Opportunities for Bandwidth Adaptation in Microsoft Office Documents

Eyal de Lara[†], Dan S. Wallach[‡], and Willy Zwaenepoel[‡]

[†] Department of Electrical and Computer Engineering
[‡] Department of Computer Science
Rice University
{delara,dwallach,willy}@cs.rice.edu

Abstract

Microsoft Office, the most popular office productivity suite, produces large documents that can result in long download latencies for platforms with limited bandwidth. To reduce latency and improve the user's experience, these documents need to be adapted for transmission on a limited-bandwidth network.

To identify opportunities for adaptation, we characterize documents created by three popular applications from the Microsoft Office suite: Word, PowerPoint, and Excel. Our study encompasses over 12,500 documents retrieved from 935 different Web sites.

Our main conclusions are: 1) Microsoft Office documents are large and require adaptation on bandwidth-limited clients; 2) embedded objects and images account for the majority of the data in these documents, with image types being the most popular non-text content, suggesting that adaptation efforts should focus on these elements; 3) compression considerably reduces the size of these documents; and 4) the internal structure of these documents (pages, slides, or sheets) can be used to download elements on demand and reduce user-perceived latency.

1 Introduction

Microsoft Office is the most popular productivity suite for creating documents. Its popularity derives, to some extent, from its ability to create *compound documents* that include data from more than one application. The potentially large size of these documents results in long download and upload latencies for mobile clients accessing the documents through bandwidth-limited links [3, 11, 22]. To reduce latency and improve the user's experience, compound documents and the applications that operate on them need to adapt to the available bandwidth.

To identify opportunities for adapting compound documents we need to understand their main characteristics. However, most studies of content types, especially those done on the Web [4, 21, 23, 24] have consistently ignored compound documents or treated them as opaque data streams, ignoring the rich internal structure that can be used to enhance bandwidth adaptation. In this paper we present an analysis of Office compound documents downloaded from the Web. We focus on those characteristics of Office documents that have implications for bandwidth-limited clients, and identify opportunities for adaptation. Although we report our findings with an emphasis on bandwidth-limited clients, we believe that these results will be useful for office suite designers and people interested in working with compound documents in general.

We undertook this study as part of our Puppeteer project, which uses component-based technology to adapt applications for different operating environments. Puppeteer is well suited for adapting compound documents that include data generated by several software components. By exposing the hierarchy of component data in the compound document, and making calls to the run-time APIs that the components expose, Puppeteer adapts applications without changing their source code. In contrast, traditional adaptation approaches have not been successful for applications that operate on compound documents mainly because the complex and proprietary nature of these applications thwarts source code modifications [9, 10] and the inclusion of several complex data types, usually embedded in a single file, makes system-based adaptation hard [12, 18, 20].

For this paper, we studied compound documents generated by three popular applications of the Office suite: Word, PowerPoint, and Excel. We chose to focus on Office applications based on four factors. First, Office is the most widely-used productivity suite. Moreover, a significant number of Microsoft Office documents are available on the Web, enabling us to gather the data for our experiments. Second, the Office file formats, although proprietary, are reasonably well documented. Third, the Office applications are highly integrated with each other and have published run-time APIs that can be used by Puppeteer to adapt the applications. Fourth, Office 2000 supports two native file formats: the proprietary OLE-based binary format and a new XML format. By using Office 2000 to convert old files to the new XML format, we can compare the tradeoffs of using a proprietary binary-based file format against a modern standards-based text format, both as intermediate formats suitable for document editing, and as publishing formats, suitable only for reading.

Although we concentrate exclusively on Office documents, we believe that our results apply to compound documents generated by other productivity suites. Since most of these suites support roughly the same features (embedding, images, etc), and document content is driven largely by user needs, it is likely that the main characteristics of documents produced by various productivity suites (*e.g.* distribution of document size, percentage that have images, number of pages, slides, etc.) would be similar.

We downloaded over 12,500 documents, comprising over 4 GB of data, from 935 different sites. Our main results are:

1. Office documents are large, with average sizes of 196 KB, 891 KB, and 115 KB for Word, PowerPoint and Excel respectively. Their large sizes suggest a need for adaptation in low bandwidth situations.

2. Office documents are component rich. 18.19% of Word documents and 46.38% of PowerPoint documents have at least one embedded component. Images were the most common component type.

3. In large documents, images and components account for the majority of the data, suggesting that they should be the main target of the adaptation effort.

4. For small documents, the XML format produces much larger documents than OLE. For large documents, there is little difference.

5. Compression considerably reduces the size of documents in both formats. Moreover, once compressed there is no significant difference in the sizes of the two file formats.

6. XML formats are easier to parse and manipulate than the OLE-binary formats.

The rest of this document is organized as follows. Section 2 provides some background on compound documents and their enabling technology. We also discuss relevant characteristics of the three Office applications that we use in this study. Section 3 describes the documents we used in our experiments. Section 4 presents our experimental results. Section 5 discusses the relevance of our findings to other productivity suites. Finally, section 6 discusses our conclusions.

2 Background

To its user, a compound document appears to be a single unit of information, but in fact it can contain elements created by different applications. A compound document could, for instance, consist of a spreadsheet and several images embedded into a text document.

In the general case, every data type in a compound document (spreadsheet, text, image, sound, etc.) is created and managed by a different application. The different applications used to create the document can be thought of as *software components* that provide services that are invoked to create, edit, and display the compound document.

In the remainder of this section we review the technologies used by Office to enable compound documents. We start with an overview of COM, OLE, and Automation. We then talk about the two native file formats supported by Office. Finally, we present a taxonomy of components found in Office applications.

2.1 COM, OLE and Automation

Office compound documents are based on the Component Object Model (COM) [5] and the Object Linking and Embedding (OLE) [6] standards, which govern the interactions between the various software components used to create compound documents.

COM enables software components to export well-defined interfaces and interact with one another. In COM, software components implement their services as one or more COM objects. Every object implements one or more interfaces, each of which exports a number of methods. COM components communicate by invoking these methods.

OLE is a set of standard COM interfaces that enable users to create compound document by *linking* and *embedding* objects (components) into container applications, hence the name OLE.

Automation is an OLE technology, which enables third party applications to remotely control Office applications. Puppeteer adapts applications, to a large extent, by invoking Automation interfaces to modify application behavior when executing on bandwidth limited platforms. For example, using Automation interfaces, Puppeteer can adapt a large PowerPoint presentation by loading only a couple of slides, instead of the full presentation, before returning control to the user. While the user works on these slides, Puppeteer loads the remaining slides in the background, and as new slides become available, it instructs PowerPoint to append them to the presentation.

2.2 File formats

Office 2000 supports two native file formats: the traditional OLE-based binary format (hereafter, "OLE archive") and a new XML-based format. The OLE archives [13, 14, 15] rely on the OLE Structured Storage Interface (SSI) to provide a unified view of the compound document in a single file. SSI implements an abstraction similar to a file system

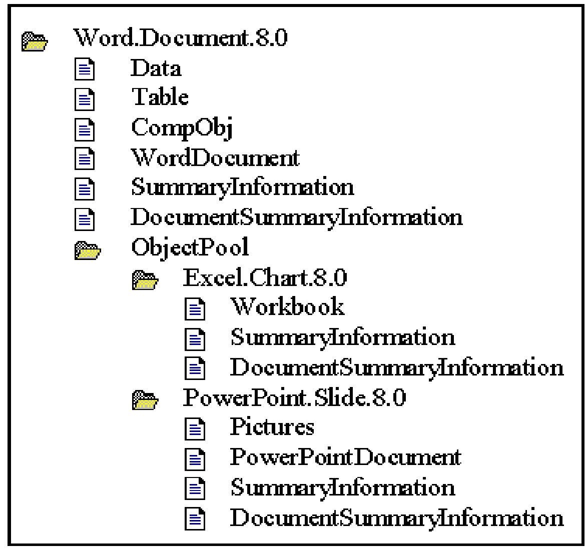

Figure 1: Word archive. The figure shows two embedded objects, an Excel chart and a PowerPoint slide, each stored in a separate SSI storage.

within a single file. It supports two types of objects: *storages* and *streams*. Storages are analogous to directories and contain streams or more storages. Streams are analogous to files and contain the components' data. Office applications vary in the way they use the OLE SSI to store embedded objects. Word and Excel, for instance, use a separate storage for every embedded component, making the component structure of the document visible to the OLE SSI. For example, figure 1 shows the structure of a Word archive with two embedded components. Notice how Word keeps each embedded object in a separate SSI storage. In contrast, PowerPoint compresses embedded object native data and stores it in the main application stream. While this strategy increases document compression, it limits the ability of third-party applications to manipulate components within a PowerPoint document.

The new XML format [17] provides a more browser-friendly option for storing Office documents. While an OLE archive appears as a single file, an XML document appears as an entire directory of XML files, approximately one per component, image, or slide. The current implementation of Office supports two forms of XML output: a compact low-fidelity representation that can be read by browsers but cannot be edited by Office tools, and a larger high quality representation that supports editing. In this study we focus on the latter XML representation because it is semantically comparable to the

OLE archive.

Aside from the number of files that they use, the two file formats differ mostly in their representation of text and formating information. Images and embedded component native data have similar representation in both formats, with the caveat that component data in the XML-based format is stored in a compressed OLE archive. Moreover, both formats keep in persistent storage two versions of the OLE components they embed. The first one consists of the embedded component's native data, which is used to initialize the component. This data is created and managed by the component itself. The second representation is a cached image of the state of the component the last time it was instantiated. This image, although created by the component, is managed by the container application. This image serves two purposes. First, it allows the document to be rendered quickly, since the code that understands the component's specific type need not be executed until the user wishes to modify the component. Second, the cached image allows the document to be rendered even on systems where some component types are not installed.

There is a significant difference in the way Office supports these two file formats. Office is able to load OLE archives incrementally over a random access file system. In contrast, XML documents must be read in their entirety before control is returned to the user, leading to higher latencies for opening and storing XML-based documents.

2.3 Component taxonomy

Conceptually, Office documents may have up to three classes of components: images, OLE-based embedded components, and virtual components. Images are graphic data that are stored and manipulated directly by the application. This includes the cached versions of any embedded components and any graphic data that the application manipulates directly. OLE-based embedded components are data created using a separate application, as described above. Among the most common types of embedded components are components that implement image types. To differentiate these image types (which are created by a separate application, and hence a type of component) from the primitive images managed by the application we will use the term "image components." Finally, virtual components are objects that are not implemented as OLE-based components but that are perceived by the user as separate entities (*i.e.*, pages in Word, slides in PowerPoint, and sheets in Excel).

Application	Documents	Sites
Word	6481	236
PowerPoint	2167	334
Excel	4056	378

Table 1: Data set. This table presents for each application, the number of documents and the number of Web sites from which they originated.

3 Data set

We collected Word, PowerPoint, and Excel documents from the Web. First, we used the AltaVista search engine [1] to obtain an initial set of URLs. In the first two weeks of October 1999, we searched for pages having links to files with suffixes we were interested in (`doc`, `ppt`, and `xls`). For example, we used the query *link:ppt domain:edu* to search for HTML pages in the edu domain that have links to PowerPoint documents. Then, we used GNU Wget [19] to recursively retrieve documents from our initial search results.

The reliance on a search engine to obtain the documents raises the question of the set representativity. On one hand, a search engine is likely to produce results that are dependent on the popularity of certain pages and documents, skewing the distribution towards these particular document types and producing a non-random set of documents. On the other hand we observe that our documents are fairly well distributed among domains, covering a wide range of user types. Moreover, the shape of the document size plots of section 4.1 and their close fit to the power-law distribution are similar to the results obtained by Cunha *et. al.* [7] in a study of client-based traces covering over half a million user requests for WWW documents.

All downloaded documents were in the binary OLE archive format. Because Office file formats vary from one version of Office to another, we first converted all our data to the Office 2000 formats. We removed documents that appeared to be corrupt or were not actually Office documents. The `doc` suffix, in particular, tends to be used by many appli-

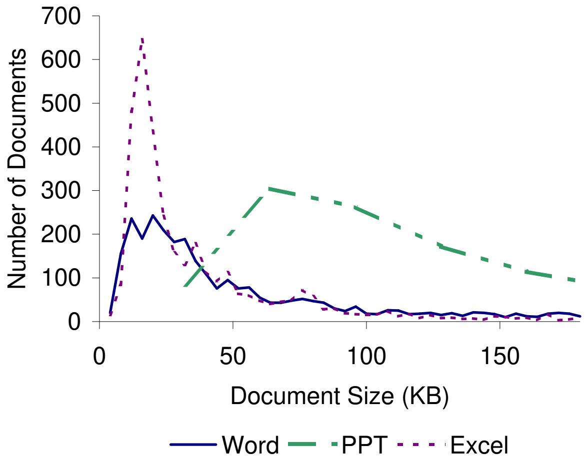

Figure 2: Size distribution of Word, PowerPoint, and Excel documents. Shown are documents with sizes up to 180 KB.

cations other than Microsoft Word. We also eliminated duplicates, removing approximately 5% of our data set.

We converted all the data to Office 2000 formats and we obtained the XML-based representation using Office's OLE Automation interfaces [16]. We wrote a simple Java application that uses OLE Automation to remotely control Office applications to perform data conversions.

Table 1 shows a summary of the documents. For each application, it presents the number of documents, and the number of Web sites from which they originated.

4 Experimental results

This section presents statistics we have measured for Office documents and the components within them. Based on these statistics, we identify opportunities for adapting the documents to bandwidth limited clients.

4.1 Document size

Table 2 shows general statistics for Word, PowerPoint, and Excel documents[1]. The most striking aspects of the data are the large average size of the documents and the large standard deviations of our sample.

Statistic	Application		
	Word	PowerPoint	Excel
average (KB)	196.24	891.48	115.02
stdev (KB)	528.44	2145.35	438.70

Table 2: Document size statistics.

Figure 2 shows the size distribution of Word, PowerPoint, and Excel documents. The histogram plots documents with sizes up to 180 KB. We observe that the distributions have the same general shape: a cluster around a common small value with a fairly long tail.

Figure 3 characterizes the distributions' tails by plotting document size frequencies for documents larger than 100 KB on a log-log scale. The linear fit[2] of the transformed data ($y \sim x^{-1.7124}$) with $R^2 = 0.8938$ suggests that the tail of the size distribution closely follows a power-law distribution, which is consistent with the large standard deviations of Table 2. The log-log scale histograms for the individual Word, PowerPoint, and Excel documents are not shown here since they are all similar to the cumulative distribution, with linear fits of $y \sim x^{-1.5254}$, $y \sim x^{-1.332}$, $y \sim x^{-1.7485}$, and $R^2 = 0.8612$, $R^2 = 0.8352$, and $R^2 = 0.8226$, respectively.

Interestingly, these results are similar to the findings of Cunha *et. al.* [7] where the size of HTML-based Web documents was found to follow the power-law distribution. However, while Cunha *et. al.* found that most HTML documents are quite small (usually between 256 and 512 bytes), Office documents tend to be much larger. Common sizes of Word and Excel documents size range from 12 KB to 24 KB, and common PowerPoint documents range from 48 KB to 80 KB.

4.2 Size breakdown

Figure 4 shows the breakdown of document sizes for Word documents. For every size category it shows

[1] The document size measurements for table 2 and figures 2 and 3 are based on the *raw* documents retrieved from the Web rather than the normalized Office 2000 translations described in Section 3.

[2] R^2 value ranges from 0 to 1 and reveals how closely the estimated trendline approximates the actual data. The closer the value is to 1, the better the estimate.

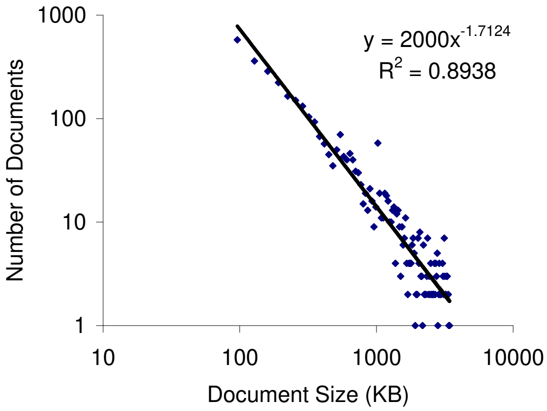

Figure 3: Size distribution of larger Office documents on a log-log scale. Document size frequencies are measured with 16384 byte bins.

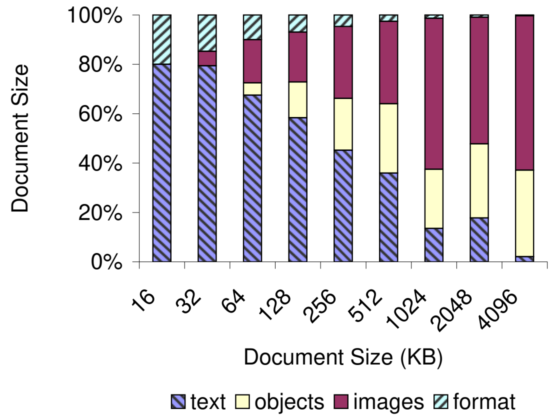

Figure 4: Size breakdown of Word documents. The plot shows that as documents get bigger, images and embedded component data account for most of the document's size.

the contributions of text, formatting information, embedded objects, and images to the documents size. We measured similar breakdowns for PowerPoint and Excel document, but because of space concerns we do not include them in this paper. The PowerPoint documents showed a similar trend to that of figure 4, while in Excel documents the text component accounts for over 95% of the document size in all the size categories.

Figure 4 show that small Word documents are dominated by text and formatting information. For larger Word documents, however, image and embedded component data become the prevalent contributors to document size. This data strongly suggests that efforts to improve access to compound documents should focus on the image and the embedded component data.

One possible optimization would be to remove the embedded component native data from documents that are fetched exclusively for reading. As described in section 2.2, this data is only necessary when editing an embedded component. Users are still able to display the document using the cached image of the component. We measured the savings of this schema and found that it would lead to a reduction in bandwidth requirements for Word and PowerPoint documents as high as 35% and 21%, respectively. PowerPoint documents show less potential benefit because PowerPoint compresses its components data before storing it in the OLE archive, whereas Word does not use compression.

4.3 Comparing OLE archives and XML

The results of our comparison are shown in table 3 and figures 5, 6, and 7. The data reveals that the XML representation can be significantly larger, requiring up to five times more space. XML efficiency is particularly low for small files, which according to our data are the most prevalent. However, XML efficiency improves dramatically as documents get larger.

To understand this, we must consider what happens when a document is converted from an OLE archive to XML. Text and formatting represented in XML takes more space than in Microsoft's internal representation. This explains the inefficiency of XML for small files. However, the XML conversion compresses embedded component data. PowerPoint already compresses its embedded component data, but Word and Excel do not. Because larger documents tend to be mostly images and components (see Figure 4), the XML representation becomes more efficient for large documents, and is even more efficient than the OLE archive for Word documents larger than 1 MB. Excel documents are primarily text and are most efficiently represented as OLE archives.

4.4 Compression

For the OLE archives, we compressed the document by applying gzip to the OLE archive. For the XML

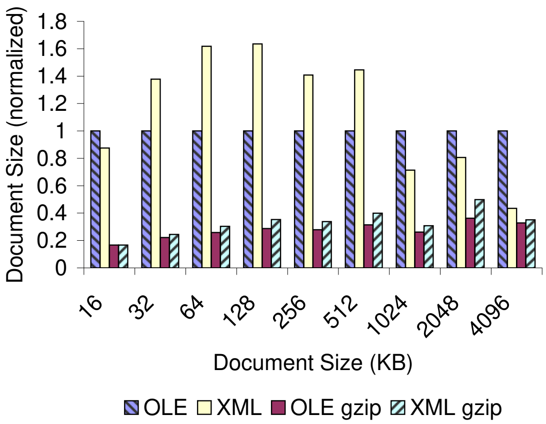

Figure 5: Size distribution of Word documents, with and without compression, for OLE archive and XML formats. Sizes are normalized by the size of the uncompressed OLE archive.

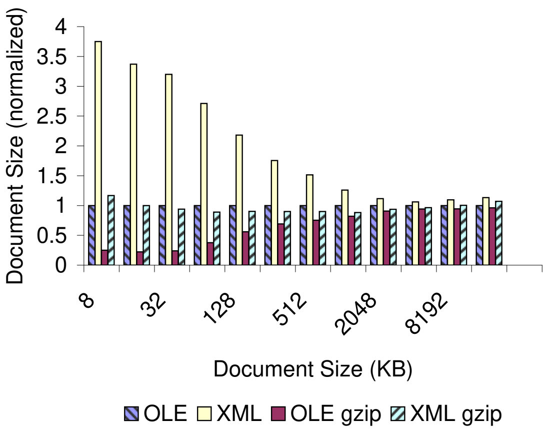

Figure 6: Size distribution of PowerPoint documents, with and without compression, for OLE archive and XML formats. Sizes are normalized by the size of the uncompressed OLE archive.

format, which uses several files, we compressed each file separately. This strategy emulates the potential benefits of a network infrastructure with built-in compression.

The results of these experiments are shown in table 3 and figures 5, 6, and 7. Compression has a dramatic effect on reducing the size of both OLE archives and XML files, achieving savings as high as 77% for the OLE and 90% for XML. Moreover, the difference in size between compressed OLE and compressed XML representations is small enough to be insignificant. This implies that neither representation has an inherent bandwidth advantage when used across a network.

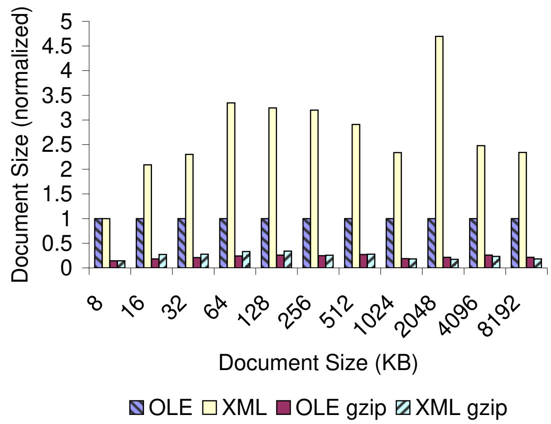

Figure 7: Size distribution of Excel documents, with and without compression, for OLE archive and XML formats. Sizes are normalized by the size of the uncompressed OLE archive.

4.5 Garbage collection

For OLE archives, Office optimizes "save" operations by appending modifications to the end of the file rather than rewriting the whole file every time. While this optimization allows for much faster document saves, it can lead to a significant increase in file sizes. If the user deletes or rewrites a substantial portion of a document and saves it, the original data, now garbage, will be retained. The extra data does not pose a problem for clients accessing the document over random access file systems, enabling the application to skip the dead data. Clients accessing documents over protocols that do not support random access, such as HTTP, are forced to download the whole document before opening it. The end result is fetching extra data that is never used.

In contrast, when a user asks Office to "save as," a new document is written from scratch, without any garbage that may have been in the original document.

We measured the changes in file size for OLE archives by using the "save as" operation. In this experiment we only considered documents that were already in Office 2000 file formats. Other documents are not included because the "save as" operation not only results in garbage collection but also reformats the documents to the Office 2000 formats, which may change document size.

		Application					
		Word		PowerPoint		Excel	
Format	Statistic	raw	gzip	raw	gzip	raw	gzip
OLE	average (KB)	209.19	61.43	579.53	481.18	110.23	25.67
	stdev (KB)	534.59	248.89	1671.36	1597.20	401.83	97.88
XML	average (KB)	226.43	74.14	795.17	549.03	336.90	28.37
	stdev (KB)	583.79	297.21	1851.92	1713.56	1562.04	92.02

Table 3: Size statistics for documents in raw OLE, OLE compressed with gzip, raw XML, and XML compressed with gzip. The statistics for OLE differ from those presented in Table 2 due to the conversion to Office 2000 formats.

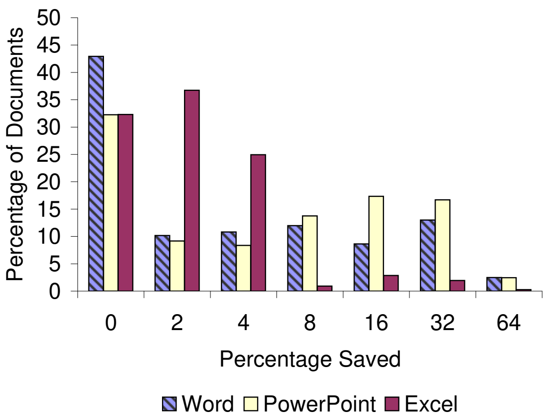

Figure 8: Percentage saved by garbage collection of OLE archive documents.

Figure 8 shows the results of this experiment. Most documents get some benefit from garbage collection. Interestingly, 24% of Word documents and 35% of PowerPoint documents achieve saving greater than 16%.

4.6 Components

In this section we first explore the effects of components on document size. We then present detailed statistics for the three types of components found in Office documents: images, embedded components, and virtual components.

4.6.1 Components and document size

We compared the sizes of Office documents with and without embedded components. Unsurprisingly, documents with embedded components are significantly larger. For example, the average size of Word documents with components is 557.28 KB, relative to an average of 112.32 KB for documents without components. PowerPoint and Excel documents show similar trends: PowerPoint documents average 1334.43 KB with components and 493.58 KB without, and Excel documents average 509.71 KB with components and 109.18 KB without.

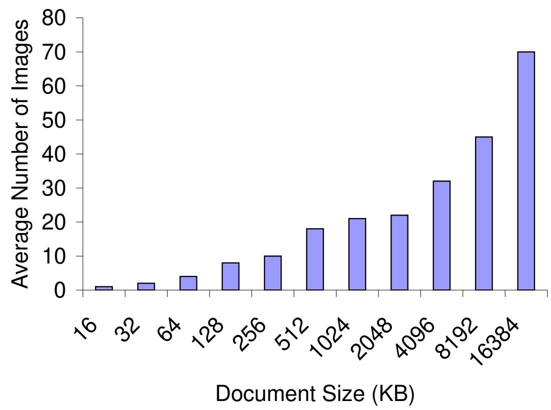

Figure 9: Average number of images in PowerPoint documents.

	Application	
Statistic	Word	PowerPoint
% of documents with images	34.62	77.01
avg. distinct images	6.01	10.62
avg. image size (KB)	21.58	47.82

Table 4: Images statistics for Word and PowerPoint documents. The table shows the percentage of documents that have at least one images, the average number of images in documents with images, and the average image size.

4.6.2 Images

Images are the most common type of non-text data found in Office documents. As table 4 shows, 34.62% of Word and 77.01% of PowerPoint documents have at least one image. We do not present

Statistic	Application Word	PowerPoint	Excel
% with components	18.19	46.38	1.42
number of component types	55	11	8
average number of components	6.71	9.18	9.05
average component size (KB)	37.62	18.51	26.01
stdev (KB)	141.78	109.33	133.37

Table 5: Embedded components statistics. The table shows the percentage of documents that have at least one embedded components, the number of different component types, the average number of components in a document, and the average and standard deviation of the size of embedded components.

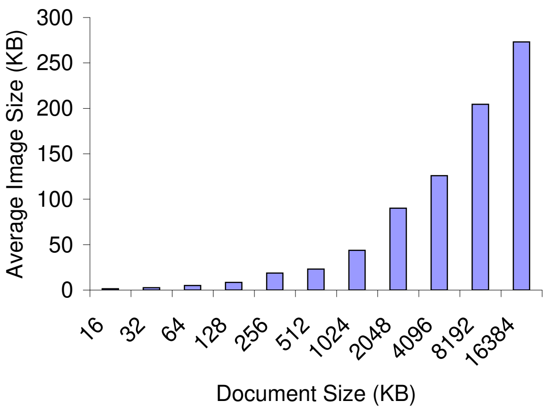

Figure 10: Average image size in PowerPoint documents.

results for Excel documents as very few of them have any images at all.

Figures 9 and 10 show the average number of distinct images and the average size of images for PowerPoint documents. We plot the number of distinct images instead of the total number of images because Office applications cache a single copy of every image regardless of the actual number of times the image appears in the document.

Both plots show similar trends, with increases in the number and size of images as documents get bigger. These results are consistent with the findings of section 4.2, where the size contribution of images to document size becomes the dominant factor as document size increases. The results for Word are similar, and are omitted for brevity.

We compared the average size of images in Office documents to the findings of previous Web studies [2, 23]. In general, these studies report the average size of images between 5 KB and 22 KB. In comparison, Office documents, especially PowerPoint documents, tend to have larger images. These results suggest that image distillation and other adaptation techniques are at least as important for compound documents as they are currently for Web documents.

We measured the reuse of images across our PowerPoint documents by calculating the Adler-32 checksum [8] of the image's data and counting the number of documents that have images with the same signatures. We found that of the 16,189 images embedded in PowerPoint documents, only 14,016 are distinct, while 1,241 images, or 8.85%, appeared in more than one document. We calculated the potential bandwidth savings of a perfect cache for a PowerPoint client reading all the documents in our dataset that came from the same Web site. We found that 26% of the Web sites get some bandwidth savings from the perfect cache, while 11% of the sites see reductions in required bandwidth that are greater than 20 %.

4.6.3 Embedded components

The data in table 5 shows that Office documents are rich in component data, with 18.19% of Word documents and 46.38% of PowerPoint documents having at least one embedded component. Furthermore, the data shows a high diversity of component types, with Word documents having the highest diversity.

Table 6 shows the popularity and average size of component types for Word, PowerPoint, and Excel documents. For all three applications, image components are either the first or second most popular type. Additionally, the average size of image components is among the largest of all types. This evidence further suggests that efforts toward reducing file size should focus on image types.

We observed that for all three applications, the av-

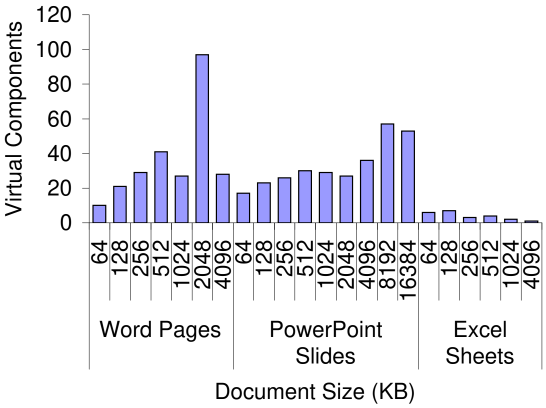

Figure 11: Average number of Word pages, PowerPoint slides, and Excel sheets.

erage number of embedded components and the average size of the components increases as documents get bigger. This trend is similar to the one shown in figure 9 and 10 for images, and is consistent with the findings of section 4.2, where the contribution of embedded components to the document size grows significantly as document size increases.

4.6.4 Virtual components

Table 7 and figure 11 show the average number of pages, slides, and sheets found in Word, PowerPoint and Excel documents. The substantial number of virtual components suggest that Office applications should be adapted to fetch virtual components on demand. To some extent this is already done by the applications when reading OLE archives in random access file systems or by Web browsers reading the XML representation. However, when Office applications open documents where random access is not available or when reading from the XML representation, they download the full document before returning control to the user. While providing universal random access support is likely to prove difficult, we believe that the current Office XML filters can be improved to support on-demand fetching. Alternatively, Puppeteer could provide this type of adaptation. As describe in section 2.1, Puppeteer could fetch the virtual components on demand and use OLE Automation to append them to the application.

Statistic	Word Pages	PowerPoint Slides	Excel Sheets
average	11.95	20.59	5.22
stdev	27.76	17.48	6.49

Table 7: Virtual components. The table shows statistics for pages in Word, slides in PowerPoint, and sheets in Excel documents.

5 Generality of results

In this section we discuss the applicability of our findings to productivity suites other than Office. For this discussion we assume that modern office productivity suites support roughly the same features (embedding, images, etc.) and that document design is driven largely by user needs.

While the average size of documents of different productivity suites is likely to be dependent on the specifics of the applications, the data suggest that the shape of the size distribution of documents would be similar for other productivity suites. We base this claim on the similarities we observed in the size distribution of Word, PowerPoint, and Excel documents. Although each application has a different file format (both for OLE archives and XML), the size of their documents follow the power-law distribution closely.

The breakdowns of document sizes are likely to be similar among productivity suites. While the specifics of how data is divided between images and components might change (*e.g.*, a productivity suite might implement all images as components), the increasing contribution of images and components to document size as documents grow is likely to hold true. Likewise, document design being driven by user needs, the structure of the documents (*i.e.*, the number of pages, slides, etc), the number of embedded components found in documents, and the popularity of images as the most common non-text type of content are likely to be similar.

6 Conclusions and discussion

We characterized compound documents generated by the three most popular applications of the Microsoft Office suite: Word, PowerPoint, and Excel. Our focus was on identifying opportunities

	Application					
	Word		PowerPoint		Excel	
Component	Avg. Size (KB)	% of Occur.	Avg. Size (KB)	% of Occur.	Avg. Size (KB)	% of Occur.
Equation	0.74	51.12	0.81	1.82		
Other Image			28.68	38.80	4.99	5.00
Word Picture	80.68	14.81	1.31	0.15	2066.83	1.00
Clip Art	10.38	8.93	5.00	41.12	2.56	15.00
Excel Sheet	153.46	8.53	53.18	5.27		
OLE Link	23.80	3.90				
Paint Brush	315.75	1.71			17.91	42.00
MS Draw	7.28	1.35				
PowerPoint	41.74	0.96				
Word			23.28	8.01	28.17	32.00
Graph			3.26	3.86		
Sound			3.35	0.02		
Other	97.52	8.68	8.72	0.94	2.39	5.00

Table 6: Average size and popularity of component types in Word, PowerPoint, and Excel documents.

for adapting these documents to the constraints of bandwidth-limited clients. Our study encompassed over 12,500 documents, comprising over 4 GB of data, retrieved from 935 different Web sites.

We identified the following opportunities for adaptation:

1. For large documents, images and components account for the majority of the data. Moreover, images and image components are the most common non-text data found in Office documents. These results suggest that components, and in particular images should be the main focus of any adaptation efforts. We are currently in the process of adding quality-aware transcoding and caching of images and components to Puppeteer and plan to measure the savings of these techniques.

2. For read only documents, discarding the native component data results in savings of up to 35% and 21% for Word and PowerPoint respectively.

3. Garbage collection of OLE archives achieves savings greater than 16% for 24% of Word and 35% of PowerPoint documents.

4. Compression achieves savings of 77% for OLE archives and 90% for XML. Moreover, once compressed there is no significant difference in the sizes of the two file formats. Since XML formats are significantly easier to parse and manipulate than OLE archives, they are a more attractive target for adaptation.

5. The structure of Office documents (pages, slides, and sheets) can be used to download elements on demand and reduce the time that users wait before they can start work on the document.

Furthermore, our experience studying the Office file formats resulted in the following insights:

1. The data suggests that the “save as” operation is largely misunderstood by users. The large savings that we show from garbage collection suggest that users do not understand the implications of *fast-save* mode (the default), instead believing the “save as” operation to be a way to create a copy of the document.

2. The lack of built-in support for compression in OLE archives has forced designers to implement ad-hoc solutions to achieve high performance. This experience suggests that a compression feature would be a desirable addition to OLE archives.

3. OLE archive formats are likely to remain the preferred intermediate format for Office documents, while the XML-based format will likely be the format of choice for Web publishing. The XML-based format has the advantage that it can more easily be interpreted by application other than Office (*e.g.*, Web browsers). It is also amenable to widespread browser techniques that improve user perceived latency, such as incremental rendering and fetch on-demand. On the flip side, the current imple-

mentation of Office 2000 does not implement incremental loading or writing of XML-based documents, leading to higher latencies for opening and storing XML-based documents than those experienced on similar OLE archive documents. Moreover, some of the Office formats do not yet have XML equivalents.

References

[1] Alta Vista home page. http://www.altavista.com.

[2] Arlitt, M. F., and Williamson, C. L. Server workload characterization: the search for invariants. In *ACM SIGMETRICS Conference* (Philadelphia, Pennsylvania, 1996).

[3] Bagrodia, R., Chu, W. W., Kleinrock, L., and Popek, G. Vision, issues, and architecture for nomadic computing. *IEEE Personal Communications 2*, 6 (Dec. 1995), 14–27.

[4] Bray, T. Measuring the Web. In *The World Wide Web Journal* (1996), vol. 1-3.

[5] Brockschmidt, K. *Inside OLE*. Microsoft Press, 1995.

[6] Chappell, D. *Understanding ActiveX and OLE*. Microsoft Press, 1996.

[7] Cunha, C. R., Bestavros, A., and Crovella, M. E. Characteristics of WWW client-based traces. Tech. Rep. TR-95-010, Boston University, Apr. 1995.

[8] Deutsch, P., and Gailly, J. L. ZLIB compressed data format specification version 3.3. `http://src.doc.ic.ac.uk/Mirrors/ftp.cdrom.com/pub/infozip/doc/rfc1950.txt`, May 1996.

[9] Fox, A., Gribble, S. D., Chawathe, Y., and Brewer, E. A. Adapting to network and client variation using infrastructural proxies: Lessons and perspectives. *IEEE Personal Communications 5*, 4 (Aug. 1998), 10–19.

[10] Joseph, A. D., deLespinasse, A. F., Tauber, J. A., Gifford, D. K., and Kaashoek, M. F. Rover: a toolkit for mobile information access. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP '95)* (Copper Mountain Resort, Colorado, Dec. 1995), pp. 156–171.

[11] Katz, R. H. Adaptation and mobility in wireless information systems. *IEEE Personal Communications 1*, 1 (1994), 6–17.

[12] Kistler, J. J., and Satyanarayanan, M. Disconnected operation in the Coda file system. *ACM Transactions on Computer Systems 10*, 1 (Feb. 1992), 3–25.

[13] Microsoft Corporation. *Microsoft Excel File Format*. Redmond, Washington, 1997. MSDN Online, `http://msdn.microsoft.com`.

[14] Microsoft Corporation. *Microsoft PowerPoint File Format*. Redmond, Washington, 1997. MSDN Online, `http://msdn.microsoft.com`.

[15] Microsoft Corporation. *Microsoft Word File Format*. Redmond, Washington, 1997. MSDN Online, `http://msdn.microsoft.com`.

[16] Microsoft Press. *Microsoft Office 97 / Visual Basic Programmer's Guide*, 1997.

[17] Microsoft Press. *Microsoft Office 2000 / Visual Basic Programmer's Guide*, 1999.

[18] Mummert, L. B., Ebling, M. R., and Satyanarayanan, M. Exploiting weak connectivity for mobile file access. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles* (Copper Mountain Resort, Colorado, Dec. 1995).

[19] Niksic, H. Gnu Wget. `http://www.gnu.org/manual/wget/ps/wget.ps`, Sept. 1998.

[20] Noble, B. D., Satyanarayanan, M., Narayanan, D., Tilton, J. E., Flinn, J., and Walker, K. R. Agile application-aware adaptation for mobility. *Operating Systems Review (ACM) 51*, 5 (Dec. 1997), 276–287.

[21] Pitkow, J. E. Summary of WWW characterizations. In *Proceedings of the Seventh International World Wide Web Conference* (Brisbane, Australia, Apr. 1998).

[22] Satyanarayanan, M. Hot topics: Mobile computing. *IEEE Computer 26*, 9 (Sept. 1993), 81–82.

[23] Sedayao, J. "Mosaic will kill my network!" - studying network traffic patterns of Mosaic use. In *Proc. of the 2nd International WWW Conference* (Chicago, Illinois, 1994).

[24] Woodruff, A., Aoki, P. M., Brewer, E., Gauthier, P., and Rowe, L. A. An investigation of documents from the World Wide Web. In *The World Wide Web Journal* (1996), vol. 1-3.