Improving Data Availability in Cloud Storage with Efficient Replication

Julia Myint, Thinn Thu Naing
University of Computer Studies, Yangon, Myanmar
juliamyint@gmail.com, thinnthu@gmail.com

Abstract

Data availability is one of the key requirements for cloud storage system. Data replication has been widely used as a mean of increasing the availability in traditional distributed databases, peer-to-peer (P2P) systems, and grid systems. These strategies were all developed for specific platforms and application types and have been tailored to the characteristics of the underlying system architectures and application requirements. Cloud systems differ from these previous frameworks in that they are designed to support large numbers of customer-oriented applications, each with different quality of service (QoS) requirements and resource consumption characteristics. Aiming to provide high data availability, and improve performance and load balancing of cloud storage, efficient replication management scheme is proposed. In replication management, the system provides optimum replica number as well as weighting and balancing among the storage server nodes and experimental results prove that the proposed system can improve data availability depending on the expected availability and failure probability of each node in PC cluster.

Keyword – availability; cloud storage; Hadoop Distributed File System; replication management

1. Introduction

Cloud computing promises to increase the velocity with which applications are deployed, increase innovation, and lower costs, all while increasing business agility. With the cloud computing technology, users use a variety of devices, including PCs, laptops, smart phones, and PDAs to access programs, storage, and application-development platforms over the Internet, via services offered by cloud computing providers. Advantages of the cloud computing technology include cost savings, high availability, and easy scalability. [3]

Companies such as Google, Amazon and Microsoft have been building massive data centres over the past few years. Spanning geographic and administrative domains, these data centres tend to be built out of commodity desktops with the total number of computers managed by these companies being in the order of millions. Additionally, the use of virtualization allows a physical node to be presented as a set of virtual nodes resulting in a seemingly inexhaustible set of computational resources. By leveraging economies of scale, these data centres can provision CPU, networking, and storage at substantially reduced prices which in turn underpins the move by many institutions to host their services in the cloud. [3]

Cloud storage (or data storage as a service) is the abstraction of storage behind an interface where the storage can be administered on demand. Further, the interface abstracts the location of the storage such that it is irrelevant whether the storage is local or remote (or hybrid). Cloud storage infrastructures introduce new architectures that support varying levels of service over a potentially large set of users and geographically distributed storage capacity.

The rest of this paper is organized as follows. In the next section, the related papers with our paper are discussed. In section 3 and 4, system framework and high availability for cloud storage system are presented. In section 5, replication management scheme is demonstrated. Then the proposed system is evaluated in section6. Finally, the paper is concluded.

2. Related Work

There are large amount of researches in the design of cloud storage. Many of these researches are file system based storage system such as GFS[12] and HDFS[1]. These architectures are master-slave routing paradigm. In those storage systems, replication management is performed by using default replica number.

Y.V.Lokeshwari and et al. [16] proposed optimized cloud storage with high throughput Deduplication approach. Energy efficient and resilient data center storage is described in [9]. Inaddition, Xiaofei Zhang and Lei Chen [14] studied fault tolerance strategies for durable storage on the Cloud. Hoang Tam Vo and et al. [5] proposed ecStore, a highly elastic distributed storage system with range-query and transaction support. The design of ecStore is based on distributed index structure called BATON (Balanced Tree Overlay Network) [8]. Despite the effect of single master node failure, it is organized into peer-to-peer virtual network structure on storage nodes. EcStore also applied data migration technique to balance the load of storage node and it needs to restructure the BATON whenever load unbalanced. Other cloud storage systems that used data migration for load balancing are Dynamo [4], Pnuts[2] and Cassandra[6].

Replication management has been active research issue in storage system. Qingsong Wei [11]

proposed a cost-effective replication management scheme for cloud storage cluster. In that paper, blocking probability is used as a criterion to place replicas among data nodes to reduce access skew, so as to improve load balance and parallelism. In [15], cloud storage is designed based on hybrid of replication and data partitioning. Two level DHT approach is proposed for widely distributed cloud storage.

3. System Framework

The proposed system architecture consists of three layers, i.e., application layer, storage client layer and PC cluster. The PC cluster layer provides the hardware and storage devices for large scale data storage. The application layer provides interfaces for clients who store their files, databases, and multimedia data and so on. The second layer is an interface of application layer and storage devices. The access process in cloud storage system is illustrated in figure 1.

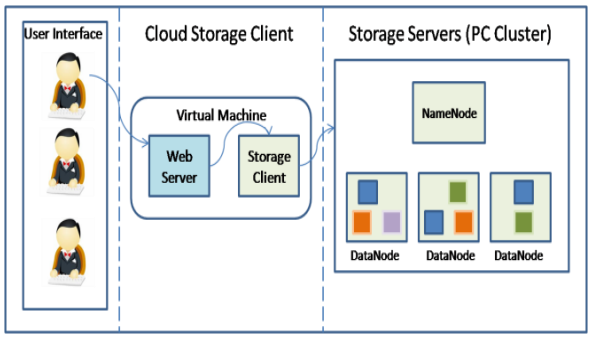

Figure 1: Access process in cloud storage infrastructure

3.1 Cloud Application

In cloud technology, users browse applications through browsing interfaces. The applications may involve data storage and file retrieval applications. Then requests arrive at the web server, and if requests are data storage requests, they are forwarded to cloud storage clients by a request controller. After that, clients interact with storage servers called PC cluster to fetch requested files.

3.2 Cloud Storage Client

Cloud storage client is an interface between user interface and storage servers. User applications access the storage servers using the storage client which is a code library that exports the cloud storage client. When an application reads a file, the storage client first asks the Name node for the list of Data nodes that host replicas of the blocks of the file. It then contacts a Data node directly and requests the transfer of the desired block. When a client writes, it first asks the Name node to choose Data nodes to host replicas.

3.3 PC Cluster

The designed storage system is based on PC cluster for cloud. The PC cluster consists of a single Name node and a number of Data nodes, usually one per node in the PC cluster, which manages storage attached to the nodes that they run on. In the proposed system, the Name node manages the whole PC cluster. It also executes file system namespace operations like opening, closing, and renaming files and directories. It also determines the mapping of blocks to Data nodes. The Data nodes are responsible for serving read and write requests from the clients. The Data nodes also perform block creation, deletion and replication upon instruction from the Name node.

4. Replication Management Scheme

Data replication has been widely used as a mean of increasing the data availability of distributed storage systems in which failures are no longer treated as exceptions. It is clear that the reliability of a system will generally increase as the number of replicas or the degree of replication increases since more replicas will be able to mask more failures. However, it is a key issue how the degree of replication will affect system availability [13].

The proposed replication management scheme is subdivided into three processes. To achieve high availability, availability constraints are considered in three processes of replication management as shown in figure 2. In the availability constraints, it takes not only the file availability but also storage server availability into account. It is assumed that the file availability level is given by Cloud storage user and the highest availability constraint is set to 1.

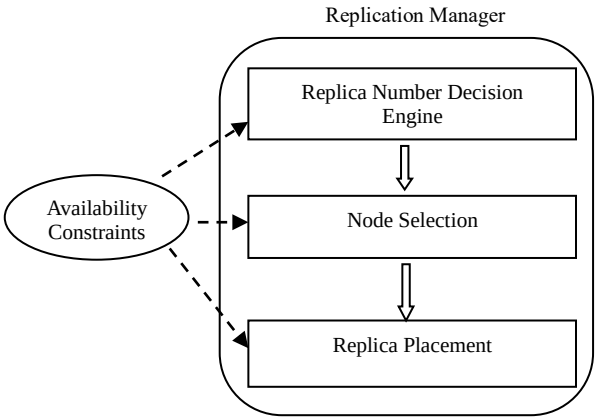

Figure 2: Three processes of proposed replication management scheme

4.1 System Model

The cloud storage system is implemented with PC cluster system. It is composed of N independent heterogeneous nodes storing of M different blocks $b_1,b_2,\ldots,b_M$. We model failure

probability of data node S_i as f_i, the optimum replica number of a data block as R_{opt}. The availability of the system, α is defined as the fraction of time that the system is available for serving user requests. The system can be in one of the two states: the normal state or the idle state.

In normal state, at least one data node is available for serving user requests. However, idle state, P_0, is the state in which all data nodes containing the data block b_j have failed.

4.2 Replica Number Decision Engine

When replica number reaches a certain point, the file availability is equal to 1, and adding more replicas will not improve the file availability any more. The lower the node failure ratio, the less replica number it requires for file availability to reach 1. Therefore, we can only maintain minimum replicas to ensure a given availability requirement [13].

With replica number increasing, the management cost including storage and network bandwidth will significantly increase. In a cloud storage system, the network bandwidth resource is very limited and crucial to overall performance. Too much replicas may not significantly improve availability, but bring unnecessary spending instead.

With attention to this, we developed a model to express availability as function of replica number which is adapted from Primary Site Approach described in [13]. This model is used to determine how much minimal replica should be maintained to satisfy availability requirement as shown in equation 1.

$$\alpha \leq \min(1-\Pi_{i=1}^{R0} f_i,\ 1-\Pi_{i=1}^{R1} f_i, \ldots, 1-\Pi_{i=1}^{Rn} f_i) \qquad (1)$$

According to the equation (1), Name node calculates optimum replica number R_n to satisfy expected availability with average data node failure probability f_i .In case of data node failure and current replica number of a block is less than R_n , additional replicas will be created into data node cluster.

4.3 Node Selection

In the proposed system, there is an assumption that each data node has its own specifications such as memory, CPU, disk space, failure probability. Therefore, the proposed strategy takes these specifications into account to choose the Data nodes for storage. Each node has its own weight which is the function of the total storage capacity and node availability shown in equation 2 and 3.

$$DN_{weight} = DN_{DiskSpace} \times DN_{avail} \qquad (2)$$

$$DN_{avail} = 1 - DN_f \qquad (3)$$

In equation 2 and 3, DN_{weight} means the weight of Data node and $DN_{DiskSpace}$ and DN_{avail} imply the storage capacity and availability respectively. In equation 3, DN_f is failure probability of that Data node. For example, if the capacity of drive in PC is 80 gigabytes and failure probability is 0.2, the weight of that PC node is 64. According to the weight of each node, binary weighted tree is applied to search highly available nodes in the PC cluster. Binary tree offers more flexibility in the ways in which a system can be reorganized.

An example binary tree illustrating result of this process for a system with six PCs is shown in figure 3. Whenever a new data block is added to the storage cluster, higher weighted nodes are selected by using binary tree and the list of node id to store the replicas is returned to the client. According to the list, the client stores the data block on storage cluster. The proposed data placement algorithm is shown in figure 4. The Name node applies the proposed data placement algorithm to find the list of Data nodes in the PC cluster.

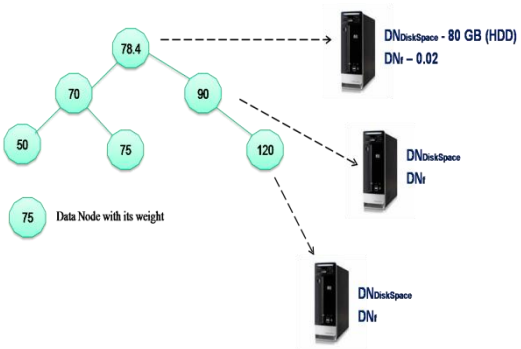

Figure 3. A binary tree for a cluster with six PCs

4.4 Replica Placement

After determining the number of replicas the system should maintain to satisfy availability requirement, we will consider how to place these replicas efficiently to maximize performance and load balancing. In this paper, efficient data placement algorithm is used to place replicas among data nodes. The replica placement algorithm is shown in figure 4.

ReplicaPlacement Algorithm
Input : failure probability $\mathbf{P_f}$, availability **α**, data block **b**
Output : List of DataNodes **DN**
Begin
ReplicaCount = Calculate using equation (2)
i ← 0
SelectedNodes = null
While (i < ReplicaCount)
DN[i] = choose largest weight node in PC cluster
Calculate weight of each node using equation (3) and (4)
Remove selected node **DN[i]**
i ← i+1
End While
Return **DN**
End

Figure 4. Replica placement algorithm

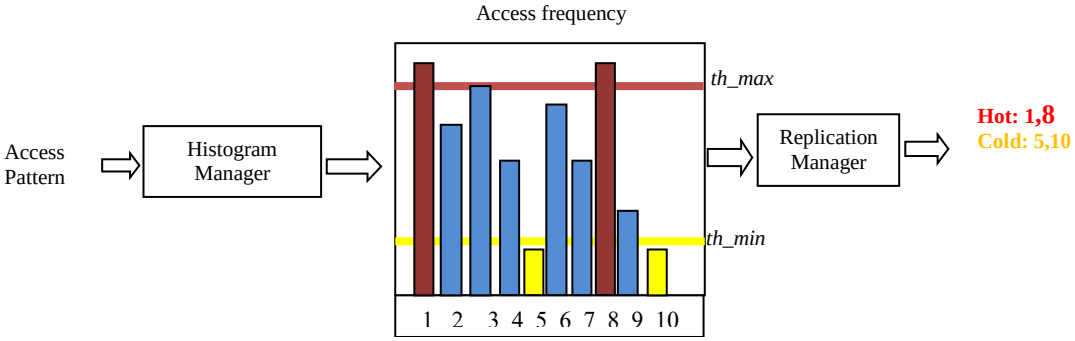

Figure 5. Process of Control Strategy

4.5 Control Strategy

In cloud environment, workloads and access pattern may change frequently. To adaptively satisfy availability and load balance according to dynamic environment in terms of node failure and access pattern changes, dynamic replication based on access frequency is proposed. The main aim is to improve elasticity of cloud storage system by reducing and increasing storage space in cloud storage.

By monitoring the workload changes and access frequency for each file in current window T_i, the system dynamically adjusts replica number and replica location in next time window T_{i+1} with updated parameters input from Data nodes. The system not only maintains reasonable replica number in the system, but also dynamically adjusts replica number and replica location according to access frequency. During the run time, some data file will have much higher access frequencies than the others due to skewed access patterns.

In particular, data access log is used to approximately estimate the access frequency of each data file. In the proposed system, two thresholds are used to identify minimum and maximum access frequency. The parameters *th_min* and *th_max* are thresholds for minimum access times and maximum access times. In case of access times for a particular data file more than *th_max*, the file is identified as hot data and it is replicated more than initial replication factor. Similarly, the file in which access frequency is less than *th_min* is considered to reduce replication factor. Figure 5 shows control strategy of the proposed system.

4.6 Process Flow of the Scheme

In the proposed storage system, the client contacts Name node with availability setting and file name. The Name node inserts the file name into the file system hierarchy, calculates the replication factor based on equation 1 and quickly searches the Data node list to obtain a list of Data nodes to be stored for each data block. The Name node responds to the client request with a list of Data nodes for each block, the destination data blocks and replication factor. Then the client writes each block of data to selected Data nodes. The process flow is described in figure 6.

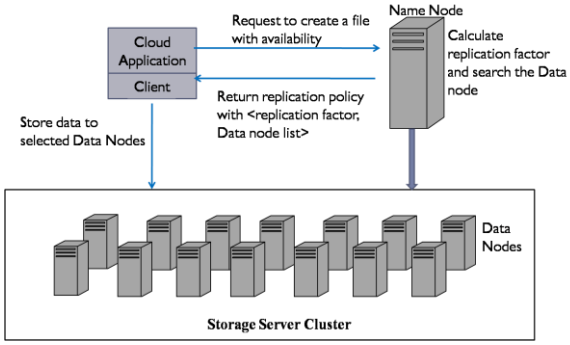

Figure 6. Process flow of the scheme

5. Evaluation

The proposed cloud storage system is implemented with Hadoop storage cluster (HDFS). Hadoop was chosen for several reasons. First of all, Hadoop distributed file system is the most common file system used in cloud application. Secondly, Hadoop is open source, making it easier to obtain, profile, and modify when necessary. Thirdly, it is designed for commodity hardware, significantly lowering the expense of building a research cluster. An example of Hadoop architecture is shown in figure 7.

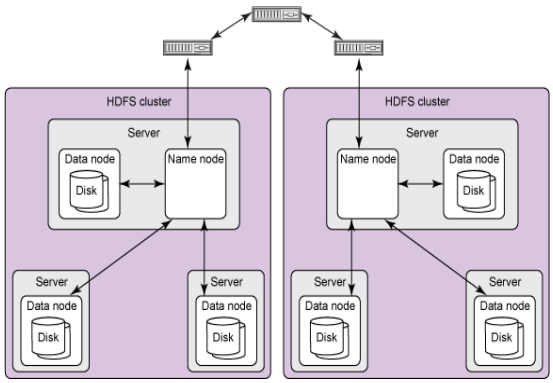

Figure 7. Hadoop Cluster Architecture

In HDFS, the replication factor is configurable per file and can be changed later. However, HDFS does not provide policy to determine the replication factor. In proposed

replication management, optimum replica number is determined before data is stored to storage servers and the optimum replica number can recover not only failure probability but also expected availability.

5.1. Storage Servers Setup

The tested cluster consists of five heterogeneous nodes, whose parameters are summarized in Table 1. All nodes are connected with 1 Gbps Ethernet network. In each node, Ubuntu server 10.10 is installed. Java version is 1.6.2 and Hadoop version is 0.20.2. The size of HDFS block is 64 MB.

Table 2. Five nodes in storage server cluster.

Node	CPU Model	CPU (Hz)	Memory	Disk Space
Name Node	Pentium ® Dual-Core	2.00 GHz	2GB	250 GB
DataNode1	Pentium P4	1.5 GHz	512 MB	80GB
DataNode2	Pentium P4	1.5 GHz	512 MB	80GB
DataNode3	Pentium P4	1.5 GHz	512 MB	80GB
DataNode4	Pentium P4	1.5 GHz	512 MB	80GB

5.2. Data Distribution over the Cluster

The original HDFS is evaluated with the proposed storage cluster without using equation 1. During the evaluation, 10 files are stored in the cluster. Each file has 1000 MB size and the total storage is about 9.765625 GB. In HDFS, the replication factor has to be set before the data is stored and it is set to 1, 2 and 3. The figure 8 shows the distribution of storage in the cluster.

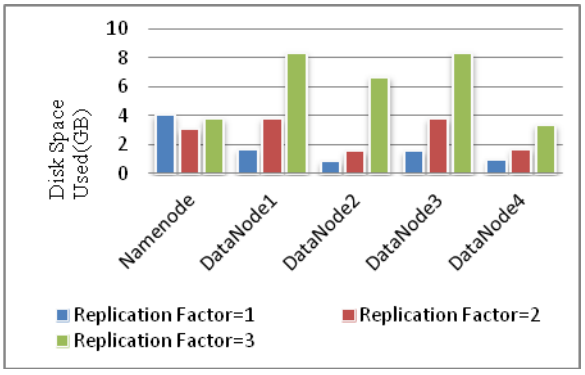

Figure 8. Data Distribution of HDFS Cluster

5.3. Analytical Result of Data Availability

To analyze the data availability of the proposed system, an analytical model for the proposed system is considered by varying the parameters shown in table 3.

Table 3. Parameters used in analysis.

Parameters	Description	Values
p	Failure probability of a single machine	0.01, 0.1,0.2
n	Number of machines in the cluster	10, 30, 60
r	Number of replicas for each block	equation (1)
b	Total number of data blocks in the cluster	80GB*1024*n/ 3*64
α	Expected availability	0.99

In the proposed system, since the system is designed for a storage system using inexpensive computer machines, these machines may fail down. We assume machine failures are dependent on failure probability. In our analysis, data availability is modeled as equation 4.

$$P_{avail(n,p,b,r)} = \sum_{f=0}^{n} \binom{n}{f} * p^{f} * (1-p)^{(n-f)} * \left(1 - \binom{f}{r}/\binom{n}{r}\right)^{b} \quad (4)$$

To analyze the availability of proposed system, equation 4 is used with the variation of four parameters n, p, α, r for the values described in table 3. In equation 4, $\binom{f}{r}$ and $\binom{n}{r}$ are combinational choice functions**.** In the proposed system, replica number r is computed by using equation 1. According to the storage servers setup in section 5.1, the disk space of each Data nodes is 80 GB and the block size is 64 MB. Therefore, each machine has $(80*1024)/64/r$ blocks. For simplicity, the expected availability α is set as 0.99 for all cluster size. The results of analysis are shown in figure 9.

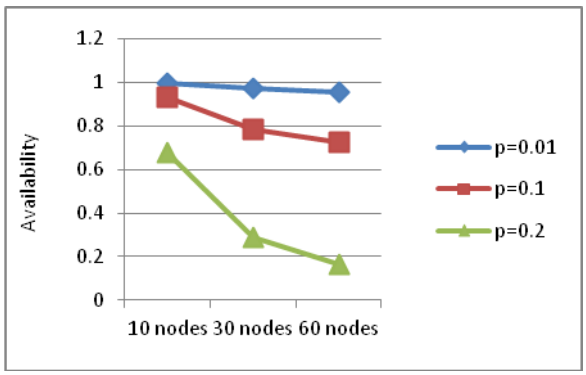

Figure 9. Availability Comparison

According to figure 9, the availability decreases steadily with larger cluster size. Therefore, the proposed system can achieve high availability in

lower failure probability and small cluster size.

6. Conclusion

Cloud computing has been one of the main trends in IT industry over the last few years. In cloud computing, the infrastructure-level storage is an essential component. As the system is designed for a storage system using inexpensive computer machines, these machines may fail down and load unbalance which can be slow down and longer data retrieval time. Consequently, we propose replication management scheme for choosing optimum replica number and load balancing. As can be seen from experimental results, the proposed replication management scheme is able to adapt failure probability and can achieve high data availability.

References

[1] Borthakur. "The Hadoop Distributed File System: Architecture and Design". *The Apache Software Foundation*, 2007.

[2] F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni. "Pnuts: Yahoo!'s Hosted Data Serving Platform",*In VLDB*, 2008.

[3] Furht, A.Escalante, *Handbook of Cloud Computing*, ISBN 978-1-4419-6523-3, Springer Science+Business Media, *LLC* 2010.

[4] G. Decandia, D. Hastorun, M. Jampani, G. Kakulapati,A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall and W. Vogels. "Dynamo: Amazon's Highly Available Key-value Store", *In SOSP*, 2007.

[5] H. T. Vo, C. Chen , B.C. Oo, "Towards Elastic Transactional Cloud Storage with Range Query Support", *International Conference on Very Large Data Bases* , Singapore, September 13-17 2010.

[6] H. V. Jagadish, B. C. Ooi, and Q. H. Vu, "BATON: A Balanced Tree Structure for Peer-to-Peer Networks". *In VLDB*, 2005.

[7] J. Wu1 and et. al., "Recent Advances in Cloud Storage*", Proceedings of the Third International Symposium on Computer Science and Computational Technology(ISCSCT '10)*, China, 14-15,August 2010, pp. 151-54.

[8] Lakshman, P. Malik, "Cassandra - A Decentralized Structured Storage System*", ACM SIGOPS Operating Systems Review, Volume 44 Issue 2*, April 2010.

[9] Lin and et.al., " eStore: Energy Efficient and Resilient Data Center Storage", *IEEE International Conference on Cloud and Service Computing*, Hong Kong, China, Dec 2011,

[10] P. Jalote, *Fault Tolerence in Distributed Systems*, ISBN 0-13-301367-7, A Pearson Education Company, 1994.

[11] Q. Wei and et. al., "CDRM: A Cost-effective Dynamic Replication Management Scheme for Cloud Storage Cluster*", IEEE International Conference on Cluster Computing*, 2010.

[12] S. Ghemawat, H. Gobioff, S. T. Leung, "The Google File System", *Proceedings of 19th ACM Symposium on Operating Systems Principles(SOSP 2003)*, New York USA, October, 2003.

[13] Sage. A. W. Scott . L.B. Ethan and et.al., "Ceph: A Scalable, High-Performance Distributed File System", *Proceeding of 7th conference on operating system design and implementation (OSDI'06)*, November 2006.

[14] X. Zhang and L. Chen, "Fault Tolerance Study for Durable Storage on the Cloud", *IEEE International Conference on Cloud and Service Computing*, Hong Kong, China, Dec 2011.

[15] Y. Ye and et. al., "Cloud Storage Design Based on Hybrid of Replication and Data Partitioning", *16th International Conference on Parallel and Distributed Systems*, 2010.

[16] Y.V. Lokeshwari and et.al., " Optimized Cloud Storage with High Throughput Deduplication Approach", *IEEE International Conference on Cloud and Service Computing*, Hong Kong, China, Dec 2011.

[17] Z.E.D.G. Eines, "Quality of Availability for Widely Distributed and Replicated Content Stores", *Ph.D Thesis*, June 2004.