PAPER

A Kind of Optimization Method of Loading Documents in OpenOffice.org*

Yuqing LAN[†a)], ***Member***, **Li LI**[†], ***and*** **Wenbin ZHOU**[†], ***Nonmembers***

SUMMARY As a giant in open source community, OpenOffice.org has become the most popular office suite within Linux community. But OpenOffice.org is relatively slow while loading documents. Research shows that the most time consuming part is importing one page of whole document. If there are many pages in a document, the accumulation of time consumed can be astonishing. Therefore, this paper proposes a solution, which has improved the speed of loading documents through asynchronous importing mechanism: a document is not imported as a whole, but only part of the document is imported at first for display, then mechanism in the background is started to asynchronously import the remaining parts, and insert it into the drawing queue of OpenOffice.org for display. In this way, the problem can be solved and users don't have to wait for a long time. Application start-up time testing tool has been used to test the time consumed in loading different pages of documents before and after optimization of OpenOffice.org, then, we adopt the regression theory to analyse the correlation between the page number of documents and the loading time. In addition, visual modeling of the experimental data are acquired with the aid of matlab. An obvious increase in loading speed can be seen after a comparison of the time consumed to load a document before and after the solution is adopted. And then, using Microsoft Office compared with the optimized OpenOffice.org, their loading speeds are almost same. The results of the experiments show the effectiveness of this solution.

key words: *OpenOffice.org, loading document, optimization, asynchronous import*

1. Introduction

OpenOffice.org [1] is a cross-platform-friendly software suite which can run in Windows, Linux, Macintosh, Solaris and other operating systems. It is compatible with other office suites. What's more, OpenOffice.org is a free software, and anyone is free to download, use, and develop it further [2]. Therefore, it is becoming more and more popular. However, OpenOffice.org is relatively slow in loading documents, particularly large files, and sometimes, the users have to wait for a intolerable period of time. Consequently, it is necessary to optimize the loading mechanism of OpenOffice.org in order to solve this problem.

As for document loading, People have been studying how to solve the problem of slow document loading speed of OpenOffice.org, and many ways are available, such as pre-loading techniques [6]: an instance of OpenOffice.org is written into the memory before any document is opened, so that subsequent documents can be opened in a shorter time. However, although the speed is faster, the instance will occupy system resources. There are still many other ways: parameters can be added to optimize the loading speed of OpenOffice.org, or a new and much steadier version of Gcc can be adopted to carry out the compiling process, or a larger memory can be used, or configuration of OpenOffice.org (such as changing the size of Graphics cache) can be performed. These methods, however, are an optimization of operating system, rather than of OpenOffice.org itself. This paper will put forward an asynchronous loading method to improve the loading speed of OpenOffice.org by changing the mechanism of loading documents.

Asynchronous loading method has been studied extensively in different areas. Among those researches, the optimization of the loading speed of web pages gains the most attention. Zhang Xincun, from Shanghai Jiaotong University, implemented a method of asynchronous loading of web pages based on product structure tree, which enables equipment manufacturing enterprises to organize and access a huge amount of product data rapidly and accurately in a web environment, thus facilitating the publishing and promotion of product information as well as the digitalization of management [3]. Andi Ahmad Dahlan, from Ritsumeikan University, put forward a method called Asynchronous Predictive Fetch to optimize the loading speed of web pages [4]. The emergence of Ajax technology is an indication of the full development of the asynchronous loading method of web pages [5]. Asynchronous loading method can also be used in the field of digital circuit. For example, Paul J. Patchen and Arlington put forward a method called asynchronously loadable D-TYPT FLIP, which, compared with conventional method, used less active devices and smaller filp-flop cell. As for documents loading in OpenOffice.org, there is hardly any research in asynchronous loading method. This paper proposes a kind of asynchronous loading method which can greatly increase the loading speed of documents in OpenOffice.org.

2. The Principle of Loading Documents in OpenOffice.org

2.1 Supported File Formats of OpenOffice.org

As one of the world's leading open source software, OpenOffic.org can compete with software products of MS Office, greatly breaking its monopoly in the field of office

Manuscript received March 23, 2011.
Manuscript revised October 12, 2011.
[†]The authors are with the Beihang University, 100191, China.
*This paper is supported by The National Project of Core Electronic Devices, High-end General Chips and Basic Software-"Domestic basic software testing for integration and application" under Grant No.2009ZX01045-005-002.
a) E-mail: lanyuqing@buaa.edu.cn
DOI: 10.1587/transinf.E95.D.778

software. Now OpenOffic.org is becoming more and more popular among enthusiasts of open source software. Office suites, like RedOffice Chinese 2000, NeoShine Office, are all successful secondary developments on the basis of OpenOffic.org.

In the information age, documents are still the main medium of information exchange. However, for various reasons, each software adopts its own format as its document storage standard. To avoid the incompatibilities between various documents, OASIS (Open Document Format for Office Applications), an international consortium, proposed to create structured documents, that is to use a definite, unambiguous way to organize the parts of a document. In order to solve the problem of incompatibilities between documents, OASIS defines the standard formatted file for office software development - OpenDocument [7], [8]. OpenDocument is a new generation of standard file format for office software. It provides formats for word processing, spreadsheets, presentations and other formats. OpenOffice.org uses OpenDocument as its default storage format for the documents. In fact, an OpenOffice.org document is a compressed file (.zip). If we change the file extension of an OpenOffice.org document to "zip", and unzip it, we can see that there are a lot of XML (eXtensible Markup Language) files, which contain the contents and information of documents. It is parsing the XML files that makes it slower to open an OpenOffice.org document. By using XML files to store content and information, OpenOffice.org can separate content from style, and make its documents much smaller. At the same time, when a document is stored as XML format, its content can be accessed not only through OpenOffice.org, but also through certain Programming [9];

2.2 The Principle of Loading Documents in OpenOffice.org

OpenOffice.org supports many kinds of file formats, including its own format OpenDocument as well as the format of MS-Document. The loading processes of them are similar.

While loading a document, OpenOffice.org will first be read in the whole document at a time, and then display it. Obviously, when the document is very large, users will have to wait for a long time and sometimes the waiting can be intolerable. Because MS Powerpoint is widely used, here we will take the loading of MS Powerpoint as an example to illustrate the process of loading a document [10] (the cases of other documents are similar).

OpenOffice.org uses class SdDrawDocument to model the whole presentation [11], and when opening a PPT document, OpenOffice.org creates a SdDrawDocument object for it. Each page in the slide is represented by an object SdPage. The following table lists classes closely related to the loading of PPT documents. They are illustrated as Table 1.

Table 1 Classes related to the loading of PPT documents.

class	main function
`DrawDocShell`	Provides some essential operations such as Open and Save
`SdPPTFilter`	Document filter, used to filter PPT documents
`SdPPTImport`	Provides the interface for method to import PPT documents
`ImplSdPPTImport`	Implements the import process of PPT documents
`SdrPowerpointImport`	Imports one page of a PPT documents
`SlideSorterView`	Provides a sort view for all slides
`DrawView`	Draws views for a slide
`SdDrawDocument`	Draws the whole document
`SdPage`	Represents a page of a document

The process of loading a PPT document is as follows:

(1) When a PPT document is opened, object DrawDocShell is created to provide some essential operations, such as Open, Save and Save As. Then method ConvertFrom() is called. This method will call the relevant classes of filter according to the type of the file to be imported. Since here the file type is PPT document, an object of class SdPPTFilter is created.

(2) After object SdPPTFilter is created, the method Import() of object SdPPTFilter is called to import the document. This method then performs some initialization settings and property settings and hands the job to class SdPPTImport.

(3) class SdPPTImport provides the interface for method Import() to import PPT documents.

(4) The object of class ImplSdPPTImport will call method Import() to implement the import process of PPT documents. Method Import() first creates a master page, imports an object for it, and then starts importing slides. The process of importing is like this: first the number of pages is obtained, then a pointer SdPage is created for every slide, finally method ImportPage(), a method of class SdrPowerpointImport (the parent class of class ImplSdPPTImport), is called to insert the contents of the slide to the object to which SdPage points. Thus a slide is imported. All the slides can be imported by repeating Step Four.

(5) After all the slides are imported, SlideSortView and DrawView are created to display them.

The process of importing a PPT document is illustrated in Fig. 1:

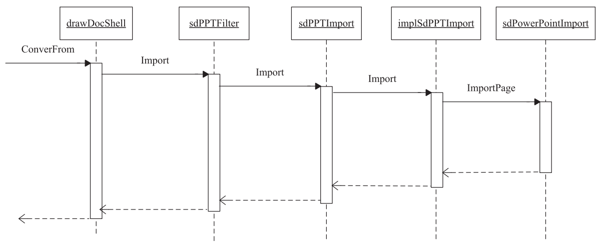

Fig. 1 The process of importing a PPT document.

Till now the overall process of importing a PPT document is discussed. This process also applies to the case when importing other types of documents except that different filters are called. This mechanism causes OpenOffice.org to open a document slowly. Therefore, it is necessary to optimize this mechanism.

2.3 The Bottleneck of Loading Documents in OpenOffice.orgs

Based on test, we have found out that for OpenOffice.org, in the process of importing a document, the method ImportPage() of SdrPowerpointImport, which is used to import one page of all slides, is the most time-consuming method. Although it takes a short time to import a single page, when there are many pages, time consumed can be astonishing. Therefore, to increase the speed of importing documents, the accumulation of time must be minimized.

In OpenOffice.org, Import(), a method of class ImplSdPPTImport, will first import all slides of a document at a time before they are drawn and displayed. This mechanism has led to the accumulation of time consumed to import the slides by calling method ImportPage() of class SdrPowerpointImport. The bottleneck of opening a PPT document lies on importing all slides of a document at a time before they are displayed. Therefore, to reduce the time consumed to load the PPT documents, we can first import a part of the document and immediately draw and display it, and then, while displaying, import the remaining part step by step. In this way we get the asynchronous importing method.

3. Optimization Algorithm for Loading Documents in OpenOffice.org

3.1 Ways to Improve the Efficiency of Loading Documents in OpenOffice.org

In Sects. 2.2 and 2.3, this paper has analyzed the process and bottleneck of loading documents in OpenOffice.org. Nevertheless, besides mechanism of importing, it also takes much time for OpenOffice.org to parsing and typesetting the documents. Therefore, there are three ways to increase the speed of loading documents: First, to increase the speed of parsing the documents by adopting fast parsing algorithms; Second, to increase the speed of typesetting by improving the strategy of typesetting; Third, to increase the speed of importing by optimizing the mechanism of importing. This paper will adopt the last method to optimize the mechanism of loading documents.

3.2 Optimizing and Improving OpenOffice.org

From the above analysis, we know the mechanism of loading documents in OpenOffice.org and optimizing the process of loading means to import parts of a document asynchronously instead of importing a whole document at a time. The overall thought of asynchronous import is like this: first, a threshold is defined as FirstLoadPageNum, to indicate the number of slides imported at a time; then get the total number of pages of a PPT document TotalPageCount. When it is less than FirstLoadPageNum, it is imported as usual; otherwise, a pointer pPage is constructed to point to this page, and its contents will not be imported. These pointers are saved in a List for later asynchronous import. After all the pages are scanned, they will be drawn and displayed. At this time, a timer will be started in the background to trigger the mechanism of asynchronous import, which fetches a pointer pPage, deletes it from a List, fills its contents and then calls the relevant method to import this page. Then the page is inserted into the queue of drawing. This process can be call a cycle. Subsequent cycles will go on until the whole document is imported. That is the overall thought of asynchronous import, which avoids the advantage of importing the whole document at a time.

3.3 The Improved Methods of Loading Presentation Document in OpenOffice.org

The Sect. 3.2 describes the general idea of asynchronously loading the document in OpenOffice.org. In this section, we load a specific document in OpenOffice.org, that is a presentation file, to illustrate how to realize asynchronous importing algorithm. As mentioned in Sect. 2.2, a presentation document in OpenOffice.org is expressed by class SdDrawDocument, while a slide in the document is represented by class SdPage. To achieve algorithm of asynchronous loading presentation file in OpenOffice.org, the class closely related to the loading documents should be needed to make the following modified expansion:

Step1: Modify the class SdPage representing a slide, and add two Boolean variables mbIsImported and mbIsImporting to the class SdPage, which are used to respectively indicate the slide that has been imported completely and the slide that is in the process of loading, and then respectively add methods of the corresponding setter and getter.

Step2: Modify the class SdDrawDocument representing the whole presentation, add a boolean variable mbDocLoaded to the class, denote that the current document has been imported completely, and then add the corresponding methods of setter and getter. By default, the member variable is TRUE, only if the total number of document pages exceeds the threshold value FirstLoadPageNum, its value is set to FALSE. Adding a vector mNotLoadPPTVec to store the slide pointer pPage whose contents are not imported.

Step3: Adding the method ImportSlidePage() to the class ImplSdPPTImport which realizes the importing of slide in order to indicate that a slide has been imported. In the method ImportSlidePage(), determining whether a given pPage has been imported or is being imported. If it has been imported or is being imported, then return FALSE, which indicates that the page needs no longer be imported again; If the page has not been imported, setting the pPage's state as being imported, and doing some pre-settings before imported, then calling the method ImportPage() of the class SdrpointpointImport to complete the import of the page.

Step4: In the class SdPPTImport, add the method ImportSlidePage(), indicating that a slide has been imported. This method will delegate the task of importing a slide to the method ImportSlidePage() of class ImplSdPPTImport.

Step5: In the filter class of SdPPTFilter, add the method ImportSlidePage(), which calls ImportSlidePage() of the class SdPPTImport to complete the import of one slide.

Step6: Add the method StartLoadTimer() in the class SdDrawDocument, which is used to express that a timer is started at intervals.

Step7: Add a method in the drawing class SlideSorter-View which is used to invoke the method StartLoadTimer(), and complete the import of not imported page.

3.4 Asynchronously Loading Document Algorithm of OpenOffice.org

After the modification to the relevant class during the OpenOffice.org loading procedure in the Sect. 3.3, the algorithm of asynchronous importing documents in OpenOffice.org can be described as follows:

Step1: Utilize the modified method in the Sect. 3.3 to replace the relevant class in the OpenOffice.org.

Step2: Open the document, create the object DrawDocShell, and call the method ConvertFrom() of DrawDocShell.

Step3: The ConvertFrom() method invokes different filters according to the type of imported files. Assuming that imported file is the format of PPT (similar for other formats), so as to create the object of the class SdPPTFilter, and import documents by calling the method Import() of SdPPTFilter.

Step4: After SdPPTFilter is gone through some operations of initialization, call the method Import() of the class SdPPTImport;

Step5: The method Import() of the class SdPPTImport delegate the task of import to the method Import() of the class ImplSdPPTImport which is class SdPPTImport's realization.

Step6: Obtain the total number of document pages, and judge whether it exceeds the initial one-time importing threshold FirstLoadPageNum, if not, go to Step 6; If the number exceeds the specified threshold FirstLoadPageNum, go to Step7.

Step7: Call the method ImportPage() of class SdrPowerpointImport to import the slide, and determine whether it is the last page, if it is, go to Step 8, otherwise go to Step 5.

Step8: Construct a null pointer pPage, then inserte pPage without contents into the vector mNotLoadPPTVecof the class SdDrawDocument, and set mbDocLoaded as FALSE, indicating that the document load is not completed. At the same time, set mbIsImported of the page pointed by pPage as FALSE, indicating that page has not been imported. Judge whether the document number is the last page, if it is, go to Step 8, otherwise go to Step 5;

Step9: Create SlideSortView and DrawView to display the document, followed by the drawing and displaying of the document;

Step10: In the process of drawing determines whether the document is imported completely, if not, the SliderSorterView calls StartLoadTimer () to start the timer, and go to Step 10, otherwise go to Step 13;

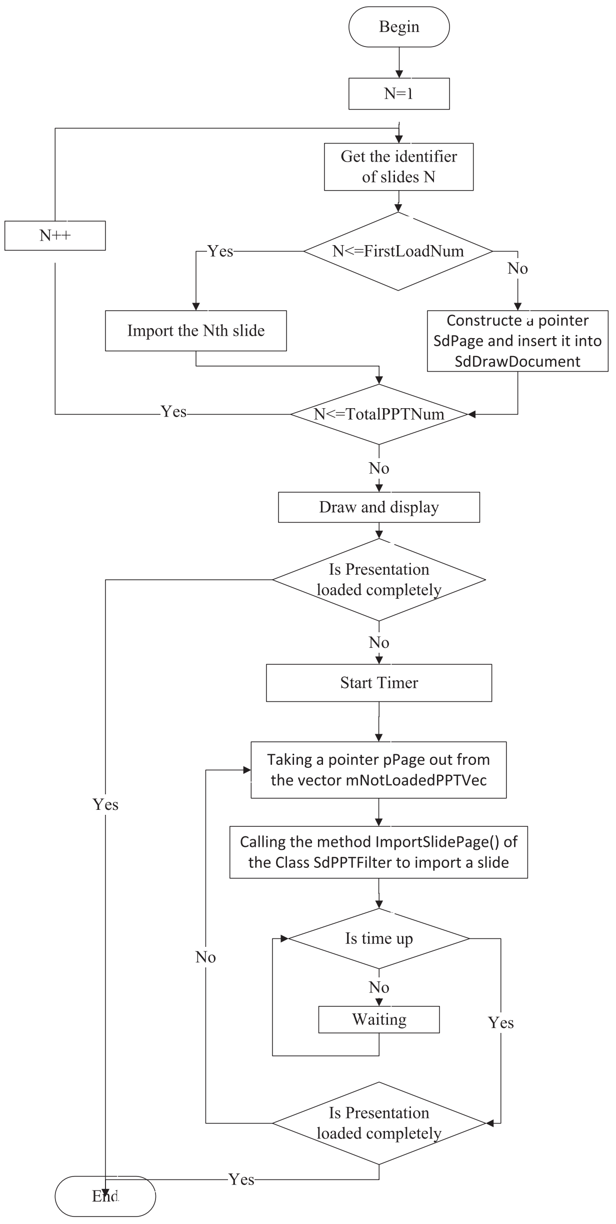
Begin
N=1
Get the identifier of slides N
N++
N<=FirstLoadNum
Yes
No
Import the Nth slide
Constructe a pointer SdPage and insert it into SdDrawDocument
N<=TotalPPTNum
Yes
No
Draw and display
Is Presentation loaded completely
No
Yes
Start Timer
Taking a pointer pPage out from the vector mNotLoadedPPTVec
Calling the method ImportSlidePage() of the Class SdPPTFilter to import a slide
Is time up
No
Waiting
Yes
Is Presentation loaded completely
No
Yes
End

Fig. 2 Asynchronously loading document flow chart of OpenOffice.org.

Step11: Activate the mechanism of importing document, and determine whether the mNotLoadedPPTVec of SdDrawDocument is empty, if empty, go to Step 13; If not, take a pointer pPage out from the vector mNotLoadedPPTVec and remove it from the mNotLoadPPTVec, then fill the content of pPage.

Step12: Call the method ImportSlidePage() of the class SdPPTFilter to import a slide, and set the mbIsImported of this slide as TRUE.

Sterp13: Judge whether the time of timer is up. if it is, go to Setp 10, otherwise continue to wait until the time is up, then go to Step 10.

Step14: The document is imported completely. These are the steps of the asynchronous importing algorithm of OpenOffice.org. Figure 2 is a flow chart of the algorithm.

4. Evaluation

4.1 The Theory of Simple Linear Regression

We use regression analysis theory in mathematical statistics to research the correlation between the page number of documents and loading time. Since we only focus on the impact of loading speed on document pages, so we choose the simple regression analysis.

First of all, let's look at the regression theory. Simple regression is to study the relationship between the random variable y and the general variable x, for each x with a determined value,y is a random variable with a determined distribution. Although the value of y can not be determined by the value of x, from statistical significance, it is hoped that the value of x can determine the average of y, that is $E(y) = \mu(x)$. The basis of regression analysis is to search the expression $E(y) = \mu(x)$ of the functional relationship between the average of y and x. $\mu(x)$ is called regression function as y on x, or simply called regression as y on x. If relation of $\mu(x)$ and x is linear, $\mu(x) = a + bx$, the problem of estimating the mathematical expectation of y by a linear function $a + bx$ is known as simple linear regression problem. According to simple regression theory linear regression model can be shown as:

$$\begin{cases} y = a + bx + \mathcal{E} \\ \mathcal{E} \sim N(0, \sigma^2), \quad \text{for } a, b, \sigma^2 \text{ have no relation with x} \end{cases} \tag{1}$$

If $\hat{a}, \hat{b}$ are written as the estimated value of a, b, the estimated value of $a + bx$ is $\hat{a} + \hat{b}x$, and it can be denoted as $\hat{y}$,that is $\hat{y} = \hat{a} + \hat{b}x$. It is called a linear regression equation of y on x, $\hat{a}, \hat{b}$ are called the regression coefficients.

According to simple linear regression theory, the uniformly minimum variance linear unbiased estimate of a, b (Best Linear Unbiased Estimate) is the least-square estimation of a, b. Therefore, a given set of sample values (x_i, y_i) can get that the least-square estimations of a, b is:

$$\begin{cases} \hat{a} = \bar{y} + \hat{b}\bar{x} \\ \hat{b} = \frac{L_{xy}}{L_{xx}} \end{cases} \tag{2}$$

Of which,

$$\begin{cases} L_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2 = \sum_{i=1}^{n} x_i^2 - n\bar{x}^2 \\ L_{xy} = \sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) = \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y} \end{cases} \tag{3}$$

Set $Q(a, b) = \sum_{i=1}^{n}(y_i - a - bx_i)^2$, demand $Q(a, b)$ on the a and b partial derivatives, and set them to be zero. Then we can get:

$$\begin{cases} \frac{\partial Q}{\partial a} = -2 \sum_{i=1}^{n}(y_i - a - bx_i) = 0 \\ \frac{\partial Q}{\partial b} = -2 \sum_{i=1}^{n}(y_i - a - bx_i) = 0 \end{cases} \tag{4}$$

Order The system of equations to get:

$$\begin{cases} na + n\bar{x}b = n\bar{y} \\ n\bar{x} + \sum_{i=1}^{n} x_i^2 b = \sum_{i=1}^{n} x_i y_i \end{cases} \tag{5}$$

Solve the equations can get the value of $\hat{a}$ and $\hat{b}$.

4.2 Time Consumed Comparison for OpenOffice.org Before and after Optimization

In order to verify the correctness and effectiveness of the algorithm in the above asynchronous importing documents, recompilation [12] and setup the modified OpenOffice.org suite. After that, we design the following experiment.

1. Selecting PPT documents with different pages, we choose the pages of PPT documents are 20, 50, 100, 200, 400, 800 and 1600.

2. In order to make little different contents in each slide of each document, the same master plate and basically the same content in each slide will be used. In this way the negative impact of experimental results caused by different contents will be reduced.

3. Due to man-made reasons, there is error in each measurement of importing time, so the approach of repeated measurements and taking average can be used to reduce error.

4. The hardware conditions used in the experiment are: CPU: Intel (R) Core ^{TM}2 Duo P8600, Main frequency: 2.4 GHZ, Memory: 3 G, Operating System: Windows XP Professional SP3, OpenOffice.org version: 3.1

MATLAB is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages. In this paper, MATLAB is used to analyze data and draw graphics: regress, a library function, is used to perform regression calculation, and plot function, legend function is used to graphics drawing.

The Table 2 and Table 3 is the experimental data. They are the time consumed [13] comparison for OpenOffice.org importing different sizes of documents before and after using asynchronous importing methods, of which Table 2 is the time consumed before optimization and Table 3 is the time consumed after using the asynchronous importing algorithm.

Bringing the values before optimization into Eq. (1), the regression equation before optimization is:

Table 2 Time consumed in loading PPT documents with different number of pages in OpenOffice.org after optimization (unit: millisecond).

Pages	1st	2nd	3rd	Average
20	2101	2311	2399	2370
50	3455	3699	3770	3641
100	5915	5761	5633	5770
200	10316	10464	10095	10292
400	20898	21061	20854	20937
800	48656	48576	48526	48587
1600	111116	106353	106118	107862

Table 3 Time consumed in loading PPT documents with different number of pages in OpenOffice.org after optimization (unit: millisecond).

Pages	1st	2nd	3rd	Average
20	1130	1310	1424	1288
50	1613	1698	1802	1704
100	2206	2343	2466	2338
200	3694	3517	3527	3579
400	6790	6674	6970	6811
800	16265	16567	17112	16648
1600	46637	46534	47897	47022

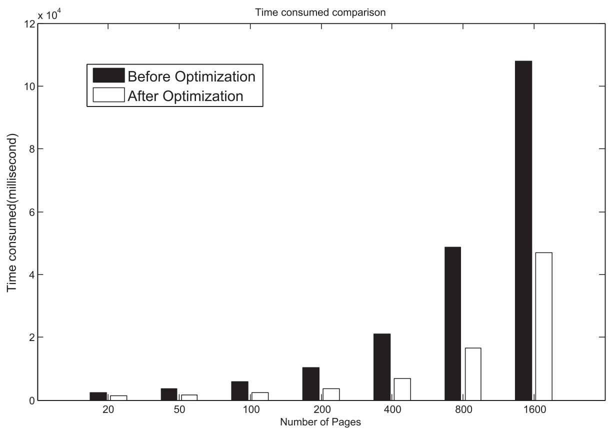

Fig. 3 Histogram of time consumed comparison for OpenOffice.org.

$$\hat{y}_{before} = -1819.8 + 66.9x \tag{6}$$

Bringing the values after optimization into Eq. (1), the regression equation after optimization is:

$$\hat{y}_{after} = -1572.3 + 28.5x \tag{7}$$

It can be seen from the slope of the regression equation that the optimized slope is much less than the slope before optimization, which explicates the optimized growth rate is much smaller than that before optimization. Suppose that the slope before optimization is k_1, and the optimized slope is k_2, then $k_1/k_2 \simeq 3$, Therefore, the method of asynchronous importing algorithm described in this paper makes the speed of loading PPT documents nearly 3 times faster in OpenOffice.org, which is consistent with observations of the conclusion.

Through the analysis of the above experimental data, the variation tendency can be seen from histogram and graphical chart drawn in Fig. 3 and Fig. 4. It can be seen from the figures that with the increase of the numbers of presentations' pages, the time of OpenOffice.org loading documents also increases with an approximate linear growth; and It can be seen that the optimal time for OpenOffice.org loading documents has a huge reduction, which is basically $\frac{1}{3}$ of time consumed before optimization.

4.3 OpenOffice.org vs MS Office

In the same experiment settings, we collect statistics about time consumed in loading PPT documents with different number of pages in MS Office 2007. The results are listed in Table 4.

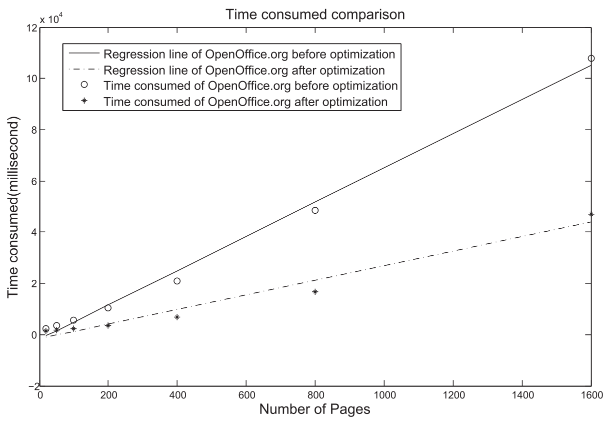

Fig. 4 Graphical chart of time consumed comparison for OpenOffice.org.

Table 4 Time consumed in loading PPT documents with different number of pages in MS Office.

Pages	1st	2nd	3rd	Average
20	1913	1842	1882	1879
50	2066	1931	2006	2001
100	3177	3053	3147	3126
200	5594	5491	5341	5475
400	6970	6487	6138	6532
800	14465	13973	13727	14055
1600	31212	30689	30683	30861

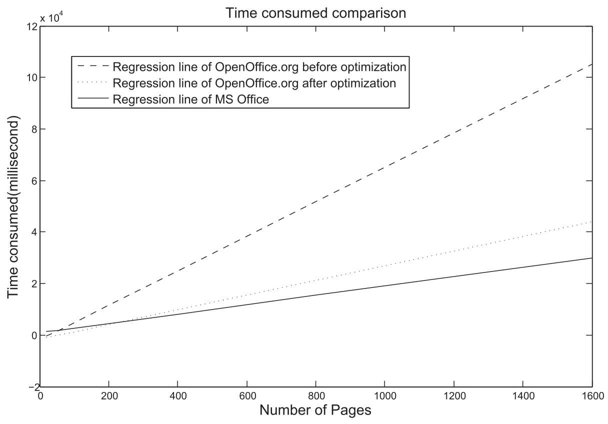

Fig. 5 Graphical chart of time consumed comparison for OpenOffice.org and MS Office.

Bringing the values of time consumed to Eq. (1), we can get a regression equation:

$$\hat{y}_{MS} = 927.6 + 18.12x \tag{8}$$

By analyzing and comparing data in Table 4, Table 3 and Table 2, a diagram can be drawn, as illustrated in Fig. 5.

The diagram shows that, with the increase of the number of pages, time consumed in loading documents is increasing for both OpenOffice.org and MS Office, and that after optimization, they consume almost a same period of time. In this way, user-friendliness of OpenOffice.org has been improved greatly, making a big step forward in the completion with MS Office.

4.4 Discussion

Compared with MS Office, OpenOffice.org has many advantages: it is platform independent, free, and has its standard of file formats. Besides, it can be updated more frequently. However, it takes more time to load a document, and its interface is not so user-friendly. After the adoption of the asynchronous loading algorithm put forward in this paper, the gap of loading speed between OpenOffice.org and MS Office has been bridged. Although the above experiment is conducted in the same hardware environment, and the same documents are used, the startup items may be different for MS Office and OpenOffice.org, which may lead to inaccuracy of the results. Disabling all the startup items of both can reduce the time consumed, and at the same time the results may be more precise. However, users generally don't disable all those items of their own accord, and these items are not the major factors influencing the loading of documents. In addition, experiment errors occur inevitably. Therefore, maintaining the default settings while testing meets the requirement of the experiment better.

5. Conclusion

Through the analysis and research of the process of loading documents in OpenOffice.org, the algorithm of asynchronous loading documents is presented in this paper, and this algorithm is applied into the process of loading the PPT document in OpenOffice.org. According to the experimental results of asynchronous importing PPT document in OpenOffice.org, we can conclude that the proposed algorithm can greatly improve the speed of loading the document in OpenOffice.org. In terms of loading other formats of documents in OpenOffice.org, the algorithm is equally effective, only replacing the corresponding filters and the loading class. Thus, the proposed algorithm of asynchronous importing documents in OpenOffice.org is universal and quite effective. Because OpenOffice.org is a cross-platform office software, and it can be operated in different platforms of operating systems, and the proposed asynchronous loading algorithm in this paper is the optimization within the internal OpenOffice.org, it can be compiled on different platforms to improve the loading speeds

Acknowledgements

We would like to thank the staff of the China Standard Software CO., LTD for providing useful advice on the understanding of source code of OpenOffice.org, and Jerry for helping us with the validity of English grammar of this paper. Thanks for all those who have helped us. Without their careful guidance and help, we couldn't have accomplished the experiments successfully. Without their kindness, this study would have been much more difficult.

References

[1] OpenOffice.org, "The Free and Open Productivity Suite," http://www.OpenOffice.org, accessed March 2. 2011.
[2] Brooks, J., OpenOffice 3.1 provides feature, performance boost. eWeek, 2009. 26(10).
[3] X. Zhuang, Y. Wu, and C. Li, "Design and realization of web-based asynchronous loading product structure tree," Machine Tool and Hydraulics, no.4, pp.64–66, 2006.
[4] A.A. Dahlan and T. Nishimura, "Implementation of asynchronous predictive fetch to improve the performance of ajax-enabled Web applications," Information Integration and Web-based Applications and Services (iiWAS), 2008 International Conf. On. 2008.
[5] Y. Chen, "Asynchronous Web development mode based on AJAX," J. Chengdu University (Natural Science Edition), vol.26, no.4, pp.313–316, Dec. 2007.
[6] F. Fedora, "How to make OpenOffice load faster," http://forums.fedoraforum.org/archive/index.php/t-33413.html, accessed March 2. 2011.
[7] OASIS, "Open document format for office applications (OpenDocument) v1.0," OASIS Standard, http://docs.oasis-open.org/office/v1.0/OpenDocument-v1.0-os.pdf, accessed March 2. 2011.
[8] OASIS, "Open document format for office applications (OpenDocument) v1.1," OASIS Standard, http://docs.oasis-open.org/office/v1.1/OpenDocument-v1.1.pdf, accessed March 2. 2011.
[9] Y. Li, "Research and analysis of OpenOfflc.org document structure," Computer Technology and Development, vol.16, no.10, pp.59–61, Oct. 2006.
[10] OpenOffice.org wiki, "OpenOffice.org Impress," http://wiki.services.openoffice.org/wiki/Impress, accessed March 2. 2011.
[11] OpenOffice.org wiki, "OpenOffice.org 2.0 Developer's Guide," http://wiki.services.OpenOffice.org/wiki/Documentation/DevGuide/OpenOffice.org_Developers_Guide, accessed: March 2. 2011.
[12] Wang Shu-bin, "Compile analysis and research of OpenOffice," J. Soochow University (Engineering science edition), vol.25, no.2, pp.58–62, April 2005.
[13] Y. Li, P. Xiao, and W. Deng. "The method to test Linux software performance," Computer and Communication Technologies in Agriculture Engineering (CCTAE), 2010 International Conf. On. 2010.

Yuqing Lan born in May 1969, Ph.D., Associate Professor in the Computer science and engineering school of Beihang University, specialized in software engineering and operating system, published 38 papers, 21 of them are included in EI. This paper is supported by The National Project of Core Electronic Devices, High-end General Chips and Basic Software-"Domestic basic software testing for integration and application" under Grant No.2009ZX01045-005-002.

Li Li born in Feb 1989, graduate student, studying at Beihang University. Research Interest is Software Engineering.

Wenbin Zhou born in July 1985, graduate student, studying at Beihang University. Research Interest is Software Engineering.