Java is usually deployed as a servlet in a servlet engine like Apache Tomcat. On the other hand, Axis2 C can be deployed as a shared library called mod\_axis2.so in Apache HTTP server.

Compared to Axis2, SOAP support in PHP is provided with a monolithic architecture, implemented as an extension (built-in library in PHP runtime) written in C. The extension is directly invoked by the core runtime engine in a PHP process via a defined C interface. The PHP language available from [11] is an interpreter capable of caching intermediary code (APC cache),whereas Java has a Just-in-time compiler. The two main methods to link a PHP runtime with an HTTP server are the shared library approach in mod\_php and the FastCGI server approach. With the mod\_php, a PHP runtime runs inside an HTTP server process. With FastCGI, multiple PHP processes can be spawned off as separate external processes and communicate with an HTTP server via a well-defined FastCGI protocol.  
 In terms of automatic memory management, PHP uses reference counting whereas Java uses garbage collection. So the runtime performance of PHP already includes the cost of memory management, but a tragic consequence of garbage collection in Java is that you can not predict when it will take place.

**3.2. XML Parser and Processing Model**

It is well known that XML processing, especially serialization and de-serialization, is a key factor that affects the overall performance of a web services engine. The XML processing model of Axis2 is based on AXIOM (Axis Object Model) which provides a simple API for SOAP. AXIOM is based on the StAX API (Streaming API for XML), and Axis2 Java uses Woodstox, which is a JSR 173 conforming StAX parser implementation, and the C implementation of Axis2, Axis2 C, uses libxml2 to process SOAP messages. This XML parser is the same as the SOAP engine in PHP.  
 The SOAP engine in PHP, which is implemented as a Cbased extension, uses libxml2 (libxml-2.5.4 in PHP 5.2.5)[11]. LibXML2 is a C library that provides a streaming pull-parser and a SAX-like interface as well as a DOMlike approach, but the SOAP extension in PHP only uses a SAX interface rather than the streaming API. The article in [8] describes the performance comparison between the Libxml2 XML Parser written in C and other XML parsers including Woodstox [10] (StAX Pull Parser written in Java). The results of this article demonstrate that the SAX-like implementation of LIBXML2 in C performs best in all benchmarks with various document sizes ranging from small sized documents (4.5KB, 13.5KB, 50KB, 95KB) to big-sized documents (455KB, 905KB,4000KB). For most document sizes it provided one-third to twice as much throughput as other XML parsers. The processing speed of creating object models such as AXIOM or DOM is also reported in [10], which shows that LIBXML2 is much faster than all otherimplementations

**3.3. Programming Model and Deployment Model**

In this section we compare the web service engines in terms of software productivity such as programmability and ease of deployment for those who implement and publish their web services.

First Axis2 Java has various features such as the ability to handle objects as Plain Old Java Objects (POJO) which allow programmers to easily develop and publish their web services. In contrast with Axis1, it is quite straightforward to implement a web service. If a developer has a POJO then all required files including WSDL, service skeleton files, client stub files and the AAR (Axis2 Archive) deployable archive file are automatically generated with a provided ant script . However during the testing phase, this tool will be used repeatedly to periodically modify, publish, and debug the code. Axis2 C has also most of the features provided by Axis2 Java. However it is hard for non-programmers to debug the web service implementation and generated code from WSDL, especially those who are not experienced in handling segmentation faults with the C debugger. Lastly Table 2 shows a code snippet for publishing a quote stock web service through SOAP support in PHP. The location of the WSDL file is specified in the argument of the SoapServer construct. In publishing, you can simply locate the PHP script in an appropriate web directory. Unlike Java, no service description file and compilation are required, nor is it necessary to generate stub code. A developer can simply register a handler function to a SoapServer. Meanwhile, it is also straightforward to write client side code. You can simply obtain the result as demonstrated by the following PHP code fragment.  
$client = new SoapClient(‘StockQuoteService.wsdl’);  
$quote = $client->getStockQuote ($symbol);  
Although runtime errors are possible in PHP, since data is dynamically typed and not checked at compile time, programmers can directly edit and debug the file in an agile fashion without compiling

**3.4. Functionalities**

WS-\* Standard Support  
As described in the previous section, Axis2 is designed to have a flexible architecture to support an assortment of WS-\* standards. In constrast, the standard SOAP extension provided by the current PHP distribution supports subsets of SOAP 1.1, SOAP 1.2, and WSDL 1.1 specifications. However, the Web Services Framework for PHP [16] by WSO2 ™ has started to support full set of WS-\*. REST Support Axis2 supports SOAP 1.1 and SOAP 1.2, as well as the REST style of web services. A single service could be exposed both as a SOAP style as well as a REST style service simultaneously. PHP is considered to be the language to use when implementing RESTful services. It is quite straightforward to implement such services in PHP with little additional coding.  
Asynchrony Support  
The major contribution of JAX-WS 2.0 is to support asynchronous web services. The SOAP extension provided in the PHP implementation available in [11] currently does not support this feature because the PHP language itself does not support threading. Possible alternative implementations would be similar to Java New I/O (NIO) introduced in Java 1.4 or to extend PHP to support threading.

**4 Quantitative Comparison**

This section describes a quantitative comparison among 3 SOAP engines: the SOAP engine in PHP5, Axis2 Java,and Axis2 C. Especially for PHP, we also compare two web servers, Lighttpd [9] and Apache 2

**4.1. Testing Methodology**

The overall architecture on the server side follows a three-tier architecture comprised of a web server, a SOAP engine (web services engine), and the web service implementation itself. We prepared 4 variations to be tested shown in Table 1. We examined numerous web servers and connectors available today, but we focused on the most popular configuration as much as possible.

For PHP, we selected two representative HTTP servers:the most widely used Apache HTTP server (version 2.0.63), and Lighttpd (1.4.19) [10] which has achievedwide acceptance from the community (including use for commercial web sites) because of its high performance. Aconnector (often called SAPI) shown in Table 1 indicates a method of connecting the PHP runtime with an HTTPserver. In Apache, we selected the mod\_php approach where the PHP runtime and also its SOAP engine runs inside the same process as an Apache HTTP server. For Lighttpd, we selected FastCGI as a connector with which multiple PHP processes which run as external processes and communicate with an Apache HTTP server via the well-defined FastCGI protocol. We configured the number of FastCGI processes to 16.

Next, for Axis2 Java, we used Apache Tomcat (5.5.26) as the servlet engine where Axis2 Java runs as a Javaservlet. For Axis2 C, we use Apache 2.2.26 for the HTTP server where Axis2 C engine runs as mod\_axis2 inside thesame process as the Apache http server.

**4.2. Experimental Environment**

We used IBM IntelliStation M Pro with a 3.4 GHz Xeon™ uniprocessor and 3GB RAM running Fedora Core 7(Linux Kernel 2.6.21) as the server. Tomcat was configured to use the IBM Java Virtual Machine: J9 VM 1.5.0 Build 2.3. We used an IBM IntelliStation M Prowith a 2.4 GHz Xeon ™ dual-core processor and 1GB RAM running Fedora Core 7 as the client. The server and client are connected via a Gigabit Ethernet with a 0.14 millisecond network latency. Apache Bench 2.0.40 was used for to measure the throughput. Apache Bench was configured to use 100 concurrent connections and the HTTP keep-alive option was set on.

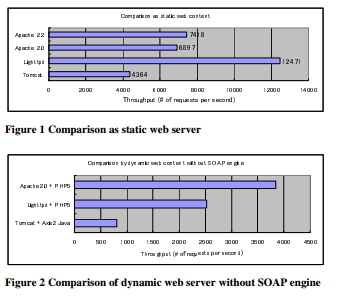
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Server Configuration Label | HTTP Server | Connector | SOAP Engine | Web Services |
| Axis2 Java | Tomcat 5.5.26 (Java) | None (thread) | Axis2 Java 1.3 (Java) | Java |
| Axis2 C | Apache 2.2.8 (C) | mod\_axis2 | Axis2 C 1.3.0 (C) | C |
| PHP+Lighttpd | Lighttpd 1.4.19 (C) | FastCGI | PHP5 5.2.5 (C) | PHP |
| PHP+Apache | Apache 2.0.63 | mod\_php | PHP5 5.2.5 (C) | PHP |

Table 1 Server configuration for target SOAP engines

**4.3. Comparing Web Servers without SOAP Engines**

In this experiment, to consider the performance characteristics of each web server without SOAP engine, we performed two experiments. We compared the web server performance when serving a static HTML file thatincludes the same SOAP message used by stock quote web services as described in the next section. The result isshown in Figure 1. Lighttpd performs the best since Lighttpd is designed to provide optimal performancewhen serving static files by, for example, using sendfile system call to avoid redundant copying between user andkernel space. Apache 2.2 performs about 7% better than Apache 2.0 and about 70% better than Tomcat 5.5 .

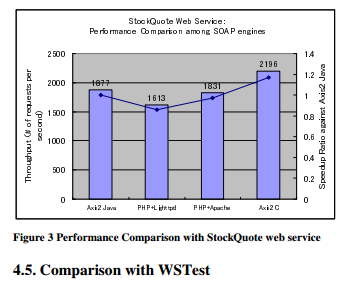
Figure 2 demonstrates the performance comparison of running a dynamic web application in each configuration without a SOAP engine. This experiment is intended to show the initial overhead for each configuration without a SOAP engine. The web application used in this experiment consumes the same request message as the Stock Quote web service used in the next section from the Apache Bench client via an HTTP POST method, and then returns the same response message as the web service. The dynamic web application for each runtime configuration is written so it can be served in a web container, such as JSP script for Tomcat and PHP script for Lighttpd and Apache.



**4.4. Comparison with Stock Quote Example**

We conduct a performance comparison between the SOAP engines shown in Table 1, namly Axis2 Java,Axis2 C, PHP+Lighttpd, and PHP+Apache, with arelatively simple web service. The web service used for our experiment is a stock quote service that returns the latest quote of a stock specified in the request SOAP message. The request message is 277 bytes long and contains a single element that represents the queried symbol name. The response is 649 bytes long and contains not only the latest stock price but also other attributes such as last trade information. Given the WSDL file for a stock quote service, a service skeleton for each SOAP engine is generated for Axis2.Since PHP currently does not include a WSDL generation mechanism, we implemented it from scratch. Table 2 shows all the code for PHP-based web service, and Table 3 shows the code snippet for Java-based web service and many other codes generated from WSDL exist.

The results are illustrated in Figure 3. The left y-axis indicates a throughput which corresponds to the number of requests processed in one second calculated by ApacheBench. The right y-axis shows the performance improvement ratio for each SOAP engine against Axis2 Java. As shown in the graph, PHP+Lighttpd performance is 86% of Axis2 Java, PHP+Apache performance is 97% of Axis2 Java, and Axis2 C outperforms Axis2 Java by 25%.Since the performance of a SOAP engine is mostly determined by the cost of serialization and de-serialization,and also since this type of web service does not spend that much time on these operations, this result clearly shows that the performance difference comes from other processing. The key difference between the first and the second PHP experiments is whether or not the PHP runtime invokes the SOAP engine extension written in C. This result demonstrates the mix of Apache 2.0 and PHP5 is less efficient than the mix of Lighttpd and PHP5. In theLighttpd configuration,



16 PHP processes run as separate process, whereas in the Apache configuration, the PHPengine is embedded in hundreds of HTTP processes.

**4.5. Comparison with WSTest**

In the next experiment, we use WSTest [13], a benchmark suite for SOAP processing. We use the echoStruct scenario in WSTest to echo SOAP messages containing an array of structured elements

**4.5.1 Overall Results**

Figure 4 shows the performance result of an experiment where the number of structures varies from 1 to 10. The left-hand y-axis indicates the throughput calculated by the same logic as the stock quote example, and the right-handy-axis indicates the performance speedup ratio compared with Axis2 Java. As shown in the graph, Apache+PHPoutperforms Axis2 Java between 1.87 and 5.97 times, and PHP+Lighttpd outperforms it by between 1.65 and 5.23times. Axis2 C also shows a significant performance improvement of 2.29 to 6.99 times. This demonstrates thatthe PHP SOAP engine is competitive with Axis2 C as well.Figure 5 shows the result of longer echoStruct messages with Axis2 Java and PHP and two different HTTP servers: Lighttpd and Apache. As seen in the shorter message example, the PHP+Apache case is slightly better than PHP+lighttpd. However, both cases outperform Axis2 Java by 5.8 to 17 times.

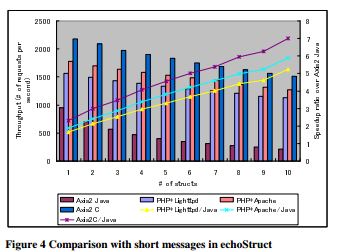
Figure 6 shows the same graph as Figure 5 along with Axis2 C. Axis2 C obtained the expected high score, but with large messages of from 80 to 100 structures, there are cases where PHP+Apache outperforms Axis2 C.

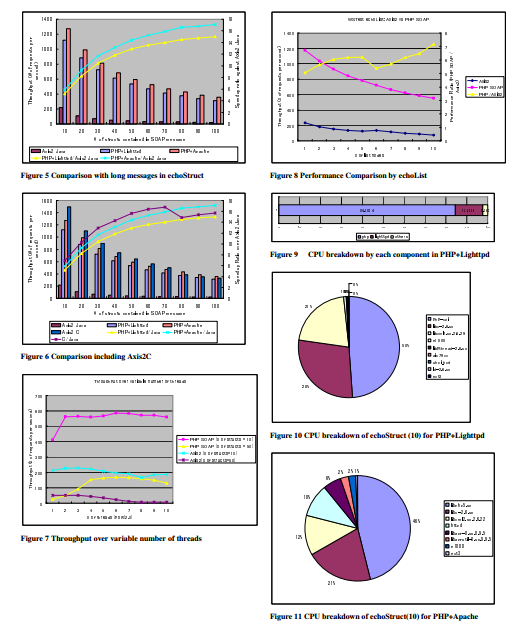
**4.5.2. Profiling**

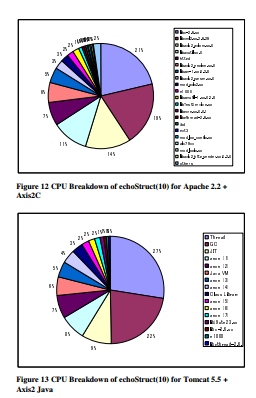
To better understand the performance characteristics, we conducted profiling for each SOAP engine in the 10 structure message echoStruct scenario. The CPU breakdown in Figure 9 shows that with PHP+Lighttpd  
84% of CPU is spent in the PHP engine and only 13 % is consumed by the HTTP server. To further investigate the CPU usage within the PHP process, Figure 10 shows that XML processing consumed 12% of the CPU., and the C library consumed 21% of the CPU. A further breakdown of functions within the c library shows that many of the functions are related to memory management. Figure 11shows the CPU breakdown of PHP+Apache. Since mod\_php PHP engine runs inside Apache HTTP server process, this graph includes both components. Figure 12 shows the CPU breakdown of Axis2C, and Figure 13 that of Axis2 Java. As shown in Figure 13, garbage collection accounts for 22% of CPU usage. Since PHP memory management uses reference counting, this memory management cost is included with the runtime engine CPU usage.

**4.5.3. Memory Footprint**

In addition to the total throughput, we also need to consider memory usage. The memory footprint of PHP in the PHP+Lighttpd case is proportional to the number of PHP processes which are spawned. One PHP process consumes approximately 5.5 MB, and 16 processes consume around 90.4 MB. On the other hand, in the Axis2 Java case, all components including the web server, SOAP engine, and web service implementation, run in a single process with multiple threads, which consumes 62.1 MB.The virtue of using FastCGI allows PHP processes to be spawned on different nodes by communicating with the http server via FastCGI protocol, but if we use only a single node, Axis2 Java can be more memory-efficient.







**5 Discussion**

In this section, we will summarize the overall comparison based on our qualitative and quantitative test results described in the previous sections, and also our experience when conducting these experiments..  
PHP vs. Java

One of the main reasons the good performance seen with PHP web services stack comes from the use of the fastest xml parser, libxml2, which is written in C. Regarding interfacing with C modules, PHP's extension  
API is designed for high efficiency (but not for robustness or portability), while the Java Native Interface (J NI) is designed for robustness and portability rather than for efficiency. If we used libxml in the Java WS stack, it would cause a significant overhead due to frequent expensive call-ins from C to Java. Please also note that  
the xml parser written in java is slower than in C because of Java's inefficient handling of strings and of disallowing (potentially type-unsafe) pointer arithmetic. However, for more system stability and productivity for some expert programmer, Java is surely in a good position.  
PHP vs. C  
The performance of the stock quote web service experiment demonstrates that Axis2 C performs well in  
comparison to other SOAP engines. Axis2 C is implemented in C, but this performance score might be expected but since Axis2 is designed to be highly modular. It is not well known whether Axis2 C still outperforms the PHP SOAP engine, which has been designed around with a monolithic approach and is also written in C. However, from the point of view of productivity, programmability, ease of deployment, and debuggability, it is well behind the other engines. Especially in terms of ease of debugging, it is not straightforward to debug web services written for Axis2 C, as the relationship between resulting core dumps from segmentation faults and the original web service definitions is not obvious.  
PHP as a SOAP engine  
Considering the recent trend of popular programming languages, PHP fulfills the fundamental language  
requirements of productivity and performance. Even though Java can meet these requirements, since PHP is used in a variety of web applications and SOAP and WS-\* has not yet been successful used in the public space (with the exception of certain robust enterprise applications), we claim that that PHP is in a good position

Related Work

Ceccet, et al. compared EJB, Java Servlet, and PHP [2][15]. They also reported that the PHP language performs the best. However the experiments conducted in the paper use traditional web application workloads such as TPC-W for benchmarking e-commerce applications. Hence this performance result is not applicable to the performance comparison of web service engines that requires a significant amount of XML processing.

As described in Section 3.2, articles such as [8] reported on the performance comparison of various XML parsers but they only focus on the XML processing so it does not include performance comparison for end-to-end scenarios. Many IT vendors provide high performance web service engines such as IBM WebSphere™ Application Server, along with performance benchmark reports. Our paper focuses on the performance implication of different  
programming languages. Several efforts [4][17] have worked on the performance optimization of XML processing especially focusing on the optimization of Java-based web service engines.

**6 Conclusions and Future Work**

This paper describes the comparison of web services engines implemented in PHP with those implemented in Java and C. Given different architecture and different programming language, the experimental results show that PHP is reasonably high performing compared to Java and C-based implementations while providing users with high software productivity. Our contribution will provide average programmers with more awareness that PHP is a viable option for publishing SOAP/WS-\* based web services in addition to the currently popular REST-style web services.

Several topics can be further explored. First we can extend a variety of web services such as other scenarios in WSTest or more real-world scenarios that also involve business logic and database access. Moreover, we can explore a wider variety of server configurations. In this paper we only use Axis2 Java and its default data binding, ADB, but other web service implementations in Java could differ from the reported result in this paper. We can  
also compare different dynamic scripting languages such as Ruby and Python.Additionally, rather than focusing on SOAP-based services for web services, in future work, we could explore performance in terms of RESTful services. XML or JSON is used as a data format after accessing web services via the HTTP Get operation, but we can compare PHP and Axis2 since PHP is popular for RESTful web services today, whereas Axis2 is not really used that way even though Axis2 provides both RESTful and SOAP functionality for same web service implementation without extra programming effort

**References**[1] Alex Ng, Shiping Chen, and Paul Greenfield, “AnEvaluation of Contemporary Commercial SOAPImplementations”, AWSA (Australasian Workshop onSoftware and System Architecture) 2004  
[2] Emmanuel Cecchet, Anupam Chandra, et.al,“Performance Comparison of Middleware Architecturesfor Generating Dynamic Web Content”, Middleware 2003  
[3] Subramanyam S., et al., “Designing a web services benchmark”, In ICWS’03. International Conference on Web Services, Las Vegas, USA.  
[4] Toyotaro Suzumura, Satoshi Makino and Naohiko Uramoto, “Optimizing Differential XML Processing by  
Leveraging Schema and Statistics”, ICSOC 2006, Springer, Lecture Notes in Computer Science, pp. 264-p276  
[5] Srinath Perasa, Chathura Herath, Jaliya Ekanayake, et. Al. “Axis2, Middleware for Next Generation Web  
Services”, ICWS 2006, URL: http://apache.axis.org/axis2/