**杭州电子科技大学**

**毕业设计（论文）外文文献翻译**

|  |  |
| --- | --- |
| **毕业设计（论文）题目** | **基于微信小程序的便笺系统设计** |
| **翻译题目** | **在云中实现协作文档处理** |
| **学 院** | **网络空间安全** |
| **专 业** | **信息安全** |
| **姓 名** | **力晶鑫** |
| **班 级** | **14083611** |
| **学 号** | **14084122** |
| **指导教师** | **吕秋云** |

# 在云中实现协作文档处理

摘要 - 文档处理是目前最广泛使用和开发的应用程序之一。随着近期高速互联网和分布式计算的发展，将文档处理转移到网络甚至是基于云的环境变成了现实。移动办公最初的益处是通过将文件转化为云为中小企业节省购买，维护和升级软件和硬件的成本。但是，这种趋势最显著的优势是使用户能够在共享云基础上进行实时协作编辑文件的功能。因此，将办公应用程序迁移到云中是办公应用发展的必然趋势。作者在文献[12]中提出了一种新的高效文档处理模型（DPC）。在本文中，我们通过Google App Engine在Google云中实施了DPC模型。我们的初步测试结果证实了提议的DPC模型使用户能够将Office文档在云中进行协作处理的可行性。

关键词 - Office文档处理，云，协作编辑。

## **I. 介绍**

今天的办公文件可能是非常复杂的，可能包含复杂的公式，插图，视频剪辑和控制信息。同时，办公文件处理程序为了反映用户需求，已演变成为大型，复杂和强大的应用程序。其中，科学论文通常需要之前开发和研究的好例子来通过协作方式润色技术报告。为了满足这样的要求，许多不同的软件系统已经由商业和学术开发人员共同开发完成。由于对此类系统的全面审查超出了本文的范围，感兴趣的读者可以查看引用[4] [5]。

合作团队的这种需求可能比文献[10]所介绍的单独的软件要困难得多。第一个问题是购买，维护和升级所需的软件和硬件的成本。第二个问题是伴随系统故障和当应用程序变得更复杂时，可能出现人为错误。为了满足快速变化的用户需求和维护市场，办公文件处理开发商通过不断增加新的产品继续升级他们的产品功能和特点。这些给人们带来额外的成本用于升级他们现有的软件和硬件来满足更大的存储空间和更快的CPU需求。换言之，人们将不得不不断投资以支持他们的日常文件制作和处理。

在我们以前的研究中，我们已经说过云计算技术可以是一种替代方法解决这个问题，因为它能够实现通过互联网使用和访问服务和存储设施。云提供的服务可能是一个Web应用程序，因此办公文件处理是其合适的候选人。用户通过网络应用程序处理的办公文件，可以创建，编辑和共享他们的文档，而无需安装复杂的软件套件。毫无疑问，用户将会保存数千美元的硬件和软件。此外，用户可以将更多精力放在基于最新文档处理应用程序的创意作品上，而无需考虑软件和硬件升级的成本。

谷歌和微软是两大提供云中办公文件处理服务的公司之一。这些在云中的应用程序的细节将在本文的背景部分中予以介绍，然而，在我们以前的研究中，我们讨论过他们都没有提供在共享文档上进行适当的协作编辑的功能。他们都实现了协作却并未实现分别处理不同的逻辑对象的功能。在本例中，多个用户可以在共享文档上编辑一个句子，甚至一个词，这会在用户之间造成混乱和混乱。为了解决这个问题，我们以前的研究提出了云中的文档处理模型（DPC模型）。DPC模型使用户能够在云中以合适的粒度协作处理其办公文档，这将在第3节中简要介绍。

本文的主要目的是描述提出的DPC模型的实现。通过实施，办公文档处理将以适当的协作编辑粒度成为云中可用的Web应用程序。用户能够通过浏览器处理他们的文档处理工作，而无需在他们自己的计算机上安装Office文档处理应用程序。

## **II. 背景**

如前所述，Google和Microsoft通过云提供办公文档处理。Google Docs是一款免费的基于网络的办公套件，由Google提供数据存储服务。它允许用户在线创建和编辑文档，同时与其他用户进行实时协作。Microsoft Office 365 是商业软件加服务，提供来自Microsoft Corporation的一组产品。Office 365包括通过Internet交付和访问的Microsoft Office套件的桌面应用程序和Microsoft服务器产品的托管版本。这两个应用都要求强调对用户之间的文档编辑协作的支持。通过我们的测试和评估，我们发现这两种产品的协作编辑没有适当的粒度。例如，Google文档可让多个用户在线编辑一个共享文档，并允许不同的用户同时编辑一个句子，甚至一个单词。但是，在Google文档中，即使用户知道发生更改的位置，也不会通知他们内容更改，因为Google文档没有突出显示更改。

如图1所示，当用户在第二行输入“段落”的单词时，另一用户开始在与第一用户相同的位置键入字符。在这种情况下，第一个用户会被字符混淆，因为他输入的字符和其他人输入的字符之间没有视觉差异。只有第一个用户被注意到有另一个用户正在编辑这个句子，但他不知道这个改变是什么。当更多的用户在处理共享文档时，这个混乱和混乱问题可能会变得更糟。这个问题的原因是编辑操作不分别基于不同的对象。文档的内容作为一个单独的对象进行处理。我们还发现Microsoft Office 365中存在同样的问题。

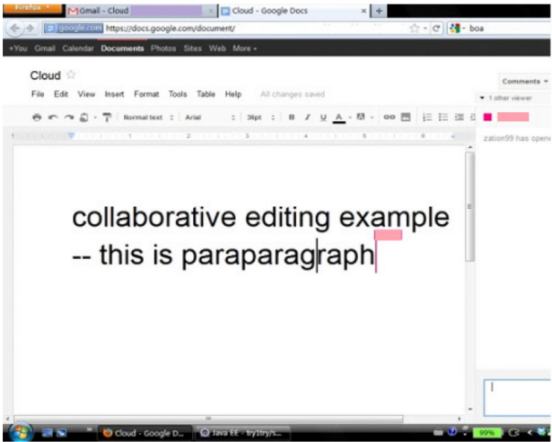


图1 在谷歌文档中协作编辑

为了解决此问题并使用户能够以合适的粒度协作处理其办公文档云，我们以前的研究提出了DPC模型，这将在下一节中简要介绍。与Google文档类似，本文中描述的DPC模型的实现构建了它的Web应用程序，并将其作为“软件即服务”部署在云中。

## **III. 背景**

DPC模型最初由作者在2010年提出，用于云中的文档处理。 它是基于XML逻辑结构的面向对象。它将文档中的可编辑组件视为不同的对象，并为用户提供对每个对象的访问权限。因此，处理共享文档的多个用户可以实时地基于不同的对象执行协作编辑。这种机制为文档处理协作提供了更合理的粒度，因为DPC模型将文档内容作为逻辑对象处理，而不是将其视为字符串流。

根据DPC模型，整个文档将被划分为表1中列出的13个对象，其中包括9个包含基本对象的复合对象和4个原子基本对象。

## 表1：DPC目标描述

|  |  |  |
| --- | --- | --- |
|  | 对象名称 | 对象描述 |
| 复合对象 | 内容 | 指定文档的属性 |
| 内容源 | 指定元数据 |
| 页眉和页脚 | 指定页眉和页脚 |
| 样式及字体 | 指定样式和字体的设置 |
| 脚注及尾注 | 指定的脚注及尾注 |
| 注释 | 指定注释 |
| 段落 | 指定的段落属性 |
| 表格 | 指定表格 |
| 运行 | 指定在父字段中的运行内容的属性 |
| 基本对象 | 超链接 | 指定超链接 |
| 区域 | 指定区域 |
| 文本 | 指定在文档中显示的运行的文字文本 |
| 图片 | 指定的图片 |

一旦被分成DPC对象，整个文档成为DPC对象的一个单元。 每个DPC对象都是一个单位工作分配，这些分配将发送给云中的处理器。所有单位的组合是整个文件。 用户将会被引导到云中的目标对象以完成其编辑工作。在所有编辑工作完成后，DPC对象将会完成收集并组合成最终结果文件。在为了在云环境中获得完整的利用率，DPC模型还定义了以下八个公式：

|  |
| --- |
| DPC={DOC,MIDDLEWARE,PROCESSORS<bag PROCESSOR>} |
| DOC={ROOT,ASSIST\_INFO} |
| PROCESSOR={APPS<bag APP>} |
| ROOT={NONLEAF<bag COMPOSITE\_OB>,LEAF<bag BASIC\_OB>} |
| COMPOSITE\_OBJ={OBJECT,DESCENDANT<bag ROOT>} |
| BASIC\_OBJ={OBJECT} |
| OBJECT={NAME,ACCESS\_PATH,ON\_EDITING} |
| ON\_EDITING::=Busy|Idle |

有关这些公式的详细信息首先在中介绍。值得注意的是，在公式六中，DPC模型使用XPath描述的ACCESS\_PATH来表示划分后的不同对象，并引导用户到他们想要编辑的目标对象。 图2显示了在云中用ACCESS\_PATH划分后的DPC对象。 由于每个节点的XPath在XML文档中都是唯一的，因此它可以是云中DPC对象的标识符。

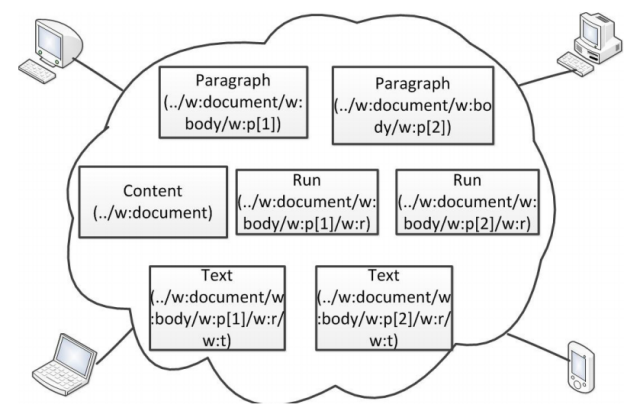
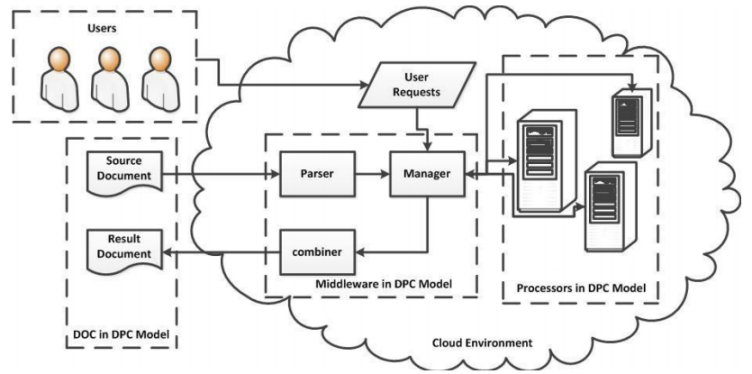


图2 DPC 对象以及他们的 ACCESS\_PATH.

## **IV. 实施**

DPC实现的Web应用程序预计将提供文档处理并使授权用户可以从浏览器访问该文档。文档的所有者能够邀请其他人同时处理同一文档。该实现使用Java编写，用于后端，JavaScript用于前端。对于前端，我们使用JavaScript库中的JQuery库来处理基本的文本编辑，例如添加和删除字符，改变字符的样式和大小等。同时，为了提供更好的编辑功能，我们将 CKEditer 在我们的实现中。CKEditor是一个在网页中使用的文本编辑器。由于我们的实现是用于处理网页上文本的Web应用程序，因此CKEditer是实现此目标的好工具。此外，CKEditer获得了灵活的开源和商业许可。

正如前一节中的公式一所定义的那样，在DPC模型的第一级，有三个主要组件：DOC，中间件和处理器。因此，在图3所示的实现中存在三个对应的主要部分。



管理员

用户请求

解析器

组合

源文件

结果文件

用户

DPC模型中的中间件

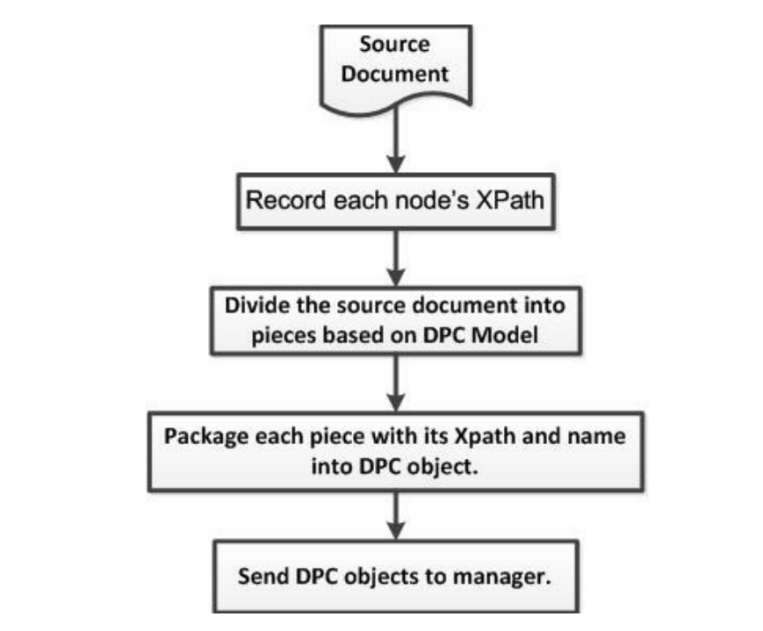
云环境

DPC环境中的进程

图3 DPC模型实施概述。

如图3所示，DOC部分包括将被上传到云的源文档以及从云下载的结果文档。在云环境中，中间件包括解析器，管理器和组合器。解析器和管理器的工作流程如图4和图5所示。组合器的工作流程是根据每个结果片段的XPath结合管理器的结果，这与解析器的反方向流程类似。

如图4所示，解析器在接收到一个源文档后，记录源中每个节点的XPath。 然后，解析器将源文档分成几部分。之后，解析器将每个片段与其对应的XPath打包到DPC对象中。最后，解析器将DPC对象发送给管理器。我们以ISO 29500格式文件为例来说明这个过程。图5中的几个XML文档描述了ISO 29500文档，其中docx格式基于ISO 29500，其分别包含“[Content\_Types] .xml”文档和两个名为“\_rels”和“word”的文件夹。 在名为“word”的文件夹中，有几个构成示例文档主要内容的XML文档。



将DPC对象发送给管理员

将每个部分的Xpath和名称

打包到DPC对象中

基于DPC模型将源文档

分成多个部分

记录每个节点的XPath

源文档

图4 解析器的主要工作流程。

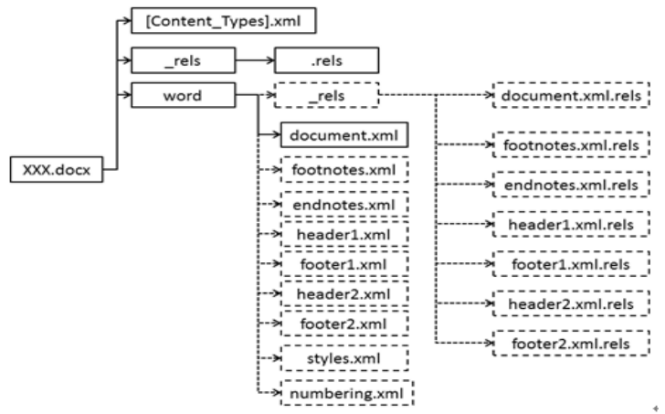


图5 ISO 29500文件的结构。

为了轻松划分示例文档以绘制DPC对象，解析器创建一个新的XML文档以包含来自源的所有的单个XML文档。有了这个，很容易记录每个DPC对象的XPath，并且保持DPC对象的完整性也很容易。 在用于将所有XML文档集成到源文档中的新XML中，解析器根据其自己的标题（如“workbook.xml”）标记每个XML文档，用于从workbook.xml标记该文档。这一整合步骤之后的源文件如图6所示。

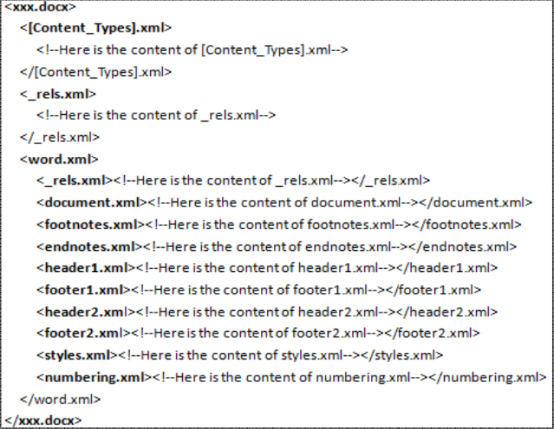


图6 整合后的源文件

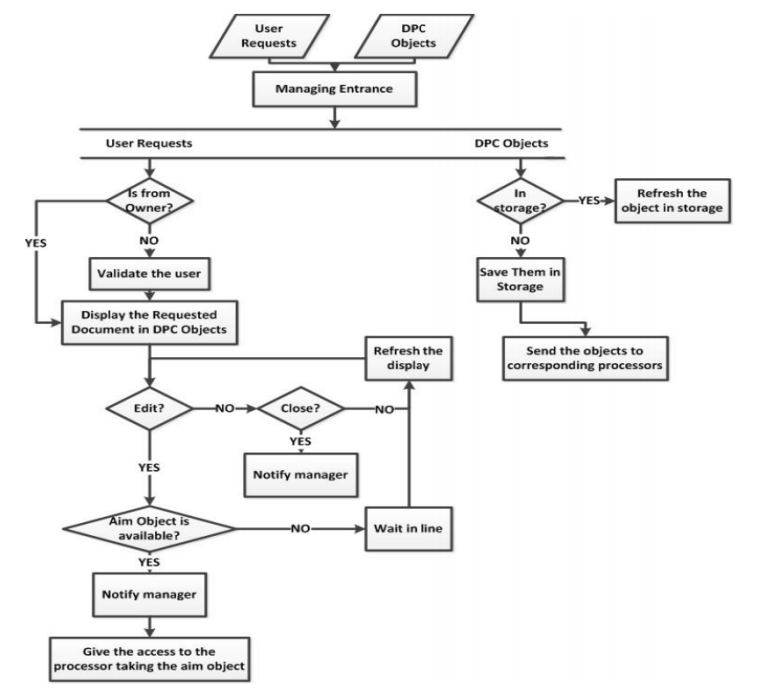
我们使用可扩展样式表语言转换（XSLT）技术来整合这些单独的文档。XSLT是一种基于XML的语言，用于转换XML文档，生成新的XML文档而无需更改原始XML文档。例如，图7显示了用于集成的XSLT样式表的主要步骤，通过它，源文档中的所有XML文档都集成到一个XML文档中。



图7 用于集成的XSLT样式表

通过集成的结果文档，解析器记录每个节点的XPath。例如，<document.xml>节点的XPath是“xxx.docx / word.xml / document.xml”。在记录XPath之后，解析器分割结果文档，然后取出相应的DPC对象。如果用户想要编辑段落的内容，他/她必须有权访问该对象。显然，会有一些不属于任何DPC对象的节点。这些节点也将作为备份发送给管理器。在从解析器或用户接收到数据后，管理器将调用相应的java servlet来处理数据。管理器的工作流程如图8所示。

如图8所示，管理者处理两种输入。第一个由用户请求组成，第二个由DPC对象组成。对于DPC对象，管理员将确定这些DPC对象是否已保存。如果输入未保存，则表示此文档首次上载到云中，管理员将其保存在其存储中。然后，管理器将发送DPC对象到相应的处理器。如果输入已保存，则表示这些DPC对象来自处理器而不是解析器，并且管理器将刷新之前保存的DPC对象。存储中的DPC对象用于向用户显示整个文档，因此需要按时刷新。



编辑？

刷新显示

将对象发送给相应的处理器

将它们保存在存储中

在存储中？

刷新存储中的对象

在dpc对象中显示请求的文档

验证用户

拥有者

DPC对象

用户反馈

管理员入口

DPC对象

提供对处理器的访问，以获取目标对象。

通知管理员

目标对象是可用的

用户反馈

排队等候

刷新显示

关闭？

关闭？

拥有者

图8 管理员的工作流程

如果输入包含用户请求，则管理员将检查这些请求是否来自目标文档的所有者。该文档的所有者之前上传了该文档，将决定哪个用户可以编辑该文档。为此，所有者需要向可以进行协作编辑的用户发送邀请。然后，用户在登录后即可进行编辑。但是，如果来自所有者，管理员将在浏览器上显示整个文档。如果不是，管理员将首先验证用户，然后浏览器可以在验证后显示文档。另一方面，如果用户想要编辑上传的文档，则该请求将指示用户想要编辑的文档的哪一部分。通过这样的请求，管理员将检查文档中的对象是否可用于编辑。如果可用，管理员将授权用户进行编辑。否则，用户需要在该行中等待目标对象。由于文档按需保存，管理员会定期刷新显示。

本文中DPC模型通过Google App Engine 部署在Google Cloud中得以实现。 Google App Engine是一个云计算平台，在Google管理的数据中心中提供平台作为服务和托管Web应用程序。 实施DPC模型的应用程序将作为托管在Google管理的数据中心中的Web应用程序启用。

## **V. 测试**

在本节中，我们介绍了我们设计的测试用例，以测试上传，单用户编辑和DPC模型实现的多用户功能的协作编辑。以下段落分别介绍了四种不同的测试用例及其结果。 这些测试用例在Firefox 7.0的浏览器上运行，用于黑盒测试的功能测试。

表2 测试用例I

|  |  |
| --- | --- |
| 类型 | 上传文档的用例 |
| 目的 | 通过DPC的实现将示例文档上传到云上，并在浏览器中显示其内容 |
| 测试数据 | DOCX的两段文档 |
| 测试步骤 | 开始一个浏览器的应用程序选择示例文档 ，点击上传按钮 |
| 测试结果 | 通过 |
| 截图 | 在图九展示 |

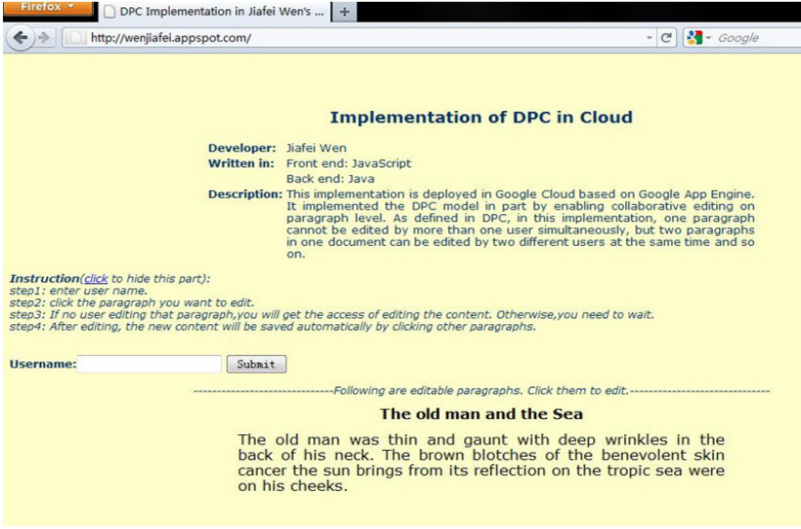


图9 测试用例截图I

如表2所示：测试用例I和图9，黑色字体的段落是示例文档的内容，它们也是浏览器上的可编辑段落。另一方面，用户可以在登录后单击内容来编辑它们，如表III：测试用例II和图10所述。当用户单击该段落时，该段落将变为可编辑字段。

表3 测试用例Ⅱ

|  |  |
| --- | --- |
| 类型 | 在浏览器中编辑文档 |
| 目的 | 编辑在浏览器上显示的一个示例文档 |
| 测试数据 | 上传测试用例I的示例文档 |
| 测试步骤 | 点击第一段并编辑内容 |
| 测试结果 | 通过 |
| 截图 | 在图十中显示 |

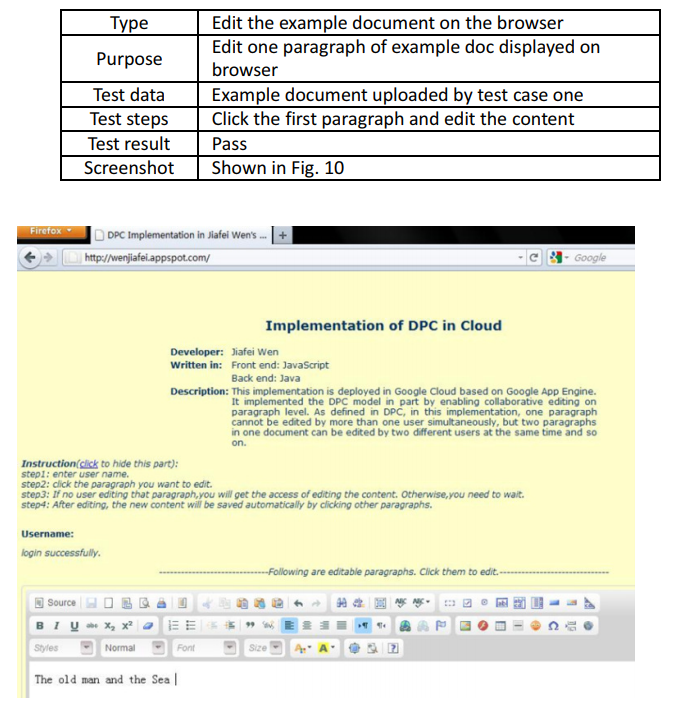


图10 测试用例II截图

当段落正在编辑中时，其他用户无法通过其他浏览器编辑该段落。如果用户点击正在编辑的段落，他/她将收到“另一个用户正在编辑该段落”的信息，如表IV：测试用例III中所述，如图11所示。

表 4 测试用例 III

|  |  |
| --- | --- |
| 类型 | 编辑一个正在编辑的段落 |
| 目的 | 使用不同的用户名编辑一个正在被编辑的段落，浏览器将会给用户一个无效的示意 |
| 测试数据 | 云中的文档 |
| 测试步骤 | 使用浏览器启动DPC的应用程序，选择示例文档，点击上传按钮 |
| 测试结果 | 通过 |
| 截图 | 在图十一中显示 |

测试用例IV和图12中显示了多个用户在共享文档上编辑不同的段落，尽管不同的浏览器同时出现。

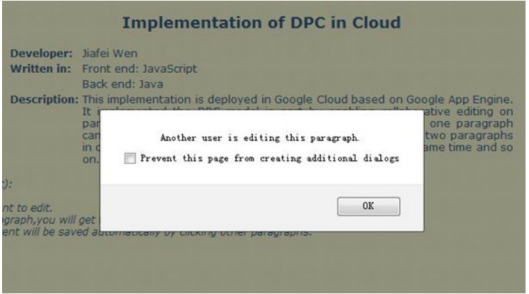
图11 测试用例III的截图

表5 测试用例IV

|  |  |
| --- | --- |
| 类型 | 两个用户通过不同的浏览器同时编辑一个共享文档的两个段落 |
| 目的 | 通过不同的浏览器进行合作编辑 |
| 测试数据 | 云中的文档 |
| 测试步骤 | 通过不同的用户名以及浏览器登录应用程序，单机一个浏览器的第一个段落；在另一个浏览器中点击第二段。两段可以同时编辑 |
| 测试结果 | 通过 |
| 截图 | 在图十二中显示 |

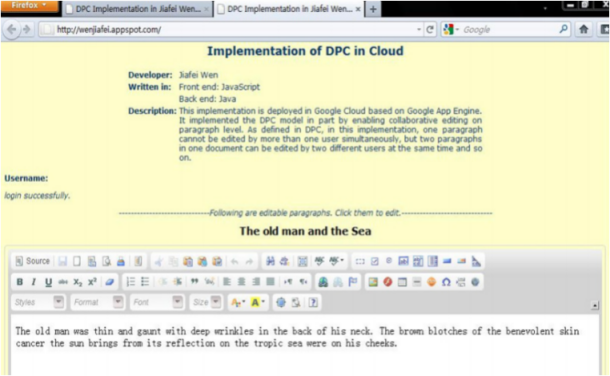


图12 在测试用例五中通过不同的浏览器实现合作编辑

## **VI. 总结**

将办公室文档处理移动到云中是IT行业最近的趋势。它不仅可以帮助用户节省软件和硬件升级成本，还可以让用户通过互联网在共享文档上进行协作编辑。 我们之前的研究已经提出了一个DPC模型，用于云中的高效办公文档处理。在本文中，我们将介绍DPC模型的实现，该模型通过Google App Engine部署在Google云中。该实现使用Java编写，用于后端，JavaScript用于前端。我们的实现证明，DPC实现的Web应用程序提供了文档处理功能，并使授权用户可以从浏览器访问文档。文档的所有者能够邀请其他人同时处理同一文档。同时，来自四个不同设计测试用例的结果进一步证实了我们提出的DPC模型提供了适当的协作编辑粒度。

## **参考文献**

1. J. C. Adler and S. Noël, “Evaluating and implementing a collaborative cffice document dystem,” Interacting with Computers, vol. 18, issue 4, 2006, pp. 1-18.
2. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brabdic, “Cloud computing and emerging IT platforms: vision, hype, and readily for delivering computing as the 5th utility,” Future Generation Computer Systems, vol. 25, issue 6, June 2009, pp. 599-616. [3] E. Kim and K. Severinson Eklundh, “How academics co-ordinate their documentation work and communicate about reviewing in collaborative writing,” Report #TRITA-NA-P9815, IPLab-151, Royal Institute of Technology (KTH), Sweden.
3. J. Newman and R. Newman, “Three modes of collaborative authoring in computers and writing: state of the art (P.O. Holt and N. Williams, Eds.),” Oxford: Intellect Books, 1992, pp. 20-28.
4. S. Noël and J. M. Robert, “How the Web is Used to Support Collaborative Writing,” Behaviour & Information Technology, vol. 22, 2003, pp. 245-262.
5. S. Noël and J. M. Rober, “Empirical study on collaborative writing: what do co-authors do, use, and like?” Computer Supported Cooperative Work: The Journal of Collaborative Computing, vol. 13, 2004, pp. 63- 89.
6. S. Petride, A. Tarachandani, N. Agarwal, and S. Idicula, “Managing and processing office documents in oracle XML database,” Proc. the 3rd International Conference on Advances, Knowledge, and Data Applications, 2011, pp. 89-95.
7. I. R. Posner and R. M. Baecker, “How people writet together, in readings in groupware and computersSupported cooperative work: assisting human-human collaboration (R.M. Baecker, Ed.),” 1992, pp. 239-250.
8. C. B. Rizzi, C. M. M. C. Alonso, E. B. Hassan, L. M. R. Tarouco, and L. M. J. De Seixas, “EquiText: a helping tool in the elaboration of collaborative texts,” Proc. 2000 Society for Information Technology & Teacher Education International Conference, pp. 2314-2319.
9. C. Sun and C. Ellis, “Operational transformation in real-time group editors: issues, algorithms, and achievements,” Proc.the 1998 ACM conference on Computer supported cooperative work.
10. Y. Tang, C. Yan, and C. Y. Suen, “Document processing for automatic knowledge acquisition,” IEEE Transactions on Knowledge and Data Engineering, vol. 6, February 1994. pp. 3-21.
11. J. Wen, S. Lam, and X. Wu, “A Model for office document processing and collaboration in the cloud,” Proc.the 2011 International Conference on Parallel and Distributed Processing Techniques and Applications, Jul. 2011, Las Vegas, USA.
12. L. Zhang and Q. Zhou, “CCOA: cloud computing open architecture,” Proc. IEEE International Conference on Web Services, 2009.
13. <http://www.google.com/google-d-s/intl/en/tour1.html.>
14. http://office365.microsoft.com/en-US/online-services.aspx.
15. XML Path Language (XPath) 2.0. W3C Recommendation, January 2007.
16. http://www.w3.org/TR/xpath20/.
17. <http://jquery.com/.>
18. http://ckeditor.com/what-is-ckeditor.
19. http://ckeditor.com/license.
20. XSL Transformations (XSLT) Version 1.0; http://www.w3.org/TR/xslt.

[22] http://en.wikipedia.org/wiki/Google\_App\_Engine.