考試科目	資訊管理論文評析	所別	資訊管理學系	考試時間	5 月 22 日(六) 第一節

請先閱讀所附論文，而後回答下列問題。

一、請簡要說明此論文的內容。(5 分)

二、假設你是某國際知名 SCI 期刊所邀請的評審，請針對該論文提出建設性建議與批判，至少應包含下列各項**(a) paper organizations and wording, (b) originality value, (c) research method, (d) practical and academic contributions, (e) suggested modification.** 並請最後提供給主編你的建議，該論文是該「接受」、「小修不審」、「大修再審」、「拒絕」？(30 分)

三、假設政治大學某 A 教授取得該作者的合作，將此 PrPl e-porfolio 系統引入台灣。請以資訊管理專家角度，提出建議該系統應如何導入配置？會牽涉哪些 Stakeholders？有哪些管理議題必須被解決、你建議如何處理？(25 分)

四、假設你已是政治大學博士生，在上述 A 教授導入一年後，你想驗證其實際成效，請提出你的實證研究計畫書，請留意該計畫書應至少包含下列各點，分別陳述清楚：**(a) Theory Backgrounds, (b) Proposed Research Model, (c) Definitions of Research Variables and Measures, (d) Testable Hypotheses and Justifications, (e) Initial Version of Questionnaire, (f) Research Procedure Design, (g) Suggested Statistical Analysis Method**。(40 分)

備註	試題隨卷繳交

考試科目	資訊管理論文評析	所別	資訊管理學系	考試時間	5月22日(六) 第一節

When cloud computing meets with Semantic Web: A new design for e-portfolio systems in the social media era

Abstract

The need, use, benefit and potential of e-portfolios have been analysed and discussed by a substantial body of researchers in the education community. However, the development and implementation approaches of e-portfolios to date have faced with various challenges and limitations. This paper presents a new approach of an e-portfolio system design based on Private–Public (PrPl) data index system, which integrates cloud computing applications and storages with Semantic Web architecture, making semantic web-based visualisation and advanced intelligent search possible. It also discusses how the distinctive attributes of the PrPl-based digital asset management system can serve as a large-scale robust e-portfolio system that can address issues with scalability, sustainability, adoptability and interoperability. With such a new distinctive design, a large-scale deployment at a state or national level becomes possible at a very cost-effective manner and also such large-scale deployment with intelligent digital asset management and search features create numerous opportunities in education.

Introduction

A portfolio is a purposeful collection of student work that exhibits the student's efforts and achievements (Paulson, Paulson & Mercer, 1991; Wiedmer, 1998) and progressive improvement over time (Barrett, 1994; Challis, 2005) in a life-long journey with academic and professional endeavours. With the advent of the digital age, the emergence of electronic portfolios (e-portfolios) appeared as a logical next step for portfolios. In fact, over half of US higher education institutions are using some form of e-portfolios (Green, 2008), and the popularity of e-portfolio has grown out from the higher education arena to flood into a broader population: Rhode Island implemented a state-wide e-portfolio system for its high school students(RINET, 2009), and Minnesota provides life-long e-portfolios for all residents (eFolio, 2009). Not only serving as a repository of digital artefacts or evidence of mastery and achievement, an e-portfolio also serves as an assessment tool for users and becomes a learning place where a student matures learning through self-reflection (Bhattacharya & Hartnett, 2007). These recognised benefits convinced many institutions of higher education in the US to college system-wide e-portfolio implementation initiatives such as e-portfolio California project funded by a grant from the California Community Colleges System Office (CVC, 2009).

Furthermore, the ease and diversity of content creation afforded by social web tools, and the ubiquity of technology use amongst learners today, the range of e-portfolio activities and the

考試科目	資訊管理論文評析	所別	資訊管理學系	考試時間	5月22日(六) 第一節

kinds of digital artefacts that could possibly be representative of an individual's e-portfolio have increased substantially. For example, educators have reported and discussed the use of e-portfolio systems in creating a collaborative learning space (Greenberg, 2004), fostering peer assessment (Stevenson, 2006), student advisement and career planning (Lorenzo & Ittelson, 2005a), content visualisation with electronic concept maps (Kim, 2006), planning and managing instruction (Sherman, 2006), and improving the effectiveness of faculty (O'Brien, 2006).

Despite the growing interest and use, there is a series of issues and challenges that austerely hinder the implementation and the wide use of e-portfolio systems firmly integrated in academic and professional activities. Therefore, this paper attempts to identify challenges and issues from current cases of implementations and development approaches with e-portfolio systems; introduces a new radical design approach as a viable solution; and shares potential implications of such design.

Current efforts to implement e-portfolio systems

Efforts in implementing e-portfolio systems in education have been pursued and well-documented in various European countries, Australia, as well as in the US. For example, the European Institute of ELearning (EIfEL) set up a Europortfolio group with a mission to 'engage upon an orchestrated effort involving both educational and corporate institutions to define, design, and develop digital portfolio systems that meet the needs of all stakeholders' (EIfEL, 2007). The Australian e-portfolio Project was awarded to a consortium of four universities—Queensland University of Technology, The University of Melbourne, University of New England and University of Wollongong—to, amongst other goals, 'provide an overview and analysis of the national and international e-portfolio contexts, document the types of e-portfolios used in Australian higher education,' and 'identify any significant issues relating to e-portfolio implementation' (AeP, 2008). In the US, the premier example is eFolio Minnesota, considered the first state-wide e-portfolio system in the US. eFolio provides every resident in the state of Minnesota a free lifetime e-portfolio with limited storage space (http://www.efoliominnesota.com/). Since it was launched in October 2002, over 90 000 residents and students in Minnesota have used the e-portfolio system. The number of registered users has grown linearly at a rate of approximately 1300 new users each month, and over the year ending in 2005, eFolio received an average of over 2 million hits and 67 000 unique visits per month (Cambridge, 2008).

With reference to the functionalities of an e-portfolio system, Greenberg (2004) described the e-portfolio as 'a network application that provides the author with administrative functions for managing and organising work (files) created with different applications and for controlling who can see the work and who can discuss the work (access)'. From a technical perspective, adopting an e-portfolio system typically involves setting up the e-portfolio application on a server, and providing a large data storage facility to house user files. Students then create accounts, add data files into the system and create different assemblages of the data files targeted towards particular audiences.

In terms of the technical implementation of e-portfolios, there is currently great diversity in the technologies being used by various institutions. The Australian ePortfolio Project (2008) reported a range of tools and applications being used by Australian universities. Some universities used extensive applications like full e-portfolio systems, virtual learning environments and adapted learning management systems, whereas others simply employed student web pages and blogs. In the US, Lorenzo and Ittelson (2005b) observed four basic approaches in implementing e-portfolio systems: (1) home-grown, proprietary systems like the DU Portfolio Community system from the University of Denver and the UW Catalyst Portfolio tool from the University of Washington; (2) open source, publicly available systems like the VT Electronic

考試科目	資訊管理論文評析	所別	資訊管理學系	考試時間	5月22日(六) 第一節

Portfolios pilot from Virginia Tech; (3) commercial, licensed systems like Avenet eFolio, which provides the e-portfolio platform for the Minnesota project; and (4) common hyper text mark-up language editors like Microsoft Frontpage and Adobe Dreamweaver.

Existing implementation issues and challenges

Despite numerous benefits of e-portfolios recognised throughout the academic community, only a handful number of institutions are reported as having mature campus-wide e-portfolio systems (Paoletti, 2006). In fact, many institution-wide implementation initiatives were stalled or are remaining at pilot stages on many US university campuses (Paoletti, 2006). The impeded acceptance of e-portfolios on many campuses can be explained by a number of barriers observed during e-portfolio implementation sites throughout the US and other parts of the world. Studies observe a range of challenges against a successful e-portfolio implementation at higher education institutions (Canada, 2002; Lorenzo & Ittelson, 2005a, b; Sherry & Bartlett, 2005; Tosh, Light, Fleming & Haywood, 2005; Wetzel & Strudler, 2005). In sum, such challenges include the insufficient technical infrastructure (hardware, software and IT support), lack of skills and knowledge among students and staff, demand on increased time commitment for users (especially on the part of teaching staff), and the problems with security and privacy of data. Especially, the myriad of technologies currently being employed, and the increase in range, file size and diversity of possible digital artefacts have given rise to a number of implementation issues and challenges:

Scalability and sustainability

Current e-portfolio systems generally provide users storage space in a centralised server for all of their data. This poses immediate problems of scalability and sustainability, especially for institutions with large populations of active users. With the increasing use of multimedia, and the ease of creation of media-rich content, there is a drastic increase in storage load imposed by any user on the e-portfolio system. For example, the 3 Mb storage space provided for each user in the eFolio Minnesota system would hardly be enough for the avid user today. In fact, Cambridge (2008) reported that users in the Minnesota system had given feedback that the 'small amount of storage space allotted to them by the eFolio software impedes their ability to include substantial personalising multimedia content'. In addition to immediate storage requirements, questions such as storage growth rates and the duration of each user's membership in the system after graduation from educational institutions have yet to be addressed.

Data transportability

As users import all their data into the centralised e-portfolio server of an institution, it is essential for users to be able to transport their information as they move from one educational institution to another. Most of the e-portfolio systems in use today are not interoperable. As Jafari (2004) described, 'we do not yet have all the necessary interoperability requirements defined for various types of e-portfolios, and this is causing ever-increasing challenges for developers of e-portfolio software environments as the projects increase in size and complexity'. This challenge in data transportability has in turn given rise to issues with data ownership. Naturally, any user would like to export the data and to carry it with him when he moves on to other institutions, but who is the real owner of the data in the e-portfolio? Is it the author or the institution?

Barriers to user adoption

'If we build it, will they come?'—this question is a line adapted from the 1989 film, Field of Dreams. Patent (2007) asked this question in the context of an e-portfolio system—highlighting that 'it is not a foregone conclusion that just because we build an e-portfolio into a course, students will want to use it, or find it beneficial in developing their learning in the ways it was designed to do'. The critical question we need to ask ourselves is whether there are barriers to the use of e-portfolios by learners.

考試科目	資訊管理論文評析	所別	資訊管理學系	考試時間	5 月 22 日(六) 第一節

An e-portfolio system is to provide a useful personal space where students can collect the digital artefacts that present evidence of their experiences and achievements. On the other hand, the learners of today, who are more accurately referred to as 'digital natives' are adept at using Web 2.0 tools. They already 'rely heavily on communications technologies to access information and to carry out social and professional interactions' (Kennedy, Judd, Churchward, Gray & Krause, 2008). 'Web 2.0 tools tend to be relatively unstructured and are characterised by an ease of publishing, a high level of interaction, self-assigned semantics (tagging) and are often media rich' (Cotterill, White & Currant, 2007). With learners already using (or fluent in) social and Web 2.0 technologies in interacting with each other and in showcasing their media artefacts, current e-portfolio systems require users to go through an additional process of duplicating the media artefacts which may already be residing in these diverse web applications, and uploading them onto the e-portfolio system. This is potentially time-consuming and laborious, and could pose as a barrier to user adoption.

Introducing PrPl

As highlighted earlier, with Web 2.0 technologies and applications, a significant amount of user data, or media artefacts, is stored in online storage services in the Internet cloud. Users today are inundated with a myriad of online services that offer data storage in the cloud for free. Such cloud computing services (eg, Google Docs, Facebook, Flickr, MySpace, Picasa, YouTube) allow users to share their data with anyone, anywhere and at anytime. Online blogging sites also offer free hosting services to users, and social networking sites provide free platforms for user interactions over the web. Furthermore, the technology trend is moving towards offering software as a service on the cloud, rather than as a standalone product. Software applications like Adobe Photoshop and Google Docs already operate on data stored in the cloud. Rather than operating an e-portfolio system apart from these online services, cloud computing offers a potential solution to the challenges to e-portfolio implementation—issues of sustainability, scalability, data portability and user adoption.

Unfortunately, until recently, in spite of its potential, cloud computing has posed more challenges than solutions. Cloud computing services range from the narrow, like uploading pictures onto Flickr, to broad ones like Facebook, where users have the ability to define applications. The bulk of cloud computing services is currently application-centric, and requires the user to explicitly upload data to different websites. This results in data scattered across diverse devices and websites. Any one form of media can be stored across a variety of different services. With the potentially exponential rise in the amount of data stored across various services online, it is becoming increasingly difficult to keep track of one's own data. Consequently, this poses problems of inconvenience and potential loss of data ownership and privacy due to reduced capacity to monitor and mange the data (Seong *et al*, 2009).

PrPl (which stands for Private–Public) is a personal-cloud computing infrastructure being developed from an on-going NSF-funded project (POMI, 2009) lead by researchers at Stanford University. Tapping on the wide range of free cloud computing services available to the user, PrPl adopts a user-centric design approach to create a federated storage system out of existing web services and applications. It is a collaborative and semantically indexed data management system which functions as a simple index of data that users may already have stored in other services. PrPl therefore allows users to continue to take advantage of the free storage services—the PrPl infrastructure then serves to present 'a unified, location-agnostic view of the data in a user's personal cloud' (Seong *et al*, 2009).

There are two key concepts in the PrPl infrastructure:

考試科目	資訊管理論文評析	所別	資訊管理學系	考試時間	5 月 22 日(六) 第一節

PrPl Semantic Index

The PrPl infrastructure creates a semantic index of all the data available to it, and presents a semantic interface that facilitates retrieval of relevant information. According to Seong *et al* (2009), 'the PrPl Semantic Index is a Semantic Web that keeps track of all the data in a personal cloud. The metadata includes enough information to answer typical queries about the data, and location of the body of the larger data types. In addition, it also keeps track of the access rights granted to our friends at fine granularity. With the help of a uniform resource identifier, the protocol eliminates data copying by allowing the application to retrieve data directly from storage'.

Personal-Cloud Butler (PCB)

In the PrPl system, each user has a PCB, a program that manages a user's personal cloud of information on his or her behalf (Seong *et al*, 2009). The PCB functions as a single interface that mediates between all the data repositories and cloud computing services of the user, thereby providing the user with one-stop access to all his or her data in the cloud. Although the detailed description of the functional algorithms involved in the security management is beyond the scope of the paper, in short, the user provides access credentials of each cloud computing service in a database for the PCB to reference as needed in managing numerous artifacts in the cloud automatically.

Therefore, the semantic index enables the user to track and organise numerous personal digital artifacts in the cloud without actually storing them (eg, large video files) in a centralised server, eliminating the need of a vast storage space. Also, the semantic attributes of the index make it possible for the user to efficiently organise and visualise personal cloud artifacts while allowing intelligent searches that go much beyond conventional keyword-based searches. In addition, PCB is a personal cloud crawler that tracks, verifies and updates personal cloud artifacts with minimum user involvement. The combination of unique features of PrPl system infrastructure enables educators to design a much more scalable and intelligent e-portfolio system.

A PrPl-based e-portfolio system

The key features of PrPl, namely the Semantic Index and the PCB, are keenly aligned to the elements of an e-portfolio system—the PrPl platform thus appears to be a natural fit to address the challenges currently faced.

As illustrated in Figure 1, PrPl functions as a simple index of data that users may already have stored in other services. It allows them to access these services in a centralised way, thus

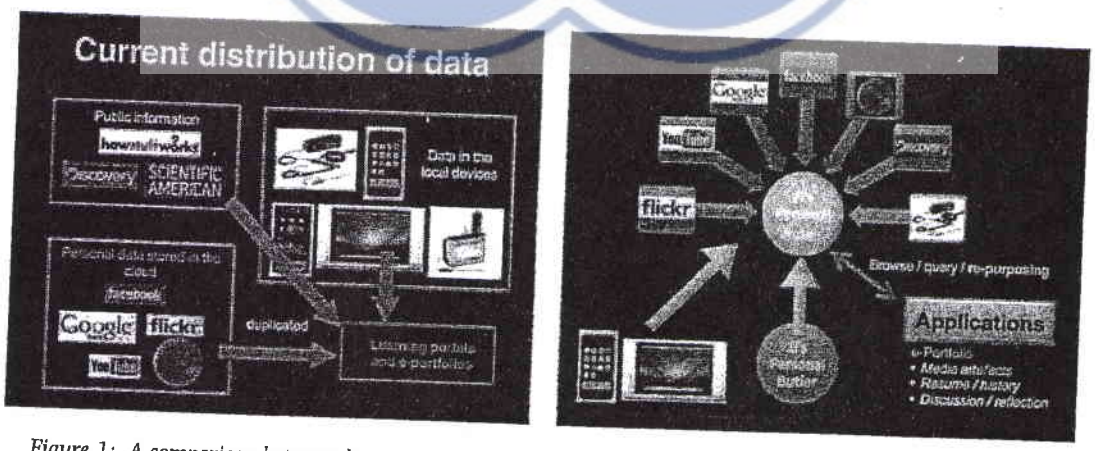

Figure 1: A comparison between how current e-portfolios manage data distributed on the cloud, and how the proposed PrPl-based e-portfolio system facilitates integration

考試科目	資訊管理論文評析	所別	資訊管理學系	考試時間	5 月 22 日(六) 第一節

presenting the functional equivalent of a closed e-portfolio platform without the drawbacks. Data in the PrPl system is semantically indexed using a resource description framework (RDF) triple store, allowing it to be retrieved using the SPARQL Protocol and RDF Query Language (SPARQL) queries. Coupled with the emerging technology in translating natural language into the SPARQL query language (See Ran & Lencevicius, 2007), more convenient ways of searching and retrieving data within the e-portfolio system can be expected.

As discussed earlier, researchers highlighted the challenges to implementing an e-portfolio system. Certainly, PrPl is not a panacea and there might be a whole new set of other issues with its full implementation. However, it addresses some of the currently well known challenges. Here, we present the attributes of a successful e-portfolio and present how a PrPl-based e-portfolio system may be aligned with those attributes:

Ease of use

According to Jafari (2004), a successful e-portfolio must offer an attractive and relatively simple interface, with little or no training required. One of the key design factors of the PrPl architecture is that of integration and simplicity. The PCB presents a single, unified view of one's data and relations in the personal cloud, hiding 'the complexities of authentication and authorisation' (Seong *et al*, 2009). The users can also continue to use their existing data storage services on the cloud and does not need to go through the laborious process of duplicating the data into the e-portfolio server—thereby concurring with the attribute of 'ease of use'.

Sustainability

Long-term sustainability is a key issue in any e-portfolio project. The sustainability of providing large storage services for a high quantity of existing users notwithstanding, an e-portfolio project also has to address questions of whether and how long to archive data from users who have left the institution. The PrPl-based e-portfolio system does not need to address issues like these, because the users' data remain with their personal data services in the cloud. Even if the user creates and showcases media-rich, storage-intensive data, this storage load is imposed on their personal cloud computing services, rather than on the PrPl-based e-portfolio system. The system merely stores small scripts which enable the PrPl system to query these other data sources. At the same time, such small database could be easily picked up and operated as a free global directory service. Hence, this system is built for sustainability, and can easily be scaled exponentially.

Advanced features

Jafari (2004) emphasised that a successful e-portfolio system needs to have sufficiently advanced features to attract users who are concurrently inundated with numerous new web-based services and products each day. Due to the fact that the PrPl architecture is built to leverage on existing, and perhaps even hitherto unreleased software applications, a PrPl-based e-portfolio system does not need to 'compete' with these new services and features. The user is free to capitalise on these services—one just needs to open up access to these services. Furthermore, the proposed e-portfolio is built on an open architecture that is flexible and dynamic, and can integrate new features and applications as they arise from the user community.

Robust integrated technology architecture

An integrated technology architecture was advocated by Jafari (2004) as he anticipated that users would need to access and make reference to multiple types of data across various platforms and housed in different servers. The PrPl architecture precisely addresses this concern. The PCB functions as an integrated interface—it maintains 'an up-to-date PrPl Semantic Index for all the data in an owner's federated storage system. The Butler keeps track of all the owner's storage

考試科目	資訊管理論文評析	所別	資訊管理學系	考試時間	5 月 22 日(六) 第一節

devices' (Seong *et al*, 2009). Users retrieve data through applications which send queries only through the Butler, hence presenting an integrated system.

Lifelong support

One great concern regarding e-portfolio systems is what happens to the data when a student graduates, or moves on to another institution, but wishes to continually update the media artefacts showcased in the e-portfolio system. Offering lifelong support would attract a user to commit her time and effort to build up the e-portfolio, as she would be able to continually build and use it even after graduation. The PrPl-based e-portfolio system allows for this, as the user can continue to build on, and load data onto her personal cloud computing storage space which can be also portable. As this does not impose excessive storage load on the PrPl server (beyond additional small scripts), this lifelong support becomes highly sustainable.

Standards and transportability

Issues of standards and transportability have plagued the e-portfolio space since its inception. The IMS Global Learning Consortium, consisting of several dozen universities and corporate partners, has developed and implemented a set of XML-based standards that allow interchange of data between e-portfolio systems. The efficacy of these standards has yet to be determined, but issues of data ownership, long-term storage, and non-portability of data remain, while the continuous rapid advancement of new technology features at various fronts make the standardisation even harder. To date, the PrPl-based indexing system for content spanning multiple data hosts remains a novel one in the e-portfolio space, and presents a viable solution that circumvents these issues. Data are still owned and controlled by the user, and data portability is a non-issue.

Semantic Web attributes of the PrPl system

'X' attribute

In his paper, Jafari (2004) refers to an 'X' attribute—'other important unknown attributes that may contribute to the success of an e-portfolio project'. With reference to the PrPl-based e-portfolio system, its ability to support semantic query is a crucial attribute for success. Semantic-aware applications have been highlighted in the 2009 Horizon Report (Johnson, Levine & Smith, 2009) as an upcoming technology piece predicted to feature even more prominently within the next four to five years. The PrPl architecture keeps track of data through the use of a semantic index—this allows for searching of data by using the semantics, or meaning, of information rather than strict key word searches. Hilzensauer, Hornung-prahauser and Schaffert (2006) have highlighted that Semantic Web technologies can be leveraged in e-portfolio systems to: (1) derive formal descriptions of artifacts; (2) overcome interoperability issues; (3) facilitate knowledge management, linking artifacts to personal development plans; and (4) support reasoning, and perform matching of users of appropriate profiles with relevant learning paths, or even job openings or ad hoc professional engagements.

Content organisation and visualisation

The PrPl-based e-portfolio system is as close as one can get to a semantic-based e-portfolio—this may yet be the crucial 'X' factor for its success, as there is much potential to leverage on both the Semantic Web and e-portfolios in education (Schaffert, 2006). The grand vision is that the PrPl-based e-portfolio system will serve as a place to collect, select, manage and present one's data or achievement for multiple purposes. The Semantic Web infrastructure enables user-defined access for different groups of audience. The system thus allows for multipurpose presentations of the portfolio assets, facilitating the repurposing of assets for user-defined targets.

In terms of content visualisation, which needs much more work in the overall project, various approaches might be taken to maximise the usability and efficiency of the user interface. Among

考試科目	資訊管理論文評析	所別	資訊管理學系	考試時間	5 月 22 日(六) 第一節

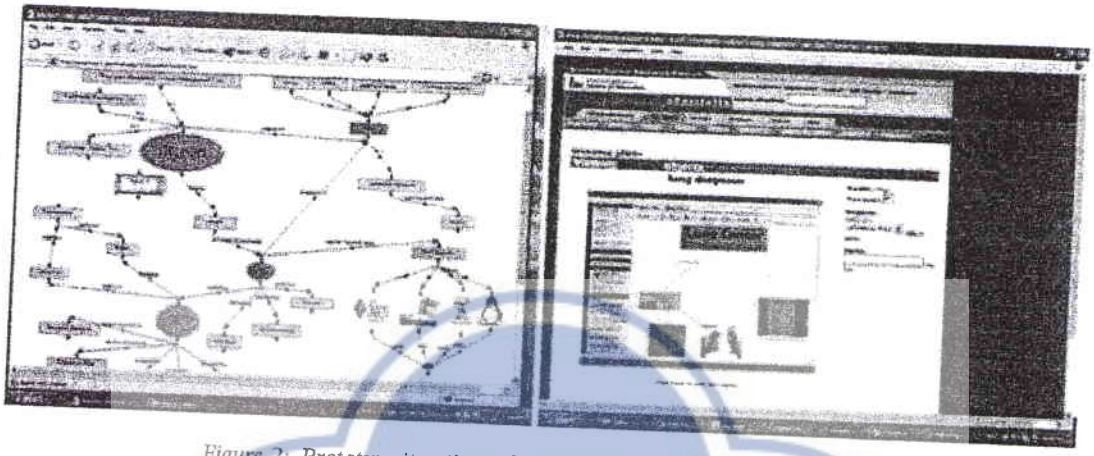

Figure 2: Prototype iterations of visual map-based e-portfolio interface

the substantial amount of relevant studies, there is a growing number of researchers who are supporting the idea of adopting a visual-mapping approach in the design of advanced digital content-management systems (Alpert, 2005; Coffey, 2005; Kim, 2006; Tergan, 2005). The consensus among them is that the integration of a visual-mapping interface in knowledge-management systems such as e-portfolio systems may lead to a better understanding of the aggregated content in comparison to systems incorporating a traditional display method (eg, conventional tree or hierarchical folder structures for organising and presenting digital contents). As an empirical study, Kim and Olaciregui (2008) examined the effects of visualisation variations and found that the visual map-based interface (ie, knowledge and content presentation on a visual map foundation) was superior to a conventional presentation method (eg, tree or hierarchical content organisation) in overall performance test, access time and retention. In other words, the visual map layout might have helped the user understand how multiple digital artefacts were meaningfully connected, organised and presented.

With PrPl-based e-portfolio system, the visuospatial layout (eg, sample shown in Figure 2) can be constructed from parsing and analysing the attributes of the Semantic Web (ie, metadata described in RDF as a series of subject-predicate-object triples) and by doing so it provides a snapshot of overall artefact organisation and may aid the user in navigating and internalising the content more efficiently. Nonetheless, since most conventional layouts of electronic resumes or web presentations are in the form of straight top-down or hierarchical fashion, the visuospatial layout will need to be one of optional interfaces the user can choose until the user finds the novel approach suitable and useful.

Unique identifier

The PrPl-based e-portfolio system will utilise a user's mobile phone number as a unique identifier. A user's mobile phone number is proposed as a novel strategy for user account identification as it is user-centric, institution-independent and portable. This resolves issues of transportability and lays the ground for future global mobile transactions and interactions (eg, short message service [SMS]-based inquires/requests/approvals/receipts/certificates, m-banking, m-learning, m-ERP (Enterprise Resource Planning), etc.) in the global knowledge economy. A user's mobile phone number, consisting of the country code, followed by area code and number, presents a number string that is unique in the world. Metadata in the PrPl-based e-portfolio system can therefore be tied to the user's mobile phone number. For example, simple metadata of a user's full name, email address and associated information can be represented as shown in Figure 3.

考試科目	資訊管理論文評析	所別	資訊管理學系	考試時間	5 月 22 日(六) 第一節

```
prpl://identity/# 16502807769 @pcb_provider.com
prpl://identity/# 16502807769 @pcb_provider.com rdf:type foaf:Person
prpl://identity/# 16502807769 @pcb_provider.com foaf:f_name "Paul"
prpl://identity/# 16502807769 @pcb_provider.com foaf:l_name "Kim"
prpl://identity/# 16502807769 @pcb_provider.com foaf:mbox ``digitalschema@gmail.com"
prpl://identity/# 16502807769 @pcb_provider.com foaf:phone ``16502807769"
```

Figure 3: Representing metadata in RDF

Furthermore, a Semantic Web infrastructure facilitates much easier search and retrieval of data and artefacts. It allows users a greater degree of control over the dissemination of the information they contain by defining what is public or private information. For example, users can make certain information, such as a list of degrees earned, classes taken or competencies achieved widely available to colleagues, potential employers and the general public as desired. More personal and specific information, such as GPA (Grade Point Average), faculty remarks, salary requirements or endorsement statements can be restricted only to potential employers who are approved with access privileges. The access privileges can be given through simple SMS approvals and set to expire if needed. Conversely, the PrPl-based e-portfolio will facilitate searching for people matching a particular criterion. For example, potential employers might wish to instantly create (ie, instead of keyword searching) a list of candidates meeting a specific set of requirements; as PrPl uses a semantic index, this type of task can be accomplished using simple queries of the pertinent information that potential candidates make available in their e-portfolios.

Conclusion

The benefits of a portfolio, and subsequently, with the advent of technology, an e-portfolio system, have been identified and extolled since a long time ago (Greenberg, 2004). Around the world, colleges, universities and governmental agencies have begun using electronic portfolios to facilitate lifelong and life-wide learning (AeP, 2008; EIfEL, 2007; Hartnell-young, Smallwood, Kingston & Hartley, 2006). However, there are yet many challenges to be overcome on the way to implementing a successful e-portfolio system that is stable, scalable and sustainable (Jafari, 2004). In this paper, a collaborative, semantically-indexed data management system, PrPl, was proposed as the platform for a future e-portfolio system. The features of this PrPl-based system which enable it to circumvent a number of issues currently faced by e-portfolio systems have also been discussed. A prototype of this system is currently being built and field tested. The potential of this system for successful implementation and application will be studied more as the project progresses further. As the development of this system parallels the development of the Semantic Web worldwide, it remains to be seen how the merging of cloud computing, the Semantic Web and the e-portfolio system will prove to revolutionise the world of e-portfolio application development, institutional implementation, and user adoption.

Most interestingly, the current strand of Semantic Web research promises a series of new possibilities applicable in education and obviously, PrPl will be harnessing the core features of the Semantic Web architecture. One apparent possibility is to make student learning artefacts more understandable by computers using the semantic web-based PrPl index database. With such e-portfolio system design, evaluating student learning outcomes (ie, products as well as the process) will become more efficient, scalable and objective. Thus, with e-portfolio objects linked to fine grained semantics and metadata, critical analyses on student learning in a longer term, examining the effect of e-portfolios on academic and professional development endeavours, and research on teacher intervention or academic program accountability will become more feasible. Subsequently, such benefits will win over resistance to use e-portfolios and more constructive and productive ways of using e-portfolios will be introduced and examined.

考試科目	資訊管理論文評析	所別	資訊管理學系	考試時間	5 月 22 日(六) 第一節

References

Alpert, S. R. (2005). Comprehensive mapping of knowledge and information resources: the case of Webster. In S. O. Tergan & T. Keller (Eds), *Knowledge and information visualisation* (pp. 220–237). Berlin, Germany: Springer.

Australian ePortfolio Project (AeP) (2008). *Australian ePortfolio project—final project report*. Retrieved February 20, 2009, from http://www.eportfoliopractice.qut.edu.au/information/report/index.jsp

Barrett, H. (1994). Technology supported assessment portfolios. *The Computing Teacher, 21*, 6, 9–12.

Bhattacharya, M. & Hartnett, M. (2007). E-portfolio assessment in higher education. *37th ASEE/IEEE Frontiers in Education Conference*, Milwaukee, WI.

Cambridge, D. (2008). Audience, integrity, and the living document: eFolio Minnestota and lifelong and lifewide learning with ePortfolios. *Computers & Education, 51*, 1227–1246.

Canada, M. (2002). Assessing e-folios in the on-line class. *New Directions for Teaching and Learning, 91*, 69–75.

Challis, D. (2005). Towards the mature e-portfolio: Some implications for higher education. *Canadian Journal of Learning and Technology, 31*(3). 49–58

Coffey, J. (2005). LEO: a concept map based course visualisation tool for instructors and students. In S. O. Tergan & T. Keller (Eds), *Knowledge and information visualisation* (pp. 285–301). Berlin, Germany: Springer.

Cotterill, S., White, A. & Currant, B. (2007). Using Web 2.0 to support PDP. *PDP-UK Newsletter, 12*, 7–8.

CVC (2009). *California virtual campus ePortfolio California*. Retrieved August 20, 2009, from http://www.eportfolioca.org/

eFolio (2009). *eFolio minnesota*. Retrieved August 20, 2009, from http://www.efoliominnesota.com/

European Institute for E-Learning (EIfEL) (2007). *Europortfolio*. Retrieved February 20, 2009, from http://www.eife-l.org/about/europortfolio

Green, K. (2008). *The 2008 Campus Computing Survey*. Retrieved August 20, 2009, from http://www.campuscomputing.net/sites/www.campuscomputing.net/files/CC2008-Executive%20Summary_3.pdf

Greenberg, G. (2004). The digital convergence: extending the portfolio model. *Educause Review, 39*, 4, 28–37.

Hartnell-Young, E., Smallwood, A., Kingston, S. & Hartley, P. (2006). Joining up episodes of lifelong learning: a regional transition project. *British Journal of Educational Technology, 36*, 6, 853–866.

Hilzensauer, W., Hornung-Prahauser, V. & Schaffert, S. (2006). Requirements for a personal development planning in e-portfolios supported by Semantic Web technology. In *6th International Conference on Knowledge Management (I-KNOW06)*, Graz, Austria.

Jafari, A. (2004). The 'Sticky' ePortfolio system: tackling challenges and identifying attributes. *Educause Review, 39*, 4, 38–49.

Johnson, L. F., Levine, A. & Smith, R. S. (2009). *Horizon Report*. Austin, TX: The New Media Consortium, 2009.

Kennedy, G., Judd, T., Churchward, A., Gray, K. & Krause, K. (2008). First year students' experiences with technology: are they really digital natives? *Australasian Journal of Educational Technology, 24*, 1, 108–122.

Kim, P. (2006). Perspectives on a visual-map-based electronic portfolio system. In A. Jafaria & C. Kaufman (Eds), *Handbook of research on eportfolios* (pp. 44–53). Hershey, PA: Idea Group Reference.

Kim, P. & Olaciregui, C. (2008). The effects of a concept map-based information display in an electronic portfolio system on information processing and retention in a fifth-grade science class covering the Earth's atmosphere. *British Journal of Educational Technology, 39*, 4, 700–714.

Lorenzo, G. & Ittelson, J. (2005a). *An overview of e-portfolios*. ELI Paper 1: 2005. Boulder: Educause Learning Initiative. Retrieved on January 5, 2009 from http://www.educause.edu/ir/library/pdf/ELI3001.pdf

Lorenzo, G. & Ittelson, J. (2005b). *An overview of institutional e-portfolios*. Retrieved July 14, 2006, from http://www.eife-l.org/publications/eportfolio/proceedings2/ep2007/papers/eportfolio/if-we-build-it-will-they-come-embedding-employability-through-eportfolio-assessment

O'Brien, K. (2006). ePortfolios as learning construction zones: provost's perspective. In A. Jafaria & C. Kaufman (Eds), *Handbook of research on eportfolios* (pp. 74–82). Hershey, PA: Idea Group Reference.

考試科目	資訊管理論文評析	所別	資訊管理學系	考試時間	5月22日(六)第一節

Paoletti, J. (2006). ePortfolio thinking: the challenge of the public research university. In A. Jafari & C. Kaufman (Eds), *Handbook of research on eportfolios* London: Idea Group Reference.

Patent, V. (2007). If we build it, will they come? Embedding employability through eportfolio assessment. *Proceedings from the European Institute for E-Learning (EIfEL) Conference 2007*. Retrieved on February 20, 2009, from http://www.eife-l.org/publications/eportfolio/proceedings2/ep2007/papers/eportfolio/if-we-build-it-will-they-come-embedding-employability-through-eportfolio-assessment

Paulson, F., Paulson, P. & Mercer, C. (1991). What makes a portfolio a portfolio? *Educational Leadership, 48*, 5, 60–63.

POMI (2009). *The Programmable Open Mobile Internet (POMI) 2020 Project*. Retrieved on February 20, 2009, from http://cleanslate.stanford.edu/pomi2020

Ran, A. & Lencevicius, R. (2007). Natural Language Query System for RDF Repositories. *Proceedings of Seventh International Symposium on Natural Language Processing*, 2007.

RINET (2009). *Rhode Island Educational Portfolio System*. Retrieved on August 20, 2009, from http://www.ri.net/RINET/products/portfolio/index.html

Schaffert, S. (2006). Semantic social software: Semantically enabled social software or socially enabled semantic web? In Proceedings of the SEMANTICS 2006 conference, pages 99–112, Vienna, Austria, November 2006. OCG.

Seong, S. W., Hangal, S., Brigham, C., Sengupta, D., Bayer, G., Seo, J. *et al* (2009). *A distributed social networking infrastructure with personalcloud butlers*. Retrieved on February 20, 2009 from http://gbayer.com/stanford/prpl/www2009_submission_872.pdf.

Sherman, G. (2006). Instructional roles of electronic portfolios. In A. Jafaria & C. Kaufman (Eds), *Handbook of research on eportfolios* (pp. 1–14). Hershey, PA: Idea Group Reference.

Sherry, A. C. & Bartlett, A. (2005). Worth of electronic portfolios to education majors: a 'two by four' perspective. *Journal of Educational Technology Systems, 33*, 4, 399–419.

Stevenson, H. J. (2006). Using ePortfolios to foster peer assessment, critical thinking, and collaboration. In A. Jafaria & C. Kaufman (Eds), *Handbook of research on eportfolios* (pp. 112–124). Hershey, PA: Idea Group Reference.

Tergan, S. L. (2005). Digital concept maps for managing knowledge and information. In S. O. Tergan & T. Keller (Eds), *Knowledge and information visualisation* (pp. 185–204). Berlin, Germany: Springer.

Tosh, D., Light, T. P., Fleming, K. & Haywood, J. (2005). Engagement with electronic portfolios: Challenges from the student perspective. *Canadian Journal of Learning and Technology, 31*(3).

Wetzel, K. & Strudler, N. (2005). The diffusion of electronic portfolios in teacher education: next steps and recommendations from accomplished users. *Journal of Research on Technology in Education, 38*, 2, 231–243.

Wiedmer, T. L. (1998). *Digital portfolios: capturing and demonstrating skills and levels of performance. (electronic validation of individual teacher performance)*. Phi Delta Kappa Inc. Retrieved February 20, 2009, from http://www.educate.vt.edu/teeps/digital.pdf

考試科目	資訊科技論文評析	所別	資訊管理學系	考試時間	5月22日(六)第一節

資訊管理學系博士班入學考試試題
考試科目：資訊科技論文評析

Google 公司於 2006 年發表 Bigtable 的架構性文件 "Bigtable: A Distributed Storage System for Structured Data," 該文件的部分內容(p.1-p.6)如附。請依所附文件內容回答下列問題。

一、詳述 Bigtable 的資料模型 (data model)。(40%)

二、詳述 Bigtable 實作(implementation) 方法。(60%)

備註	試題隨卷繳交

7-2

資訊管理學系博士班入學考試試題
考試科目：資訊科技論文評析

Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber
{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com
Google, Inc.

Abstract

Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. Many projects at Google store data in Bigtable, including web indexing, Google Earth, and Google Finance. These applications place very different demands on Bigtable, both in terms of data size (from URLs to web pages to satellite imagery) and latency requirements (from backend bulk processing to real-time data serving). Despite these varied demands, Bigtable has successfully provided a flexible, high-performance solution for all of these Google products. In this paper we describe the simple data model provided by Bigtable, which gives clients dynamic control over data layout and format, and we describe the design and implementation of Bigtable.

1 Introduction

Over the last two and a half years we have designed, implemented, and deployed a distributed storage system for managing structured data at Google called Bigtable. Bigtable is designed to reliably scale to petabytes of data and thousands of machines. Bigtable has achieved several goals: wide applicability, scalability, high performance, and high availability. Bigtable is used by more than sixty Google products and projects, including Google Analytics, Google Finance, Orkut, Personalized Search, Writely, and Google Earth. These products use Bigtable for a variety of demanding workloads, which range from throughput-oriented batch-processing jobs to latency-sensitive serving of data to end users. The Bigtable clusters used by these products span a wide range of configurations, from a handful to thousands of servers, and store up to several hundred terabytes of data.

In many ways, Bigtable resembles a database: it shares many implementation strategies with databases. Parallel databases [14] and main-memory databases [13] have achieved scalability and high performance, but Bigtable provides a different interface than such systems. Bigtable does not support a full relational data model; instead, it provides clients with a simple data model that supports dynamic control over data layout and format, and allows clients to reason about the locality properties of the data represented in the underlying storage. Data is indexed using row and column names that can be arbitrary strings. Bigtable also treats data as uninterpreted strings, although clients often serialize various forms of structured and semi-structured data into these strings. Clients can control the locality of their data through careful choices in their schemas. Finally, Bigtable schema parameters let clients dynamically control whether to serve data out of memory or from disk.

Section 2 describes the data model in more detail, and Section 3 provides an overview of the client API. Section 4 briefly describes the underlying Google infrastructure on which Bigtable depends. Section 5 describes the fundamentals of the Bigtable implementation, and Section 6 describes some of the refinements that we made to improve Bigtable's performance. Section 7 provides measurements of Bigtable's performance. We describe several examples of how Bigtable is used at Google in Section 8, and discuss some lessons we learned in designing and supporting Bigtable in Section 9. Finally, Section 10 describes related work, and Section 11 presents our conclusions.

2 Data Model

A Bigtable is a sparse, distributed, persistent multi-dimensional sorted map. The map is indexed by a row key, column key, and a timestamp; each value in the map is an uninterpreted array of bytes.

(row:string, column:string, time:int64) → string

To appear in OSDI 2006

7-3

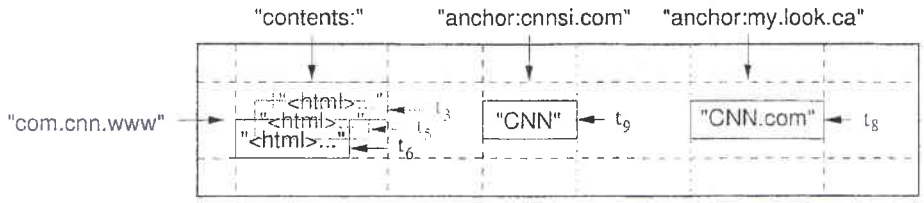

Figure 1: A slice of an example table that stores Web pages. The row name is a reversed URL. The contents column family contains the page contents, and the anchor column family contains the text of any anchors that reference the page. CNN's home page is referenced by both the Sports Illustrated and the MY-look home pages, so the row contains columns named anchor:cnnsi.com and anchor:my.look.ca. Each anchor cell has one version; the contents column has three versions, at timestamps t_3, t_5, and t_6.

We settled on this data model after examining a variety of potential uses of a Bigtable-like system. As one concrete example that drove some of our design decisions, suppose we want to keep a copy of a large collection of web pages and related information that could be used by many different projects; let us call this particular table the *Webtable*. In Webtable, we would use URLs as row keys, various aspects of web pages as column names, and store the contents of the web pages in the contents: column under the timestamps when they were fetched, as illustrated in Figure 1.

Rows

The row keys in a table are arbitrary strings (currently up to 64KB in size, although 10-100 bytes is a typical size for most of our users). Every read or write of data under a single row key is atomic (regardless of the number of different columns being read or written in the row), a design decision that makes it easier for clients to reason about the system's behavior in the presence of concurrent updates to the same row.

Bigtable maintains data in lexicographic order by row key. The row range for a table is dynamically partitioned. Each row range is called a *tablet*, which is the unit of distribution and load balancing. As a result, reads of short row ranges are efficient and typically require communication with only a small number of machines. Clients can exploit this property by selecting their row keys so that they get good locality for their data accesses. For example, in Webtable, pages in the same domain are grouped together into contiguous rows by reversing the hostname components of the URLs. For example, we store data for maps.google.com/index.html under the key com.google.maps/index.html. Storing pages from the same domain near each other makes some host and domain analyses more efficient.

Column Families

Column keys are grouped into sets called *column families*, which form the basic unit of access control. All data stored in a column family is usually of the same type (we compress data in the same column family together). A column family must be created before data can be stored under any column key in that family; after a family has been created, any column key within the family can be used. It is our intent that the number of distinct column families in a table be small (in the hundreds at most), and that families rarely change during operation. In contrast, a table may have an unbounded number of columns.

A column key is named using the following syntax: *family:qualifier*. Column family names must be printable, but qualifiers may be arbitrary strings. An example column family for the Webtable is language, which stores the language in which a web page was written. We use only one column key in the language family, and it stores each web page's language ID. Another useful column family for this table is anchor; each column key in this family represents a single anchor, as shown in Figure 1. The qualifier is the name of the referring site; the cell contents is the link text.

Access control and both disk and memory accounting are performed at the column-family level. In our Webtable example, these controls allow us to manage several different types of applications: some that add new base data, some that read the base data and create derived column families, and some that are only allowed to view existing data (and possibly not even to view all of the existing families for privacy reasons).

Timestamps

Each cell in a Bigtable can contain multiple versions of the same data; these versions are indexed by timestamp. Bigtable timestamps are 64-bit integers. They can be assigned by Bigtable, in which case they represent "real time" in microseconds, or be explicitly assigned by client

7-4

```
// Open the table
Table *T = OpenOrDie("/bigtable/web/webtable");

// Write a new anchor and delete an old anchor
RowMutation rl(T, "com.cnn.www");
rl.Set("anchor:www.c-span.org", "CNN");
rl.Delete("anchor:www.abc.com");
Operation op;
Apply(&op, &rl);
```

Figure 2: Writing to Bigtable.

```
Scanner scanner(T);
ScanStream *stream;
stream = scanner.FetchColumnFamily("anchor");
stream->SetReturnAllVersions();
scanner.Lookup("com.cnn.www");
for (; !stream->Done(); stream->Next()) {
  printf("%s %s %lld %s\n",
         scanner.RowName(),
         stream->ColumnName(),
         stream->MicroTimestamp(),
         stream->Value());
}
```

Figure 3: Reading from Bigtable.

applications. Applications that need to avoid collisions must generate unique timestamps themselves. Different versions of a cell are stored in decreasing timestamp order, so that the most recent versions can be read first.

To make the management of versioned data less onerous, we support two per-column-family settings that tell Bigtable to garbage-collect cell versions automatically. The client can specify either that only the last n versions of a cell be kept, or that only new-enough versions be kept (e.g., only keep values that were written in the last seven days).

In our Webtable example, we set the timestamps of the crawled pages stored in the contents: column to the times at which these page versions were actually crawled. The garbage-collection mechanism described above lets us keep only the most recent three versions of every page.

3 API

The Bigtable API provides functions for creating and deleting tables and column families. It also provides functions for changing cluster, table, and column family metadata, such as access control rights.

Client applications can write or delete values in Bigtable, look up values from individual rows, or iterate over a subset of the data in a table. Figure 2 shows C++ code that uses a `RowMutation` abstraction to perform a series of updates. (Irrelevant details were elided to keep the example short.) The call to `Apply` performs an atomic mutation to the Webtable: it adds one anchor to www.cnn.com and deletes a different anchor.

Figure 3 shows C++ code that uses a `Scanner` abstraction to iterate over all anchors in a particular row. Clients can iterate over multiple column families, and there are several mechanisms for limiting the rows, columns, and timestamps produced by a scan. For example, we could restrict the scan above to only produce anchors whose columns match the regular expression anchor:*.cnn.com, or to only produce anchors whose timestamps fall within ten days of the current time.

Bigtable supports several other features that allow the user to manipulate data in more complex ways. First, Bigtable supports single-row transactions, which can be used to perform atomic read-modify-write sequences on data stored under a single row key. Bigtable does not currently support general transactions across row keys, although it provides an interface for batching writes across row keys at the clients. Second, Bigtable allows cells to be used as integer counters. Finally, Bigtable supports the execution of client-supplied scripts in the address spaces of the servers. The scripts are written in a language developed at Google for processing data called Sawzall [28]. At the moment, our Sawzall-based API does not allow client scripts to write back into Bigtable, but it does allow various forms of data transformation, filtering based on arbitrary expressions, and summarization via a variety of operators.

Bigtable can be used with MapReduce [12], a framework for running large-scale parallel computations developed at Google. We have written a set of wrappers that allow a Bigtable to be used both as an input source and as an output target for MapReduce jobs.

4 Building Blocks

Bigtable is built on several other pieces of Google infrastructure. Bigtable uses the distributed Google File System (GFS) [17] to store log and data files. A Bigtable cluster typically operates in a shared pool of machines that run a wide variety of other distributed applications, and Bigtable processes often share the same machines with processes from other applications. Bigtable depends on a cluster management system for scheduling jobs, managing resources on shared machines, dealing with machine failures, and monitoring machine status.

The Google *SSTable* file format is used internally to store Bigtable data. An SSTable provides a persistent, ordered immutable map from keys to values, where both keys and values are arbitrary byte strings. Operations are provided to look up the value associated with a specified

key, and to iterate over all key/value pairs in a specified key range. Internally, each SSTable contains a sequence of blocks (typically each block is 64KB in size, but this is configurable). A block index (stored at the end of the SSTable) is used to locate blocks; the index is loaded into memory when the SSTable is opened. A lookup can be performed with a single disk seek: we first find the appropriate block by performing a binary search in the in-memory index, and then reading the appropriate block from disk. Optionally, an SSTable can be completely mapped into memory, which allows us to perform lookups and scans without touching disk.

Bigtable relies on a highly-available and persistent distributed lock service called Chubby [8]. A Chubby service consists of five active replicas, one of which is elected to be the master and actively serve requests. The service is live when a majority of the replicas are running and can communicate with each other. Chubby uses the Paxos algorithm [9, 23] to keep its replicas consistent in the face of failure. Chubby provides a namespace that consists of directories and small files. Each directory or file can be used as a lock, and reads and writes to a file are atomic. The Chubby client library provides consistent caching of Chubby files. Each Chubby client maintains a *session* with a Chubby service. A client's session expires if it is unable to renew its session lease within the lease expiration time. When a client's session expires, it loses any locks and open handles. Chubby clients can also register callbacks on Chubby files and directories for notification of changes or session expiration.

Bigtable uses Chubby for a variety of tasks: to ensure that there is at most one active master at any time; to store the bootstrap location of Bigtable data (see Section 5.1); to discover tablet servers and finalize tablet server deaths (see Section 5.2); to store Bigtable schema information (the column family information for each table); and to store access control lists. If Chubby becomes unavailable for an extended period of time, Bigtable becomes unavailable. We recently measured this effect in 14 Bigtable clusters spanning 11 Chubby instances. The average percentage of Bigtable server hours during which some data stored in Bigtable was not available due to Chubby unavailability (caused by either Chubby outages or network issues) was 0.0047%. The percentage for the single cluster that was most affected by Chubby unavailability was 0.0326%.

5 Implementation

The Bigtable implementation has three major components: a library that is linked into every client, one master server, and many tablet servers. Tablet servers can be dynamically added (or removed) from a cluster to accomodate changes in workloads.

The master is responsible for assigning tablets to tablet servers, detecting the addition and expiration of tablet servers, balancing tablet-server load, and garbage collection of files in GFS. In addition, it handles schema changes such as table and column family creations.

Each tablet server manages a set of tablets (typically we have somewhere between ten to a thousand tablets per tablet server). The tablet server handles read and write requests to the tablets that it has loaded, and also splits tablets that have grown too large.

As with many single-master distributed storage systems [17, 21], client data does not move through the master: clients communicate directly with tablet servers for reads and writes. Because Bigtable clients do not rely on the master for tablet location information, most clients never communicate with the master. As a result, the master is lightly loaded in practice.

A Bigtable cluster stores a number of tables. Each table consists of a set of tablets, and each tablet contains all data associated with a row range. Initially, each table consists of just one tablet. As a table grows, it is automatically split into multiple tablets, each approximately 100-200 MB in size by default.

5.1 Tablet Location

We use a three-level hierarchy analogous to that of a B^+-tree [10] to store tablet location information (Figure 4).

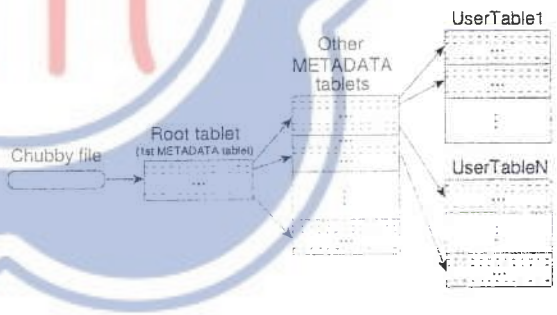

Figure 4: Tablet location hierarchy.

The first level is a file stored in Chubby that contains the location of the *root tablet*. The *root tablet* contains the location of all tablets in a special METADATA table. Each METADATA tablet contains the location of a set of user tablets. The *root tablet* is just the first tablet in the METADATA table, but is treated specially—it is never split—to ensure that the tablet location hierarchy has no more than three levels.

The METADATA table stores the location of a tablet under a row key that is an encoding of the tablet's table

identifier and its end row. Each METADATA row stores approximately 1KB of data in memory. With a modest limit of 128 MB METADATA tablets, our three-level location scheme is sufficient to address 2^{34} tablets (or 2^{61} bytes in 128 MB tablets).

The client library caches tablet locations. If the client does not know the location of a tablet, or if it discovers that cached location information is incorrect, then it recursively moves up the tablet location hierarchy. If the client's cache is empty, the location algorithm requires three network round-trips, including one read from Chubby. If the client's cache is stale, the location algorithm could take up to six round-trips, because stale cache entries are only discovered upon misses (assuming that METADATA tablets do not move very frequently). Although tablet locations are stored in memory, so no GFS accesses are required, we further reduce this cost in the common case by having the client library prefetch tablet locations: it reads the metadata for more than one tablet whenever it reads the METADATA table.

We also store secondary information in the METADATA table, including a log of all events pertaining to each tablet (such as when a server begins serving it). This information is helpful for debugging and performance analysis.

5.2 Tablet Assignment

Each tablet is assigned to one tablet server at a time. The master keeps track of the set of live tablet servers, and the current assignment of tablets to tablet servers, including which tablets are unassigned. When a tablet is unassigned, and a tablet server with sufficient room for the tablet is available, the master assigns the tablet by sending a tablet load request to the tablet server.

Bigtable uses Chubby to keep track of tablet servers. When a tablet server starts, it creates, and acquires an exclusive lock on, a uniquely-named file in a specific Chubby directory. The master monitors this directory (the *servers directory*) to discover tablet servers. A tablet server stops serving its tablets if it loses its exclusive lock: e.g., due to a network partition that caused the server to lose its Chubby session. (Chubby provides an efficient mechanism that allows a tablet server to check whether it still holds its lock without incurring network traffic.) A tablet server will attempt to reacquire an exclusive lock on its file as long as the file still exists. If the file no longer exists, then the tablet server will never be able to serve again, so it kills itself. Whenever a tablet server terminates (e.g., because the cluster management system is removing the tablet server's machine from the cluster), it attempts to release its lock so that the master will reassign its tablets more quickly.

The master is responsible for detecting when a tablet server is no longer serving its tablets, and for reassigning those tablets as soon as possible. To detect when a tablet server is no longer serving its tablets, the master periodically asks each tablet server for the status of its lock. If a tablet server reports that it has lost its lock, or if the master was unable to reach a server during its last several attempts, the master attempts to acquire an exclusive lock on the server's file. If the master is able to acquire the lock, then Chubby is live and the tablet server is either dead or having trouble reaching Chubby, so the master ensures that the tablet server can never serve again by deleting its server file. Once a server's file has been deleted, the master can move all the tablets that were previously assigned to that server into the set of unassigned tablets. To ensure that a Bigtable cluster is not vulnerable to networking issues between the master and Chubby, the master kills itself if its Chubby session expires. However, as described above, master failures do not change the assignment of tablets to tablet servers.

When a master is started by the cluster management system, it needs to discover the current tablet assignments before it can change them. The master executes the following steps at startup. (1) The master grabs a unique *master* lock in Chubby, which prevents concurrent master instantiations. (2) The master scans the servers directory in Chubby to find the live servers. (3) The master communicates with every live tablet server to discover what tablets are already assigned to each server. (4) The master scans the METADATA table to learn the set of tablets. Whenever this scan encounters a tablet that is not already assigned, the master adds the tablet to the set of unassigned tablets, which makes the tablet eligible for tablet assignment.

One complication is that the scan of the METADATA table cannot happen until the METADATA tablets have been assigned. Therefore, before starting this scan (step 4), the master adds the root tablet to the set of unassigned tablets if an assignment for the root tablet was not discovered during step 3. This addition ensures that the root tablet will be assigned. Because the root tablet contains the names of all METADATA tablets, the master knows about all of them after it has scanned the root tablet.

The set of existing tablets only changes when a table is created or deleted, two existing tablets are merged to form one larger tablet, or an existing tablet is split into two smaller tablets. The master is able to keep track of these changes because it initiates all but the last. Tablet splits are treated specially since they are initiated by a tablet server. The tablet server commits the split by recording information for the new tablet in the METADATA table. When the split has committed, it notifies the master. In case the split notification is lost (either

because the tablet server or the master died), the master detects the new tablet when it asks a tablet server to load the tablet that has now split. The tablet server will notify the master of the split, because the tablet entry it finds in the METADATA table will specify only a portion of the tablet that the master asked it to load.

5.3 Tablet Serving

The persistent state of a tablet is stored in GFS, as illustrated in Figure 5. Updates are committed to a commit log that stores redo records. Of these updates, the recently committed ones are stored in memory in a sorted buffer called a *memtable*; the older updates are stored in a sequence of SSTables. To recover a tablet, a tablet server

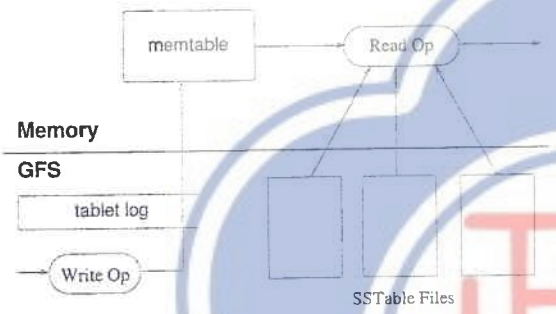

Figure 5: Tablet Representation

reads its metadata from the METADATA table. This metadata contains the list of SSTables that comprise a tablet and a set of a redo points, which are pointers into any commit logs that may contain data for the tablet. The server reads the indices of the SSTables into memory and reconstructs the memtable by applying all of the updates that have committed since the redo points.

When a write operation arrives at a tablet server, the server checks that it is well-formed, and that the sender is authorized to perform the mutation. Authorization is performed by reading the list of permitted writers from a Chubby file (which is almost always a hit in the Chubby client cache). A valid mutation is written to the commit log. Group commit is used to improve the throughput of lots of small mutations [13, 16]. After the write has been committed, its contents are inserted into the memtable.

When a read operation arrives at a tablet server, it is similarly checked for well-formedness and proper authorization. A valid read operation is executed on a merged view of the sequence of SSTables and the memtable. Since the SSTables and the memtable are lexicographically sorted data structures, the merged view can be formed efficiently.

Incoming read and write operations can continue while tablets are split and merged.

5.4 Compactions

As write operations execute, the size of the memtable increases. When the memtable size reaches a threshold, the memtable is frozen, a new memtable is created, and the frozen memtable is converted to an SSTable and written to GFS. This *minor compaction* process has two goals: it shrinks the memory usage of the tablet server, and it reduces the amount of data that has to be read from the commit log during recovery if this server dies. Incoming read and write operations can continue while compactions occur.

Every minor compaction creates a new SSTable. If this behavior continued unchecked, read operations might need to merge updates from an arbitrary number of SSTables. Instead, we bound the number of such files by periodically executing a *merging compaction* in the background. A merging compaction reads the contents of a few SSTables and the memtable, and writes out a new SSTable. The input SSTables and memtable can be discarded as soon as the compaction has finished.

A merging compaction that rewrites all SSTables into exactly one SSTable is called a *major compaction*. SSTables produced by non-major compactions can contain special deletion entries that suppress deleted data in older SSTables that are still live. A major compaction, on the other hand, produces an SSTable that contains no deletion information or deleted data. Bigtable cycles through all of its tablets and regularly applies major compactions to them. These major compactions allow Bigtable to reclaim resources used by deleted data, and also allow it to ensure that deleted data disappears from the system in a timely fashion, which is important for services that store sensitive data.

6 Refinements

The implementation described in the previous section required a number of refinements to achieve the high performance, availability, and reliability required by our users. This section describes portions of the implementation in more detail in order to highlight these refinements.

Locality groups

Clients can group multiple column families together into a *locality group*. A separate SSTable is generated for each locality group in each tablet. Segregating column families that are not typically accessed together into separate locality groups enables more efficient reads. For example, page metadata in Webtable (such as language and checksums) can be in one locality group, and the contents of the page can be in a different group: an ap-