Efficient data representation for XML in peer-based systems

Brian Tripney, Christopher Foley, Richard Gourlay, John Wilson
Department of Computer and Information Sciences
University of Strathclyde, Glasgow, UK

Abstract

Purpose – New directions in the provision of end-user computing experiences mean that we need to determine the best way to share data between small mobile computing devices. Partitioning large structures so that they can be shared efficiently provides a basis for data-intensive applications on such platforms. The partitioned structure can be compressed using dictionary-based approaches and then directly queried without firstly decompressing the whole structure.

Design/methodology/approach
We describe an architecture for partitioning XML into structural and dictionary elements and the subsequent manipulation of the dictionary elements to make the best use of available space.

Findings
The results indicate that considerable savings are available by removing duplicate dictionaries. We also identify the most effective strategy for defining dictionary scope.

Research limitations/implications
Our evaluation is based on a range of benchmark XML structures and the approach to minimising dictionary size shows benefit in the majority of these. Where structures are small and regular, the benefits of efficient dictionary representation are lost. Our future research now focuses on heuristics for further partitioning of structural elements.

Practical implications
Mobile applications that need access to large data collections will benefit from the findings of this research. Traditional client/server architectures are not suited to dealing with high volume demands from a multitude of small mobile devices. Peer data sharing provides a more scalable solution and the experiments that we describe demonstrate the most effective way of sharing data in this context.

Social implications
Many services are available via smartphone devices but users are wary of exploiting the full potential because of the need to conserve battery power. Our approach mitigates this challenge and consequently expands the potential for users to benefit from mobile information systems. This will have impact in areas such as advertising, entertainment and education but will depend on the acceptability of file sharing being extended from the desktop to the mobile environment.

Originality/value
The original work we have done characterises the most effective way of sharing large data sets between small mobile devices. This will save battery power on devices such as smartphones, thus providing benefits to users of such devices.

Keywords Peer-to-peer, Data compression, Database management, Extensible Markup Language
Paper type Research paper

1. Introduction

Growth in personal computer sales dropped almost to zero in 2008. Meanwhile two hundred million smartphones were sold, an increase of 13% on the previous year (Meyer, 2009). Although smartphone sales are set to exceed personal computer sales, 2008 also saw the purchase of 1.22 billion mobile phones. Given the overlap of functionality between these devices, there appears still to be a significant potential expansion available in the market for smartphone devices. Smartphone users are becoming accustomed to their needs for data being satisfied on demand, thereby presenting a wealth of fresh challenges for computer science. In situations where a plethora of small devices are operating in an infrastructure-light environment it is more effective to share information between local mobile peers and only involve a central server as necessary. Location awareness can be used to ensure that the right information is available at the right time.

This paper describes an architecture for sharing large collections of semistructured data between many small mobile devices with the aim of reducing their dependence on fixed server infrastructure. It also reports our implementation and evaluation of this architecture. We expect that our work will contribute to an understanding of query processing and data management in XML data models particularly in the framework of location and context-aware applications and services.

We start with a definition of the problem addressed and show how research in peer-to-peer database systems (PDBS), mobile and *ad hoc* networks (MANETs) and semistructured data provides a basis for our approach. We present the methodology of our design and results that illustrate the potential of this architecture. Finally we characterise the research and development opportunities that are made possible by our investigation.

For many years, client-server models have dominated the sharing of data in Internet contexts. More recently, peer-to-peer (P2P) data exchange has become a widely used method of resolving the issues presented by server overload. Whilst this helps to solve the problem of inadequate server infrastructure, typical conceptions of P2P topologies are based on the dynamic creation of *ad hoc* networks in a wired world. Performance of such systems is always a concern but given a balance between the number of uploading and downloading peers and the available bandwidth, adequate response times can usually be supported. Since peers in this scenario are typically desktop or laptop computers, the location of the user is not significant in the context of the data required.

We have already noted the remarkable growth in the number and power of small mobile computing devices and the way that the expectations of the user population have similarly expanded. Tourism, advertising and entertainment provide large-scale application areas where users need access to location sensitive data. Users increasingly expect instant access to information on the move, however storage restrictions imposed by limited capability mobile devices prevent the preloading of large data collections. On the other hand, partial disconnection intermittently prevents loading the data from a server ‘on the fly’. In this context, neither the client-server model nor the conventional P2P sharing are efficient enough to support data intensive applications where large numbers of users demand access to large data collections.

We address this dilemma by using P2P distribution schemes to share compressed, partitioned XML. We arrange that queries over this data can be resolved without first fully decompressing the complete structure. This extends knowledge in the area of PDBS in the context of small mobile devices. We characterise the most efficient way of propagating the physical representation of such data to peer devices that do not already possess it, an issue that has been neglected in the past. Without establishing this, succeeding generations of smartphones will continue to struggle with data intensive applications.

We assume a partial disconnection model dominated by disconnection periods and a class of applications in which localisation defines the parts of a data collection of greatest interest. Shoppers wanting to see product promotions whilst on a busy shopping street represent such a scenario. Fragmented localised data is propagated between shoppers' phones or pulled if the shopper requested data not already present. Fragments of the data are locally autonomous and the assembly is managed by federation, thereby avoiding the need for communication with a centralised directory structure. During busy periods, the database flows within the phone population but is never wholly resident on a single device. If there are few shoppers on the street, those who want the data have to wait for a server connection or pay for data via GSM. Although available bandwidth will grow in future, an approach such as ours will be necessary to limit the effects of network congestion. Compression will also save CPU cycles and consequently battery power; a significant future limiting factor for small mobile devices.

Our approach to partitioning is an essential component of this scenario since small partitions and their associated dictionaries can be propagated efficiently between phones and queried without complete decompression. The decompression load then becomes proportional to the size of the result set. The partitions are logically homogeneous so there is a strong likelihood that user queries over data relevant to a particular location can be satisfied without recourse to pulling additional data from other phones, making this approach the most effective way of servicing shoppers' needs during disconnection periods. Queries that span partitions are more expensive but processing the incoming partition as a stream will improve the effective performance of the system. The technique integrates with raw textual and multimedia data. Query results over partitioned and compressed data can be provided directly where predicate values are contained in the partitioned structures or indirectly where reference needs to be made to underlying multimedia files.

2. Related Work

Peer-to-peer database systems (PDBS) (Bonifati *et al.*, 2008) have attracted significant recent attention and have typically focused on the use of distributed hash tables (Balakrishnan *et al.*, 2003). Algebraic optimisations for managing distributed XML data structures have also been proposed (Koloniari and Pitoura, 2005). There is recognition that a variety of approaches may be necessary to exploit various communication architectures (Kangasharju and Tarkoma, 2007). Simulation of client-server and peer-to-peer networks suggest that both routes can provide similar end-user download experience but that scaling in client-server systems can only be addressed

by the additional expense of adding more servers and providing appropriate management for these systems (Leibnitz *et al.*, 2007). Aygün *et al.* (2009) propose a multilevel architecture for conceptualising P2P data management. Under this architecture, a storage module is responsible for handling the distributed elements of the system, whilst an indexing module handles the routing of queries within the system.

Early work on different kinds of index structures for XML focused on query optimisation for the Lore system (McHugh and Widom, 1999). The main thrust of this work was the development of heuristics that determine when to use each of the four specific forms of indices (value, text, link and path index) provided by their experimental base. Buneman and co-workers (2003) combine the XMill (Liefke and Suciu, 2000) approach for compact representation of atomic data with the approach for skeleton compression by sharing sub-trees to address XML join queries. Their fundamental assumption is that the skeleton of typical XML documents is small and thus can be kept in memory. The actual data is only used in the last stage of their join algorithm, avoiding unnecessary I/O operations. Kaushik *et al.* (2004) extend their original work (Kaushik *et al.*, 2002) on structural indices for path expressions to include keyword constraints on the contained atomic data. They propose a general strategy to combine structural indices with inverted lists in order to address this class of queries efficiently. Other authors have explored the compression of prior computed query results in sharing complete XML structures (Natchetoi *et al.*, 2007). Dictionary-based approaches to compressing XML typically focus on tag compression (Arion *et al.*, 2007). Alternative approaches involve the use of schema information to drive the compression (Bőttcher *et al.*, 2007). In earlier work, we investigated the use of dictionary compression techniques for representing both tags and values in XML structures (Neumüller *et al.*, 2003; Gourlay *et al.*, 2007) and other investigators have also examined the decomposition of XML into array-based structures (Buneman *et al.*, 2005).

Schemes that use bitmap or entropy coding are known to improve the processing of relational data (Cockshott *et al.*, 1998; Stonebraker *et al.*, 2005). Dictionary compression can be satisfactorily combined with structural indexing by using a hybrid data structure (Wilson *et al.*, 2006) and has been used with mobile client systems that refresh themselves from a central server (Natchetoi *et al.*, 2007). The need to process large structures that may not fit into main memory has led to the development of interest in XML stream processing (Schneider, 2003) and there is evidence that compressed streams of XML are accessible to query processing (Bőttcher and Steinmetz, 2007). Energy usage, communication bandwidth and storage have been identified as the main parameters affecting this scenario (Wolfson *et al.*, 2007). Peer-to-peer systems eschew central control in favour of a model in which functionally equivalent elements operate in a distributed environment. This provides benefits such as permanence, anonymity and search capability however, the basic operation is the location of data elements (Wehrle & Steinmetz, 2005). Whilst there are many open research questions in the domain of peer-based systems, the autonomy that they typically provide potentially reduces the necessity for management overhead (Daswani *et al.*, 2003).

Our contribution is the adaptation of bisimilarity indexing techniques to the generation of partitioned XML structures and the use of dictionary compression methods to render the most efficient representations of such data. This presents particular benefits for small mobile devices that are sharing data in peer-based environments.

3. Model

A simplified analytical model (Qiu and Srikant, 2004) can be used to assess the impact of data compression in peer-based architectures. In such a system, there will inevitably be users who will share the data elements they possess and those for whom the technical or economic benefits that are available as a consequence of sharing are insufficient to persuade them to take part (leeches). The BitTorrrent model of incentivization promotes sharing by down-grading the service to those who don't share and improving service to those who are willing to host software that will perform P2P sharing. Incentivization can also be provided by using micropayment models to offset the potential for increased call charges. However, as with all P2P-based systems, leeches are likely to be a persistent feature of the environment. The scenario involves devices that upload only (seeds), download only (leeches) and both upload and download (peers). The relationships between these kinds of devices are shown in Figure 1.

Each queue consists of a list of tasks and a server (u - upload, d - download). The queues form a closed queuing network i.e. the system has no source or sink. The model is simplified by assuming that all peers have equal capacity both to upload and download and that in a steady state, the number of uploads and downloads is independent of time. The model incorporates random churn of the peer pool via the off line rate (κ) and the abort rate (ρ). These are expressed as a rate correction to the upload rate (μ) and download rate (τ) respectively. The model is represented by the expressions shown in Figure 1. Little's Law (Little, 1961) is used to produce the service time (t) experienced by users waiting for the completion of downloads (3). Compressing data provides leverage for both upload and download and is incorporated into the model by the coefficient δ.

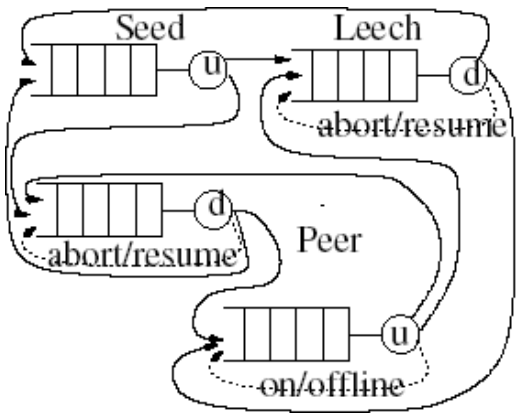

Figure 1.
Device relationships

ι	effective upload rate
β	selfish rate
μ	nominal upload rate
κ	off line rate
θ	bottleneck bandwidth
τ	download bandwidth usage by each peer
ρ	abort rate
δ	compression ratio
t	service time

$$\frac{1}{\iota} = \frac{1}{1-\beta}\left(\frac{1}{\mu} - \frac{1}{\kappa}\right) \quad (1)$$

$$\frac{1}{\theta} = \max\left(\frac{1}{\tau}, \frac{1}{\iota}\right)\delta \quad (2)$$

$$t = \frac{1}{\theta + \rho} \quad (3)$$

Figure 2. Queuing model for P2P data sharing

Assuming a nominal upload bandwidth (μ) of 12Mbits/sec, off line rate (κ) of 50 Mbits/sec, individual peer download bandwidth usage (τ) of 20 Mbits/sec and an abort rate (ρ) of 10 Mbits/sec, the analytical model produces estimates of service time shown in Figure 3. It can be seen that an increasing selfish rate increases the expected download time for other peers but that this is considerably mitigated by the effect of compression. Similarly, at a fixed selfish rate, the effective download bandwidth is improved by increasing compression.

This model predicts that compressing data improves service to users irrespective of the number of leeches in the system. In addition to these beneficial effects of compression, the effect of partitioning data structures so that only limited relevant data is sent between peers, results in additional savings. This further reduction will provide more efficient use of bandwidth and better resilience to high selfish rates than is predicted by the analytical model. Since the number of available peers varies in inverse proportion to the number of leeches, the model also suggests that peer pool variations will be mitigated by compressing data.

Compressing data provides a useful step in empowering applications running on phones to rapidly receive large volumes of data in response to requests from users. Efficient processing of XML is helped by choosing the right physical data structures and supporting them with appropriate indexing techniques. Bisimilarity (i.e. the sharing of common subtrees) allows resolution of path location steps in linear time (Kaushik *et al.*, 2004). A family of indexes ((j,k)-F+B-index) can be constructed using a range of values for forward or backward bisimilarity. Embedding the structural elements of queries into such an index graph can be done using the same algorithms that could be used to embed them into a data graph. The structural elements of the query can be resolved against the index graph but the original data graph needs to be maintained in order to resolve value predicates.

The approach we have developed (NSGraph) (Wilson *et al.*, 2006) constructs atomic data dictionaries according to the structural groupings. Consequently the part of the dictionary corresponding to a structural grouping can be incorporated into the node of the index graph representing it. The vertex identifiers used in both the dictionaries and the index graph can be replaced with the entries based on a numbering scheme, creating a unique address space for validation purposes. This approach produces a hybrid that represents the cross-product of an index graph with a signature tree where its leaf nodes are replaced by domain dictionaries. Figure 4 illustrates this using the (1,1)-F+B-index graph of the example DataGraph on the left. This approach allows queries to be resolved directly on the compressed data structure with only the returned values being decompressed. In broad terms, the utility of bisimilarity as a means of partitioning data is that elements are grouped into logically coherent sets. For example, from Figure 4, the (1,1)-F+B-index will group all *name* elements together since they have a common ancestor at one level higher up the tree (*person*) and a common descendent (a data node). In a (1,2)-F+B-index this group would be split because of the incoming edge from *editor* which is within the span of two backward nodes.

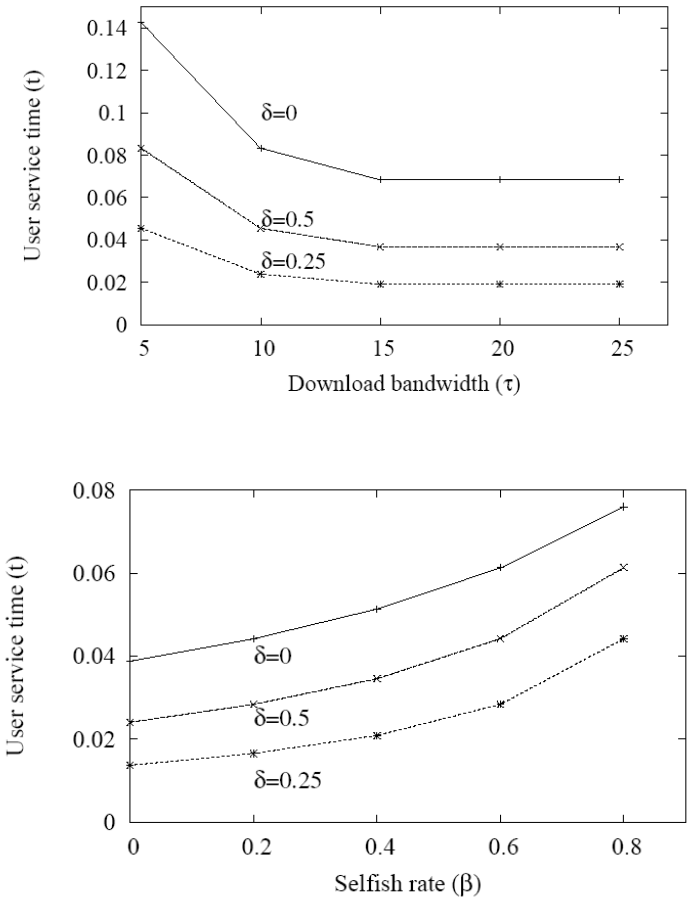

Figure 3.
Performance characteristics of the queuing model

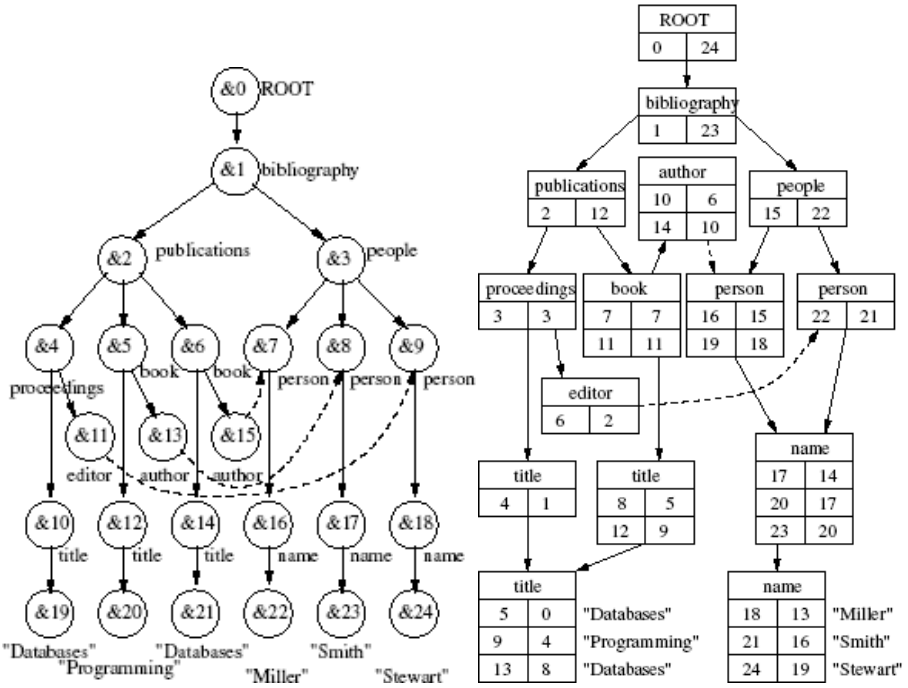

Figure 4. DataGraph and NSGraph

Memory-boundness is a significant limitation of NSGraph. Our initial implementation requires the complete graph and the associated leaf values to be present in memory. Dictionary compression and fragmentation of the compressed XML tree reduces the size of the data structure that needs to be held in memory. The NSIndex extension builds on the NSGraph model by supporting non-volatile storage of the resulting structure and providing for its direct interrogation.

This approach gives the advantage of not having to parse and store the entire data structure in memory in order to evaluate a query, as well as giving opportunities for optimisations of the data structure.

4. System Overview

The system architecture of NSIndex is split into three major modules shown in Figure 5. The NSGraph module processes the underlying source XML data structure generating a structural summarisation and associated data representation. This model of the source is fed into NSStore where the memory-based structure is mapped to the non-volatile structure seen in the lower part of Figure 5. The NSQuery system can be used to directly read the non-volatile structure and allow for the evaluation of arbitrary queries.

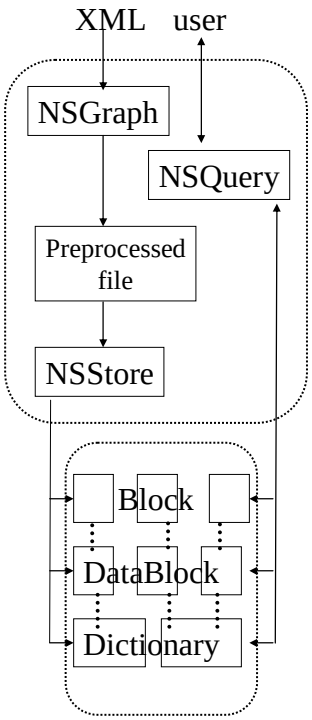

Figure 5.
NSIndex component architecture

The source data structure is processed by the NSGraph module, first building a DataGraph type structure and then applying partitioning based on the concept of bisimilarity. Elements are grouped according to their bisimilar properties and held within the multi-element vectors of the NSGraph structure. A numbering scheme is applied to retain the information previously recorded by the DataGraph edges.

To facilitate non-volatile storage of the structural summarisation, the NSStore component maps the vectors to a block-based storage scheme. The record for each entry within a Block consists of its pre-order identifier within the numbering scheme, its post-order identifier, the level (or depth) in the graph structure, and the size of the entry (including its subordinates).

A specialised type of Block, a DataBlock, is a container for data elements, each represented as pre-order, post-order, level and size, together with a token to indicate the raw data value contained in the appropriate dictionary. All data elements in one DataBlock will use the same dictionary.

Additional processing can be used to reduce redundancy in the set of data dictionaries produced by NSIndex. Two methods are currently employed to thin out the dictionaries. First, any exact duplicate dictionaries are removed, updating the dictionary references at a DataBlock level to point to the remaining copy. In the example shown in Figure 6, Dictionary 3 is removed as it is an exact duplicate of Dictionary 1. DataBlock C (which previously referred to Dictionary 1) is then updated to make use of Dictionary 1.

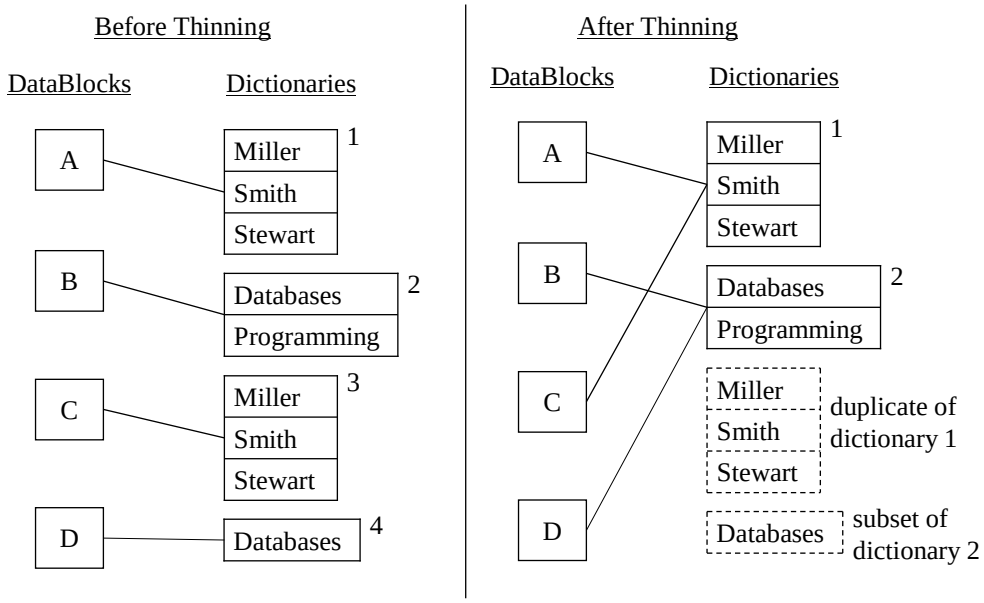

Figure 6. Dictionary thinning process

The second method searches for subset dictionaries – those whose data values are wholly contained within another dictionary of the same token size. Where a subset is found, any references to that dictionary are replaced by references to the superset dictionary, and the data tokens of each element within the affected DataBlock are updated to reflect the new dictionary used. This is shown in the example by the removal of Dictionary 4 and the update of DataBlock D to use Dictionary 2.

Following this thinning process, any individual dictionary may be used by a large number of DataBlocks. This results in significant savings in the number of dictionaries required.

5. Experimental Evaluation

Initial work on NSGraph (Wilson *et al.*, 2006) examined the effects of applying different partitioning schemes to a number of datasets. Each file was partitioned using varying levels of forward and backward bisimilarity and the number of vertices and edges in the resulting NSGraph variations were counted.

To evaluate the architecture we chose benchmark datasets that represent both randomly generated and real world data. In addition we selected structures that contain both regular and irregular branching patterns. The datasets used are summarised in Table 1.

	Regular	Irregular
Random	Orders (a subset of the TPC-H benchmark).	XMark benchmark (10Mb and 30Mb).
Real world	Legal (conviction details from a sentencing information system).	NASA (astronomical data), Medline (a 20Mb section of medical bibliographic database).

Table 1. Summary of benchmark datasets.

	DataGraph	(0,0)F+B	(0,1)F+B	(1,0)F+B	(1,1)F+B	Dataset
vertices	920678	44	83	101	3151	Legal
edges	920677	82	82	1656	3150	Legal
vertices	285004	14	23	14	23	Orders
edges	285003	22	22	22	22	Orders
vertices	951681	76	217	986	90403	NASA
edges	951680	159	216	9572	90402	NASA
vertices	997830	86	118	1112	77255	Medline
edges	997829	160	187	11719	77254	Medline
vertices	319741	78	933	7091	122964	XMark10
edges	319740	150	932	34237	122963	XMark10
vertices	1010413	79	993	15258	318184	XMark30
edges	1010412	159	992	80240	318183	XMark30

Table 2. Effects of structural summarisation

Table 2 shows the results for selected partitioning schemes. Comparison between the DataGraph and (F+B) index graph for each dataset shows that the vertex and edge count increases with increasing levels of bisimilarity. Despite this, the counts for the index graphs are less than those for the DataGraph at these levels of bisimilarity. The extent of the structural summarisation indicates that compression is available by limiting storage only to values that are characteristic of each node.

Variation in the level of forward and backward partitioning affects the number of dictionaries produced, since each dictionary represents a single vertex. The datasets were partitioned in sixteen ways using a version of the NSGraph program - these ranged from (0,0)-F+B (0-forward and 0-back, partitioning solely on data label) to (3,3)-F+B. This produced a set of uncompressed DataBlocks for each partitioning scheme.

The unique values from each DataBlock were then extracted and used to calculate the size of data and dictionaries under a minimal-bit token scheme. For each dataset, the total size of uncompressed blocks is unaffected by the partitioning scheme used (as the complete set of blocks will contain all the data values regardless of how these are distributed across blocks). However, the compressed data size is affected by the distribution of data values, as the success of the minimal-bit scheme relies upon the repetition of values within individual blocks.

For all but one of the datasets tested, compression was improved by increasing backward bisimilarity from (0,0)-F+B to (0,1)-F+B. The graph in Figure 7 shows that

in the best case an additional 15% compression was obtained over the Legal dataset. In the worst case, XMark30 showed an adverse effect of 6% as a result of the change. This is a result of the pseudo-random nature of the text within the XMark datasets, which leads to little repetition within the DataBlocks.

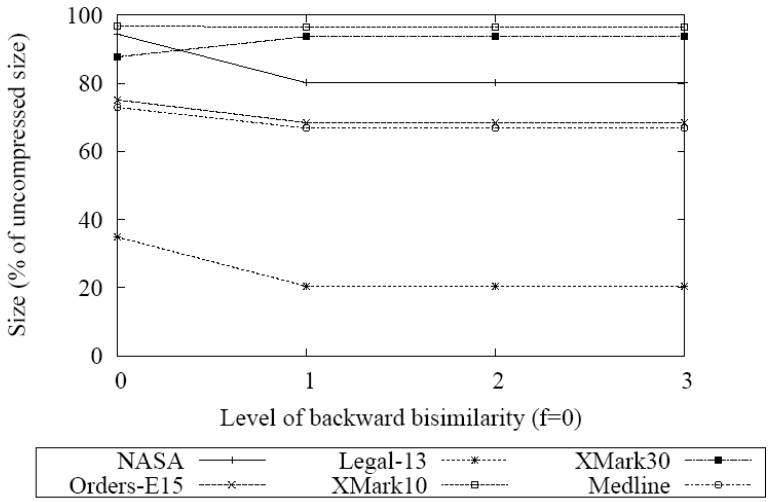

Figure 7. Effects of backward bisimilarity on size.

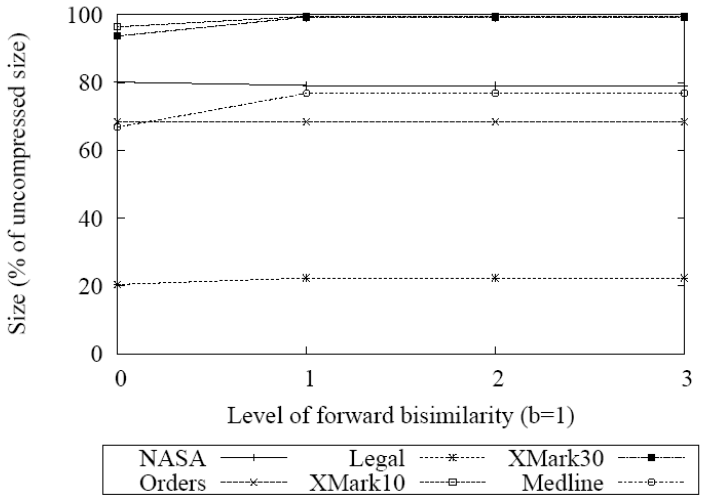

Figure 8. Effects of forward bisimilarity on size.

Further increases in backward bisimilarity had negligible effect upon compression, with only one dataset showing a slight compressed size increase of less than 0.1% between (0,1)-F+B and (0,2)-F+B).

In terms of forward bisimilarity, the move to (1,n)-F+B (where n >= 1) tended to produce an adverse effect on the compressed size (Figure 8) – up to 10% in the worst case (Medline dataset). Adding an extra discriminator to the partitioning process generally causes the data to be split into a larger number of blocks, this often means that individual data values will appear in a higher number of data dictionaries than they did without forward bisimilarity and consequently the overall dictionary size will increase. However this increase can be offset by the use of smaller tokens to represent the data values of these smaller dictionaries, leading to a reduction in DataBlock size. It can be seen that there is in fact an overall 1% improvement in compression for the NASA dataset as a result of adding forwards bisimilarity.

With no great benefit shown by partitioning using forward bisimilarity, and noticeably improved compression shown only when moving up to one level of backward bisimilarity, it would appear that the (0,1)-F+B partitioning scheme allows for greatest compression with least effort. However, with a view to sharing the data in individual blocks, the overall size is only one consideration – the number of blocks the data is split into is also a factor, as fewer blocks will necessarily be larger blocks.

In all cases the increase in backward bisimilarity from (0,0)-F+B to (0,1)-F+B produces a slightly increased number of blocks (Figure 9). The effects of changing forward bisimilarity are greater, with a change from (0,n)-F+B to (1,n)-F+B (where n >= 1) results in a considerable increase the number of blocks for some data sets (Figure 10). It follows that partitioning the data into different numbers of blocks will cause the overall compression level for that data to change, as the higher the number of blocks the data is split into, the less likely it is that repeated items will be found within a single block. This accounts for the increase in data sizes shown in Figure 8 as, especially for the Medline, XMark10 and XMark30 datasets, these are accompanied by a significant increase in the number of blocks.

It is noted that there is a level of forward and backward bisimilarity beyond which no significant changes are made to either the compressed size or the number of data blocks. This is caused by the fairly flat, regular schema of the test datasets. Files with a deeper tree structure are expected to show further changes in partitioning as bisimilarity is increased.

We consider that generally (1,1)-F+B is a reasonable compromise between level of compression and number of blocks produced. Therefore the remainder of the results shown below are obtained using that partitioning scheme as a basis.

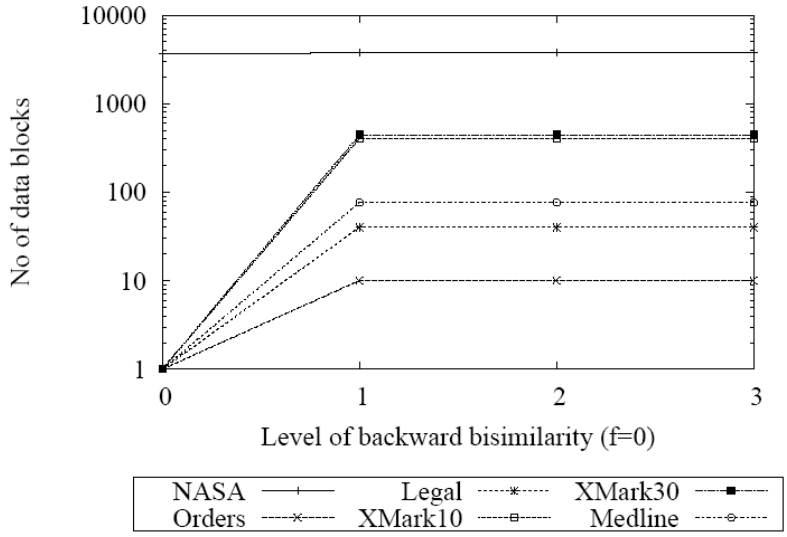

Figure 9.
Effects of backward bisimilarity on block numbers

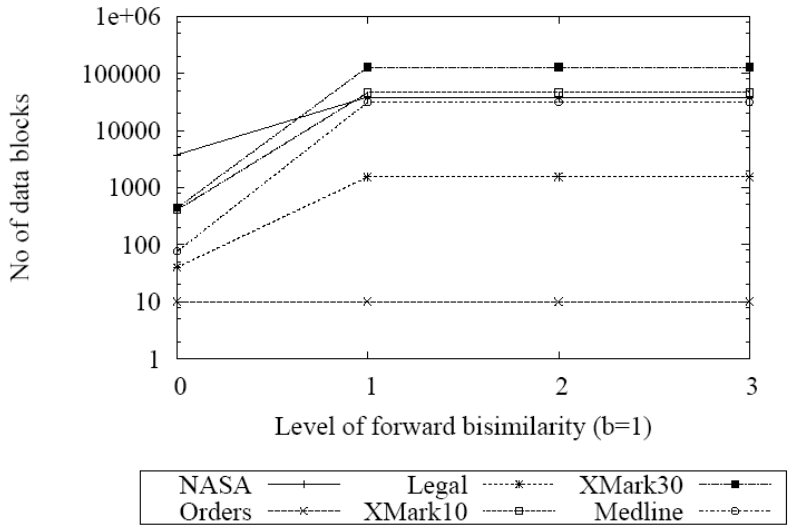

Figure 10.
Effects of forward bisimilarity on block numbers

As explained in the system overview section, additional processing can be applied to the dictionaries produced by NSIndex to reduce redundancy by removing duplicate and subset dictionaries. This thinning process found and removed redundancy in all datasets with the exception of Orders, which due to its flat structure is comprised of only ten dictionaries. Figure 11 shows the reduction in the overall number of files caused by dictionary thinning in the other datasets. In the most successful case, the Medline dataset, only 29% of the dictionaries produced were retained after the thinning process. In the worst case XMark10 retained 68% of dictionaries after thinning.

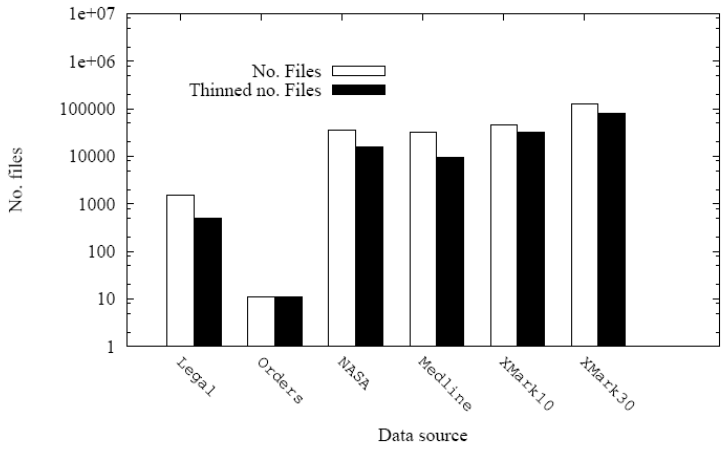

Figure 11. Thinning effect on no. of dictionaries

The consequent effects on overall dictionary size are shown in Figure 12. The results for these datasets correlate closely with the reduction in dictionary numbers. However this may not necessarily be the case for any future datasets, as any reduction of overall file size will be dependent on the sizes of the individual dictionaries removed.

Full results of the thinning process are shown in Table 3. As may be expected, the greatest savings are to be made over the real world datasets, as these are more likely to have repeated values compared with the generated benchmark data. The results show that thinning the dictionaries after partitioning can have considerable benefit in terms of data storage.

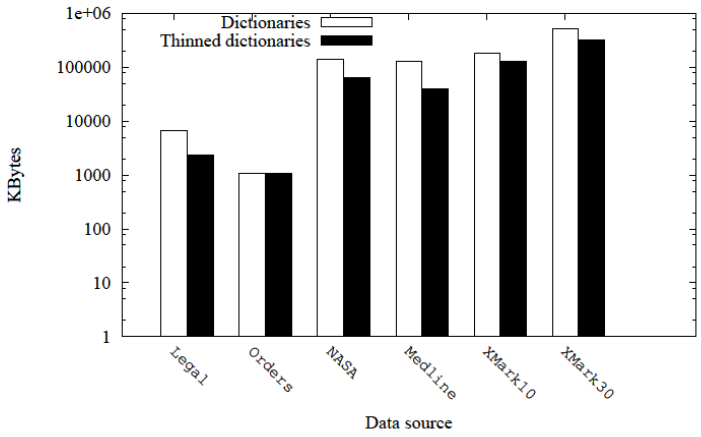

Figure 12. Thinning effect on total dictionary size

	Reduction in number of files	Reduction in dictionary size on disk
Legal	67%	64%
Orders	0%	0%
NASA	56%	55%
Medline	71%	69%
XMark10	32%	32%
XMark30	39%	39%

Table 3. Consequences of thinning on dictionary files.

6. Conclusion and Future Work

The purpose of the read-only data architecture that we have designed is to partition large XML structures so that they can be shared more efficiently between small mobile devices such as smartphones. This is achieved by using F+B-indexing to characterise the partitions produced. Our experimental work suggests that the (1,1)-F+B-index presents the best possibilities in the context of the varied benchmark datasets we have used. Furthermore, we have found that useful savings can be made by combining dictionary structures so that redundant collections are excluded. We have verified that the process of excluding such dictionaries preserves the direct query capability that enables queries to be resolved without firstly decompressing the whole data structure. This functionality is of significant advantage in the design of mobile information systems that operate on small computing devices such as smartphones. In this context, peer data sharing for applications such as advertising and entertainment will require

significant changes in user behaviour. However, the potential benefits that are available support the expectation that such changes will come about. The approach to data representation that we describe offers a way of minimising the data transfer costs that are implicit in such a model and at the same time reduces battery usage by supporting query resolution directly on compressed data structures.

Our results suggest that bisimilarity provides a basis for partitioning large data structures so that they can be compressed efficiently. The need to process large structures that may not fit into main memory has led to the development of interest in XML stream processing and the potential for compressed streams of XML to be made accessible to query processing. Stream processing will provide better response times for our target class of applications and we plan to explore the use of this approach in conjunction with sharing both dictionary and block information. Security and integrity of data in such an environment are important issues and we plan to address the implications of authentication, access control, privacy and encryption.

We are further investigating the potential for splitting unusually large dictionaries and other methods of recombining small dictionaries to optimise the use of minimal-bit tokens. We are also planning to evaluate the sharing of the partitioned elements between devices in the context of location-based data.

References

Arion, A., Bonifati, A., Manolescu, I., and Pugliese, A. (2007), "XQueC: A query-conscious compressed XML database", *ACM Transactions on Internet Technology*, Vol. 7, No. 2, Art. 10.

Aygűn, B.T., Ma, Y., Akkaya, K., Cox, G., and Bicak, A. (2009), "A conceptual model for data management and distribution in peer-to-peer systems", *Peer-to-Peer Networking and Applications*, available at: http://www.springerlink.com/content/2n67v1279t780347/ (accessed March 2010).

Balakrishnan, H., Kaashoek, M., Karger, D., Morris, R.t and Stoica, I. (2003), "Looking up data in P2P systems", *Communications of the ACM*, Vol. 46, No. 2, pp 43-48.

Böttcher, S. and Steinmetz, R. (2007). "Evaluating XPath queries on XML data streams", in *Data Management: Data, Data Everywhere, 24th British National Conference on Databases, BNCOD 24, Glasgow, UK, July 3-5, 2007.*, Springer-Verlag, London, pp. 101-113.

Böttcher, S., Steinmetz, R., and Klein, N. (2007), "XML index compression by DTD subtraction, in *Proceedings of the Ninth International Conference on Enterprise Information Systems, Volume DISI, Funchal, Madeira, Portugal, June 12-16, 2007.*, Springer-Verlag, Vienna, pp. 86-94.

Bonifati, A., Chrysanthis P.K., Ouksel, A.M., Sattler, K-U. (2008), "Distributed databases and peer-to-peer databases: past and present", *SIGMOD Record*, Vol. 37, No. 1, pp. 5-11.

Buneman, P., Grohe, M. and Koch, C. (2003), "Path Queries on Compressed XML", in *Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany.*, Morgan Kaufmann, San Francisco, pp. 141-152.

Buneman, P., Choi, B., Fan, W., Hutchison, R., Mann, R. and Viglas, S. (2005), "Vectorizing and Querying Large XML Repositories", in *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, 5-8 April 2005, Tokyo, Japan.*, IEEE Computer Society, pp. 261-272.

Cockshott, W.P., McGregor, D., Wilson, J. (1998), "High-Performance Operations Using a Compressed Database Architecture", in *The Computer Journal*, Vol. 41, No. 5, pp. 283-296.

Daswani, N., Garcia-Molina, H., and Yang, B. (2003), "Open Problems in Data-Sharing Peer-to-Peer Systems", in *Proceedings of the 9th International Conference on Database Theory, Siena, Italy, January 8-10, 2003.*, Springer-Verlag, London, pp. 1-15.

Gourlay, R., Tripney, B., Wilson, J.N. (2007), "Compressed Materialised Views of Semi-Structured Data", In *Workshop Proceedings of the 24th British National Conference on Databases, BNCOD 2007, University of Glasgow, UK, 3-5 July 2007.*, IEEE Computer Society, pp. 75-82.

Kangasharju, J. and Tarkoma, S. (2007), "Benefits of alternate XML serialization formats in scientific computing", In *Proceedings of the 2007 Workshop on Service-Oriented Computing Performance: aspects, issues, and approaches*, ACM, New York, pp. 23-30.

Kaushik, R., Krishnamurthy, R., Naughton, J.F. and Ramakrishnan, R. (2004), "On the Integration of Structure Indexes and Inverted Lists", In *Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, 30 March - 2 April 2004, Boston, MA, USA.*, IEEE Computer Society, p. 829.

Kaushik, R., Shenoy, P., Bohannon, P. and Gudes, E. (2002). "Exploiting local similarities for indexing paths in graph-structured data", in Proceedings *of the 18th International Conference on Data Engineering, 26 February - 1 March 2002, San Jose, CA.*, IEEE Computer Society, Washington, pp. 129-140.

Koloniari, G. and Pitoura, E. (2005), "Peer-to-peer management of XML data: issues and research challenges", *SIGMOD Record*, Vol. 34, No. 2, pp. 6-17.

Liebnitz, K., Hoßfeld, T., Wakamiya, N. and Murata, M. (2007), "Peer-to-Peer vs. Client/Server: Reliability and Efficiency of a Content Distribution Service", In *Managing Traffic Performance in Converged Networks, 20th International Teletraffic Congress, ITC20 2007, Ottawa, Canada, June 17-21, 2007, Proceedings.*, Springer LNCS, Germany, pp. 1161-1172.

Leifke, H. and Suciu, D. (2000), "XMILL: An Efficient Compressor for XML Data", In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, ACM, New York, pp. 153-164.

Little, J. D. C. (1961), "A Proof of the Queueing Formula L = λ W", *Operations Research*, 9, 383-387

McHugh, J., and Widom, J. (1999), "Query optimization for XML", in *Proceedings of the 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK.*, Morgan Kaufmann, Orlando, FL., pp. 315-326.

Meyer, D. (2009), "Gartner sees slowdown in smartphone sales growth", available at: http://www.zdnetasia.com/news/communications/0,39044192,62052144,00.htm (accessed 17 February 2010).

Natchetoi, Y., Wu, H., Babin, G. and Dagtas, S. (2007), "EXEM: Efficient XML data exchange management for mobile applications", *Information Systems Frontiers*, Vol. 9, No. 4, pp. 439-448.

Neumüller, M., and Wilson, J. N. (2003), "Improving XML Processing Using Adapted Data Structures", in *Web, Web-Services, and Database Systems, NODe 2002 Web- and Database-Related Workshops, Erfurt, Germany, October 7-10, 2002.*, Springer, Berlin, pp. 206-220.

Qiu, D., Srikant, R. (2004), "Modeling and performance analysis of BitTorrent-like peer-to-peer networks", In *Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 30 - September 3, 2004, Portland, Oregon, USA.*, ACM, New York, pp. 367-378.

Schneider. J. (2003), "Theory, benefits and requirements for efficient encoding of XML documents", in *W3C w3c-binary-workshop*, available at: http://www.w3.org/2003/08/binary-interchange-workshop/30-agiledelta-Efficient-updated.html (accessed March 2010).

Stonebraker, M., Abadi, D.J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., Lau, E., Lin, A., Madden, S., O'Neil, E., O'Neil, P., Rasin, A., Tran, N., and Zdonik, S. (2005), "C-Store: A Column-oriented DBMS", in *Proceedings of the 31st Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005.*, Morgan Kaufmann, Orlando, FL., pp. 553-564.

Wilson, J.N., Gourlay, R., Japp, R., Neumüller, M (2006), "A Resource Efficient Hybrid Data Structure for Twig Queries", in *Database and XML Technologies, 4th International XML Database Symposium, XSym 2006, Seoul, Korea, September 10-11, 2006, Proceedings.*, Springer LNCS, Germany, pp. 77-91.

Wehrle, K., and Steinmetz, R. (2005), “What Is This *Peer-to-Peer* About?”, Wehrle, K., and Steinmetz, R., *Peer-to-Peer Systems and Applications*, Springer, Berlin, pp 9-16.

Wolfson, O., Xu, B. and Tanner, R. (2007). "Mobile Peer-to-peer Data Dissemination with Resource Constraints", *Proceedings of the 8th International Conference on Mobile Data Management, Mannheim, Germany, May 2007*, IEEE Computer Society, Washington, pp. 16-23.