Speculative Document Evaluation

Alexander J. Macdonald
Document Engineering Lab
School of Computer Science
University of Nottingham
Nottingham, NG8 1BB, UK
ajm@cs.nott.ac.uk

David F. Brailsford
Document Engineering Lab
School of Computer Science
University of Nottingham
Nottingham, NG8 1BB, UK
dfb@cs.nott.ac.uk

Steven R. Bagley
Document Engineering Lab
School of Computer Science
University of Nottingham
Nottingham, NG8 1BB, UK
srb@cs.nott.ac.uk

John Lumley
HP Labs
Filton Road, Stoke Gifford
Bristol, BS34 8QZ, UK
john.lumley@hp.com

Abstract

Optimisation of real world Variable Data printing (VDP) documents is a difficult problem because the interdependencies between layout functions may drastically reduce the number of invariant blocks that can be factored out for pre-rasterisation.

This paper examines how speculative evaluation at an early stage in a document-preparation pipeline, provides a generic and effective method of optimising VDP documents that contain such interdependencies.

Speculative evaluation will be at its most effective in speeding up print runs if sets of layout invariances can either be discovered automatically, or designed into the document at an early stage. In either case the expertise of the layout designer needs to be supplemented by expertise in exploiting potential invariances and also in predicting the effects of speculative evaluation on the caches used at various stages in the print production pipeline.

Categories and Subject Descriptors

E.1 [Data]: Data Structures Trees; I.7.2 [Document and Text Processing]: Document Preparation Markup Languages; I.7.4 [Document and Text Processing]: Electronic Publishing.

General Terms

Algorithms, Documentation

Keywords

XML, XSLT, SVG, Speculative Evaluation, Document Layout, Optimisation

1. INTRODUCTION

The layout of pages in a VDP[1] print run is commonly described using the PPML[2] language. PPML is an XML-based markup which enables document objects on a page to be identified just prior to rasterisation. Raster Image Processing (RIP) software that takes account of a PPML script is an example of a ‘PPML consumer’. PPML requires its chunks of rasterisable content to be placed inside demarcated sections and this rasterisable content can be anything that the PPML consumer can cope with: examples might include PostScript, TIFF bitmap, PDF or SVG. In the remainder of this paper we assume, without loss of generality, that the final pre-raster form will be SVG. We envisage a sequence of single-page, variable data, advertising flyers prepared by extracting information from a database and flowing that information into a previously prepared layout template. The output of this document preparation stage (which we wish to optimise) will be a sequence of PPML-encoded pages passed down a pipeline to the PPML consumer.

In PPML, those page objects that are invariant (except, perhaps, for translations within the page) can be flagged as `Reusable Objects` and kept in a raster cache for re-use but it can reasonably be asked, at the outset, whether such caching is strictly necessary. Given the speed of modern computers it might seem feasible to render every new page in its entirety and not to waste effort and ingenuity on reusing invariant pre-rasterised material. Indeed, the simplest examples of variable data printing, such as customised black-and-white letters created by mail merge, will very probably be output on a laser printer which renders each page afresh and which does not use PPML. However, when we turn to multi-colour high-end VDP presses, such as the HP Indigo, we have a constant printing time of one 4-colour A4 page every half second. With only half a second to pull information from a database, evaluate all constraints and flows, paginate the document and then rasterise, even a powerful computer [3] will struggle to keep up as the document complexity increases. The message is that ‘fast’ is seldom fast enough and that all optimisation is to be welcomed.

Software for VDP needs to concentrate on identifying invariant objects at an early point in the document preparation workflow so that they can be marked as being reusable. This information is then used later in the pipeline by the PPML consumer. Equally important is the need to identify opportunities for workflow optimisations by using speculative evaluation and caching during the earlier process of actually preparing the PPML pages i.e. well before rasterisation takes place.

The work described here follows on from a previous paper[4] which showed how a template document built from nested layout functions could be statically optimised so that evaluation occurred more quickly. This was done by identifying, and isolating, invariant sections in individual layout functions to ensure they were evaluated just once.

This approach was shown to work well for simple linear flow functions, where bounding box dimensions of objects were demonstrably unaffected by objects rasterised later on the page. However, real-world documents tend to be built from much more

FINAL DRAFT of short paper submitted to:
DocEng’07, August 29–31, 2007, Winnipeg, Canada.

complex layout constraints (perhaps involving interdependent x and y flows) where a change in a given item's bounds may cause a profound change in the bounds of objects elsewhere on the page.

2. INTERDEPENDENCIES

To illustrate the problems faced with VDP documents we examine a hypothetical, university marketing flyer that will be sent to all students once they have been given a place on a specific course at the University of Nottingham. The flyer is to be personalised for each student, using material about the degree course, and the student, drawn from a database.

The flyer consists of three main blocks of content: a header, with the text "Welcome to the University of Nottingham", the main central block of text with a standard, text-invariant, welcoming message containing general details about the chosen degree course and, finally, an "info block" giving information specific to each student. A simplified version of the flyer is shown in figure 1.

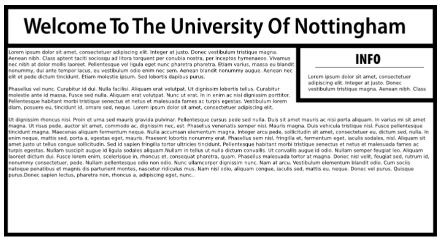

Figure 1: University marketing flyer.

The only element of the document with variable content is the info block which in figure 1 is a fixed-width, variable-height, box anchored at the top right of the layout. This box may contain information about the hall of residence to which the student has been allocated, the name of the student's personal tutor, and so on.

It is important to realise that although the content of both the header and the main body of text are invariant, the variable height info box, alongside the main body, may well cause the *appearance* of the body text to be variable (due to reflow) and therefore not cacheable, across the print run.

In the next section we investigate whether, despite the reflow of the body text, it might still be possible to identify raster-invariant sections of the text.

3. LAYOUT FUNCTION OPTIMISATION

If the info box has a known minimum height then all of the text in the main body that flows down to this height will be laid out identically across all the print runs. Figures 2a and 2b show two different instances of the flyer with the invariant area of the body text (as guaranteed by the minimum height info box constraint) shaded.

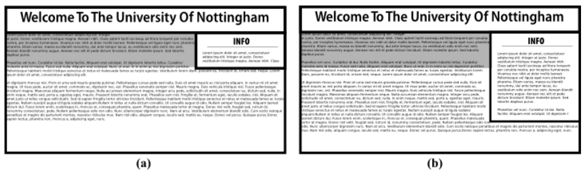
Welcome To The University Of Nottingham
INFO
(a)
Welcome To The University Of Nottingham
INFO
(b)

Figure 2: Two instances of the flyer.

With a text flow function that contains this optimisation a partial evaluation of the template document, as in figure 2a, causes the first part of the body text to be evaluated and turned into a static block of SVG. The remainder of the body text will now reflow underneath the info box starting at a new vertical offset equal to the info box's minimum height constraint.

The amount of text that has been factored out, and made raster-invariant, in this example is small but still worthwhile. If the info box remains fixed in width but increases in height then figure 2b indicates that further blocks of main body text become subject to the text width constraint imposed by the info box's presence. So, if the info box has a discrete set of just a few possible heights then a bigger range of invariant textual sub-blocks can be identified and pre-computed.

However, let us now assume that the designer of the flyer wants a completely new layout to be triggered under certain conditions (e.g the info box exceeds a given height, or the body text is for a degree course that deserves a more exotic layout). If these conditions are satisfied the designer makes what might seem a relatively small stylistic change, namely, centering the info box in the middle of the page and allowing it to expand outward in all directions. Unfortunately this change has the effect of causing such a complete reflow of the main body text that the previously discussed optimisations become impossible (see figure 3).

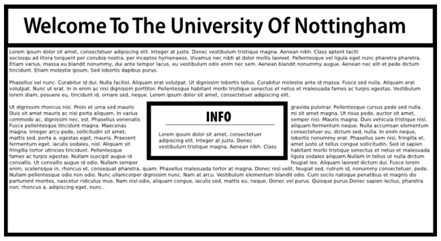

Figure 3: Flyer with centred floating box.

4. SPECULATIVE EVALUATION

We have just seen that the figure 3 variant for our flyer offers no scope for optimisation apart from the header block. The key issue here is how frequently in the print run does this variant layout occur as opposed to the more optimisable one with a right-anchored, fixed-width, variable-height, info boxes as shown in figure 2.

If it is possible to predicate the sizes of certain items (e.g. the info box in our example) within a VDP document then a speculative evaluation can be performed by binding in dummy items of the predicted sizes and evaluating the document as normal. A new conditional document is then created that contains two branches, one with the original document (e.g. no predictions whatever made about info box height) and another with an optimised version (e.g. minimum-height info box will be sufficient). The branch test will then establish if the real data fits into the speculated optimal layout and will then use that branch if the test succeeds.

5. OPTIMISING THE PRODUCTION OF THE UNIVERSITY FLYER

In the example of the marketing flyer a speculative evaluation may be able to provide considerable scope for optimisation. The marketing department will know from previous years what percentages of the students are staying in each hall of residence, how many of them are local or overseas students. This information needs to be flowed into the info box and we assume that variabilities in the student's details, or those of the hall, mean that line-break decisions for this flow cannot be predicted ahead of time. However it may well turn out that that the personalised information in the fixed-width info box can be made to fit, elegantly, into one of just a few allowed box heights. For the sake of illustration let us assume there are three such allowed heights.

By doing a speculative evaluation of this document with the three possible dummy values of heights for the info box it will be possible to rasterise, and therefore cache, the three resulting versions of the main body text. Again, we assume that the main body text, although fixed for any given undergraduate degree course, is the most expensive part of this document to rasterise because of the quantity of text and its reflow variability

Under these conditions the assembly of a PPML page simply involves the addition of the rasterised text for the info box into one of three templates having the correct info box height. Previously prepared invariant text blocks can be used for the main body text and even the portion of it flowing under the allowed heights of info box becomes a known quantity and can therefore be pre-rasterised.

The conditional tests introduced by choosing among the three speculatively evaluated page templates, once info box height is known, are unlikely to add any significant time to overall document evaluation. Nevertheless the different branches and their differing rasterised blocks may well require more disk space or RAM. Considerations of this sort will influence how many differing layouts it is worthwhile to partially evaluate.

Let us now imagine a situation where the layout shown in figure 3 is triggered if (i) the degree course in question is computer science or (ii) for any other degree course in cases where the info box height exceeds the maximum of the three allowed discrete heights. In case (i) we can presume that the boolean test on the database input tells us very quickly indeed that the main body text relates to computer science. If this is so there is no point in trying to flow info box text into the possibilities of figures 2a and 2b; one has to be resigned at the outset to the non-optimisable layout of figure 3. In case (ii) the simplest strategy is to wait and see if the rendered info box text overflows the largest allowed height for that box and, if it does, to then start up the layout processing of figure 3. On the other hand if multi-threading does not slow down the overall processing it is possible to contemplate starting a separate thread for figure 3 layout, in parallel with those trying the figure 2 layout.

6. ANALYSIS

The benefit of a truly programmatic variable document comes from the freedom it offers the document authors; it does not force them to constrain the variable parts of the document to fit into restricted 'copy holes'. This, in turn, leaves the authors free to make documents that show the data in the best possible no-compromise layout.

VDP packages such as DDF[5] offer just such complete document variability but with potentially no cacheability and with slow document evaluation. Copy-hole techniques are at the other end of the spectrum of VDP possibilities. They offer higher rasterisation speed and cacheability but with reduced scope for variability.

Our flyer case study shows that speculative evaluation can be used to automatically create fast paths in a truly variable document whenever the most frequently occurring combinations can resolve into a choice of copy-hole documents. Provided the document still retains an option for a fully variable path the method we have outlined takes advantage of copy-hole speeds while retaining the flexibility of a truly programmatic approach.

7. CONCLUSION AND FUTURE WORK

Speculation is a generic optimisation technique that is employed to great effect in other aspects of computer science and can also benefit VDP print runs. The kinds of document that will benefit the most are those that have predictable data, following patterns which allow accurate speculations to be made on the bounds of the various items in the document.

A key question is the nature of the process for discovering and exploiting possible optimisations, such as the three copy hole possibilities in our flyer. It seems certain that this burden cannot be put solely on the graphic artist (GA) who designs the flyer layout in terms of also requiring him or her to be an expert programmer[1]. To what extent the optimisation possibilities need early programmatic help and to what extent they can be 'discovered' remains a topic for investigation.

Our work plan involves examining to what extent multiple branches affects the speed of evaluation, and whether the increased space required to create the document in this way outweighs the performance gains. Part of the on-going work will involve collecting case studies and data sets for real-world print runs to help us understand which optimisations actually benefit the kind of documents encountered in VDP jobs.

8. ACKNOWLEDGEMENTS

Thanks are due to Hewlett Packard (UK) and EPSRC for supporting Alexander Macdonald's PhD studentship.

9. REFERENCES

[1] Fabio Giannetti and Royston Sellman. Anvil: VDP Segmented Workflow Toolset. In *HP Labs Technical Report HPL-2007-18*, February 2007.

[2] D. DeBronkart and P. Davis. PPML (Personalized Print Markup Language): a new XML-based industry standard print language. In *XML Europe 2000*, pages 1–14, June 2000.

[3] Felipe Rech Meneguzzi, Leonardo Luceiro Meirelles, Fernando Tarlá Martins Mano, João Batista S. de Oliveira, and Ana Cristina Benso da Silva. Strategies for document optimization in digital publishing. In *ACM Symposium on Document Engineering*, pages 163–170, 2004.

[4] Alexander J. Macdonald, David F. Brailsford, and John Lumley. Evaluating Invariances in Document Layout Functions. In *Proceedings of the 2006 ACM symposium on Document engineering*, pages 25–27. ACM Press, October 2006.

[5] John Lumley, Roger Gimson, and Owen Rees. Extensible Layout in Functional Documents. In *SPIE/EI 2006 Digital Publishing Conference*, January 2006.