Towards Transparent Personal Content Storage in Multi-Service Access Networks

Koert Vlaeminck, Tim Wauters, Filip De Turck, Bart Dhoedt, Piet Demeester

Ghent University, Dept. of Information Technology - IBBT - IMEC
Gaston Crommenlaan 8 bus 201, B-9050 Gent, Belgium
Tel.: +32 (0)9 331 49 42, Fax.: +32 (0)9 331 48 99
koert.vlaeminck@intec.ugent.be

Abstract. Anytime, anywhere and anyhow access to personalized services requires the complete decoupling of devices for accessing the service and the supporting personal data storage. When deploying such transparent personalized services, an important question that needs answering is where to install the storage servers. In this respect, this paper considers the deployment of a personal content storage service in multi-service access networks. The storage server placement problem is formulated as a binary integer linear programming (BILP) problem and a heuristic storage server placement algorithm (SSPA) is presented and evaluated. First it is assumed that servers do not fail. Consequently, the problem formulation is extended to include replication and striping and both BILP and heuristic methods are modified to cope with the additional constraints. The extended SSPA heuristic is used to analyze several resilience and striping scenarios. It is shown that the SSPA produces close to optimal results and is very efficient for optimizing server placement in personal content storage deployments.

Keywords. Distributed Storage, Server Placement, Resilience, Access Network, Personal Content

1 Introduction

In a converged media world consumers increasingly call the shots. No longer a captive, mass media audience, today's consumer is unique, demanding, and engaged. He wants anytime, anywhere and anyhow access to a personalized experience, generates his own content, mixes it, and shares it on a growing number of social networks [1]. Transparent anytime, anywhere access to such a personalized media experience requires the complete decoupling of devices for accessing the data and the actual data storage. In pursuing transparency, one important question that needs answering is where to install the storage servers. A major opportunity of emerging, converged, multi-service IP access and aggregation networks [2] is providing consumers with transparent storage, enabling anytime, anywhere access to a personal data collection, consisting of both *acquired* and *created* content.

Storage of acquired content (e.g. a digital movie and music collection) has many similarities with content distribution networks (CDNs), where content is replicated to regional servers at the edge of the network in order to tackle the performance issues of a classical central server based approach [3]. Acquired content is typically read-only and relatively popular, making it an ideal fit for content distribution networks.

Storage of created content (e.g. personalized play lists or even remixes, digital photo albums and home videos) is entirely different. Because of its read/write nature, delay is much more important and replication or caching close to the consumer a must. Furthermore, created personal content is typically a lot less popular—although the ability to share created content with (a limited amount of) other users is still desirable—and requires an ever increasing storage capacity. Where CDNs are designed to distribute a limited amount of very popular content, a (created) personal content storage service stores a huge amount of relatively unpopular content. CDN-based systems assign storage capacity per *item*, while a personal content storage service assigns capacity per *user*.

This paper presents and evaluates an algorithm for determining the best locations for deploying a storage service supporting created content, minimizing the deployment cost while guaranteeing customer satisfaction. Both access and aggregation nodes are candidate storage server locations. Servers have limited storage- and read/write capacity and can only serve a limited number of simultaneous users. Furthermore, the access and aggregation network only has limited bandwidth assigned to the storage service. Within this restricted environment, consumers expect a guaranteed minimum throughput and maximum delay for accessing their content.

After a discussion of related work in Section 2, the remainder of this paper is structured as follows: First, Sections 4 to 6 assume that system crashes and disasters are nonexistent. In that case, no redundancy is required and each data item is stored only once. These sections extend earlier work on dimensioning *read-mostly* data storage in the access and aggregation network to support *read-write* content [4]. Both a Binary Integer Linear Programming (BILP) [5] and a heuristic solution are presented. The heuristic Storage Server Placement Algorithm (SSPA) is evaluated by comparing its placement decisions to the optimal solution, acquired from a direct implementation of the BILP formulation. After that, Section 7 studies the more realistic situation where servers do fail. Both BILP solution and SSPA heuristic are extended to include replication and striping and some simulation results, illustrating the requirements for striping and resilience in terms of number of server locations and total accumulated bandwidth on all network links, are presented. Finally Section 8 summarizes the main results of this paper.

2 Related work

Many distributed filesystems exist today, ranging from client-server systems such as NFS [6], AFS [7] and Coda [8] over cluster filesystems (e.g. Lustre [9],

GPFS [10] and the Google File System [11]) to global scale peer-to-peer filesystems (e.g. OceanStore [12], FARSITE [13] and Pangaea [14]).

None of the above filesystems were designed for large-scale deployment in a service aware access and aggregation network environment. However, two of the peer-to-peer filesystems seem good candidates for this purpose: OceanStore and Pangaea. One of OceanStore's design goals is support for nomadic data using a *promiscuous caching* policy, allowing data to be cached anytime, anywhere. OceanStore uses introspection for replica management, optimizing the number and location of replicas based on observation of the access requests. Pangaea, on the other hand, uses a *pervasive replication* mechanism that replicates data wherever and whenever it is accessed.

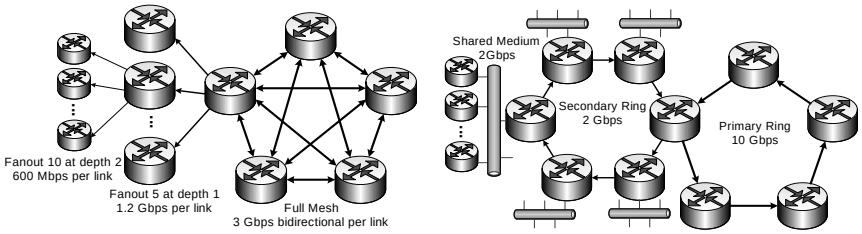
Guaranteeing fast access to a distributed filesystem, requires replication on nodes close to the user: Pangaea, with its pervasive replication mechanism, supports this by design. Another interesting replication mechanism is *fluid replication*, which creates a replica on a nearby node when the connection quality between a client and its current server becomes poor [15]. Whichever replication mechanism is chosen, a minimum set of replicas should be maintained at all times in order to ensure reliability.

Before deciding on how and where data is replicated, an important question that needs answering is where to install the storage servers. This paper investigates how servers can be deployed in the access and aggregation network, supporting fast and reliable personal content storage, minimizing cost, while guaranteeing user satisfaction. In literature, server placement was already discussed in a web server environment [16] and for the placement of regional servers for content distribution networks [3]. However, as discussed in the introduction, personal content storage, supporting created content, has different requirements.

3 Use Cases

Throughout this paper, the two typical access and aggregation network topologies were selected as use cases: a mesh of trees topology, used in DSL deployment, and a ring of rings topology, used in cable Internet access deployment, both consisting of 250 access nodes. In both topologies, all nodes are assumed to be service-aware and thus candidate locations for installing storage servers.

The mesh of trees topology is depicted in Fig. 1(a). The 250 access nodes are at the leaf nodes of five trees of depth two, aggregated per ten at depth two and per five at depth one. The available bandwidth between the access nodes and the first aggregation node is 600 Mbps. Those aggregation nodes are connected to the root of a tree, each by a 1.2 Gbps link. The five trees are interconnected by a full mesh of 3 Gbps links. The ring of rings topology, is depicted in Fig. 1(b). It consists of five unidirectional six-node secondary rings of 2 Gbps. One node connects the secondary ring to a primary ring of 10 Gbps (also unidirectional). The other five secondary ring nodes each connect to ten access nodes trough a shared medium of 2 Gbps, resulting in a total of 250 access nodes.



(a) (b)

Fig. 1: Topology (a): mesh of trees access and aggregation network topology, used for DSL deployment. Topology (b): ring of rings access and aggregation network topology, used for cable Internet access deployment.

4 Model description

This paper studies the deployment of a personal content storage service, supporting created content, in the access and aggregation network. The goal is to minimize the number of server locations, while guaranteeing fast access from any access node. Before discussing the storage server placement algorithm in the next section, this section introduces some definitions and formalizes the problem description by presenting it as a Binary Integer Linear Programming (BILP) problem [5].

Define $N = \{N_1, \ldots, N_n\}$ as the set of all nodes and $A = \{A_1, \ldots, A_m\}$ as the set of access nodes ($A \subset N$). Furthermore, define $E = \{E_1, ..., E_o\}$ as the set of all edges. An edge $E_l \in E$ represent a simplex link in the network. Delay over E_l is defined as c_l and its bandwidth as b_l. Finally define $U = \{U_1, \ldots, U_p\}$ as the set of all users.

Server limitations are defined as follows: a server location has storage capacity S, total read speed (download capacity) D, total write speed (upload capacity) D' and supports at most V simultaneous users.

Now the service requirements can be defined: a user U_u at access node A_j ($U_u@A_j$) requires a certain amount of storage capacity S_{ju}, with a downstream (read) bandwidth D_{ju} and an upstream (write) bandwidth D'_{ju}. The total required storage capacity, downstream bandwidth, upstream bandwidth from access node A_j is $S_j = \sum_{U_u@A_j} S_{ju}$, $D_j = \sum_{U_u@A_j} D_{ju}$, $D'_j = \sum_{U_u@A_j} D'_{ju}$ respectively. The maximum delay for accessing the storage service is defined as d.

Finally, before describing the BILP problem, some variables need to be defined: s_i is a binary variable indicating whether servers are installed at node N_i or not, while s_{iju} is a binary variable representing whether user U_u at access node A_j is served by a server at node N_i. Defining P_{ij} as the set of all downstream paths from node N_i to access node A_j, then p_{ijk} is a continuous variable representing the flow on the k^{th} path P_{ijk} from N_i to A_j and $p_{ij} = \sum_{P_{ijk} \in P_{ij}} p_{ijk}$ is

the total flow from N_i to A_j. Analogously, defining P'_{ij} as the set of all upstream paths from access node A_j to node N_i, then p'_{ijk} is a continuous variable representing the flow on the k^{th} path P'_{ijk} from A_j to N_i and $p'_{ij} = \sum_{P'_{ijk} \in P'_{ij}} p'_{ijk}$ is the total flow from A_j to N_i. The BILP problem can now be described as follows:

Minimize:

$$z = \sum_{N_i \in N} s_i \tag{1}$$

subject to:

$$p_{ij} = \sum_{U_u @ A_j} s_{iju}.D_{ju}/b, p'_{ij} = \sum_{U_u @ A_j} s_{iju}.D'_{ju}/b, \forall N_i \in N, \forall A_j \in A \tag{2}$$

$$p_{ij} \leq s_i.D_j/b, p'_{ij} \leq s_i.D'_j/b, \forall N_i \in N, \forall A_j \in A \tag{3}$$

$$\sum_{N_i \in N} s_{iju} \geq R, \forall A_j \in A, \forall U_u \in U \tag{4}$$

$$\sum_{N_i \in N} p_{ij} = R.D_j/b, \sum_{N_i \in N} p'_{ij} = R.D'_j/b, \forall A_j \in A \tag{5}$$

$$\sum_{\forall P_{ijk} \ni E_l} p_{ijk} + \sum_{\forall P'_{ijk} \ni E_l} p'_{ijk} \leq b_l, \forall E_l \in E \tag{6}$$

$$\sum_{E_l \in P_{ijk}} c_l \leq d, \sum_{E_l \in P'_{ijk}} c_l \leq d, \forall N_i \in N, \forall A_j \in A, \forall P_{ijk} \in P_{ij}, \forall P'_{ijk} \in P'_{ij} \tag{7}$$

$$\sum_{A_j \in A} p_{ij} \leq D, \sum_{A_j \in A} p'_{ij} \leq D', \forall N_i \in N \tag{8}$$

$$\sum_{A_j \in A} \sum_{U_u @ A_j} s_{iju}.S_{ju}/b \leq S, \forall N_i \in N \tag{9}$$

$$\sum_{A_j \in A} \sum_{U_u @ A_j} s_{iju} \leq V, \forall N_i \in N \tag{10}$$

For now, assume $R = b = 1$, as these parameters are only required for including striping and resilience in the BILP formulation (cf. Section 7). The constraints in (2) guarantee the (down- and upstream) bandwidth requirements between node N_i and access node A_j are met for each user U_u at A_j that is served by a server at N_i. Constraints (3) ensure the total flow from a node N_i to an access node A_j does not exceed the total (down- and upstream) bandwidth demand of A_j. Furthermore, together with the constraints in (2) and the fact that we're dealing with a minimization problem, the constraints in (3) implicitly guarantee that $s_i = 1$ if and only if one of the $s_{iju} = 1$. Constraints in (4) and (5)

ensure that each user U_u is assigned at least one server and that total (down- and upstream) bandwidth demand for each access node A_j is met, respectively. Link bandwidth constraints are described in (6) and delay constraints in (7). Finally, the last three sets of constraints represent the server capabilities: read and write performance of a server in (8), storage capacity in (9) and a maximum number of simultaneous users in (10).

5 Storage Server Placement Algorithm (SSPA)

As BILP solutions to a server placement problem tend to scale poorly, a heuristic algorithm was designed to solve the storage server placement problem, formalized by the BILP model in Section 4. The algorithm consists of two phases. In the first phase, servers are installed one by one at the best candidate locations, until all users are served, while respecting the maximum delay constraint, link bandwidth constraints and server constraints. The quality Q_i of a candidate location N_i is determined using local information, such as the number of access nodes that can be reached from that location, without exceeding the maximum delay d, the average delay to those access nodes and the total available bandwidth on paths with delay $\leq d$ to those access nodes. Once a server location is added, it is used to its maximum capacity (respecting server and network constraints), before a new location is selected. In the second phase, the allocation of servers to users is redistributed in such a way that, where possible, each user is served by the server location closest to his access node. During this redistribution, only locations selected during the first phase are considered. Server locations that no longer serve any users after phase two are removed.

The operation of the first phase is illustrated in Fig. 2. In the figure, Q_i is defined as the number of access nodes in range from N_i. When two candidate nodes N_i have equal quality Q_i, one is selected at random. Once a server location N_i is selected, all access nodes A_j in range from N_i are connected to a helper node T, using helper edges with delay 0 and link bandwidth D_j, the total (remaining) required downstream bandwidth from A_j. That way, a successive shortest path algorithm (from N_i to T) can be used to assign as much users U_u as possible to the newly installed server location N_i. This successive shortest path algorithm continues until N_i has no capacity left, all demand from the access nodes in range is served or no more paths with delay $\leq d$ are available. By using the remaining downstream bandwidth requirements as link capacity for helper edges connecting an access node to T, total capacity assigned to the access node is guaranteed not to exceed total demand, as storage capacity and down and upstream bandwidth are assigned to a single server location on a per user basis. This process is repeated until all users U_u on all access nodes A_j are served.

In the example of Fig. 2, each user requires a server location at maximum two hops from his access node. In the first iteration, N_3 and N_4 both have four access nodes in range. Assume N_4 is selected as the first server location. The access nodes in range from N_4, i.e. A_1, A_2, A_3 and A_4, are connected to a helper

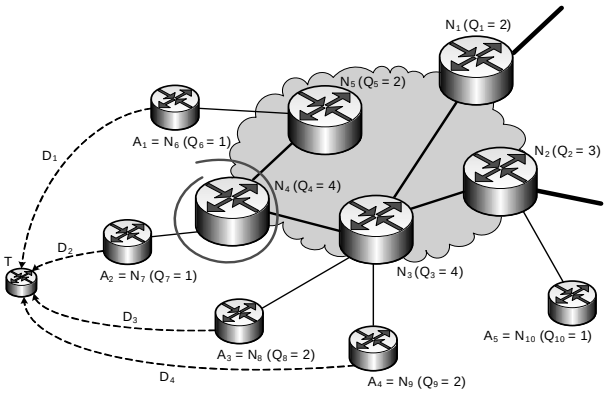


Fig. 2: Operation of the first phase of SSPA. Q_i is defined as the number of access nodes in range from N_i and each user requires a server location at maximum two hops from his access node.

node T after which a successive shortest path algorithm is executed to assign users connected through these access nodes to N_4.

In the second phase, the allocation of servers to users is redistributed in such a way that, where possible, each user is served by the server location closest to his access node. Server locations that become obsolete after this redistribution are removed. The second phase considers server locations added during phase one in reverse order. As phase one only adds a new server location when previously added locations can no longer handle additional user demand, the later added locations are the most unlikely to be expendable. Phase two only adds a server location (out of the server locations selected during phase one) when previously added server locations can't take over all user demand it served after phase one. Phase two only considers server locations on the path from U_u's phase one server location to the access node connecting U_u. That way, phase two is guaranteed to respect link bandwidth constraints.

6 Implementation and evaluation

The SSPA heuristic was implemented using the Telecom Research Software library (TRS) [17], a Java network library, which was extended to support modeling of personal content storage. For evaluating the heuristic, its placement results were compared to those of a direct implementation of the BILP formulation through CPLEX, a branch and bound solver [18]. In order to keep the ILP calculations feasible, unlimited server capacity was assumed and only downstream bandwidth demand was considered, removing constraints (8), (9) and (10) and all p' related expressions from the BILP problem formulation in Section 4. Furthermore, by only enumerating paths P_{ij} with length $\leq d$, constraint (7) can also be removed and a further speed-up can be achieved.

	BILP / SSPA		
max # hops	**1**	**2**	**3**
0 - 60 Mbps	25 / 25	5 / 5	1 / 1
61 - 120 Mbps	25 / 25	5 / 5	2 / 2
121 - 600 Mbps	25 / 25	25 / 25	25 / 25
> 600 Mbps	250 / 250	250 / 250	250 / 250

Fig. 3: Simulation results for the mesh of trees topology, depicted in Fig. 1(a). The table gives number of server locations for a varying demand per access node and increasing maximum delay.

The number of storage server locations, required for providing a fast storage service, was computed for varying total demand per access node (equally divided between 1000 users connected through each access node) and maximum delay. For the evaluation, maximum delay for accessing a server is expressed as the number of hops from the access node and the candidate node quality, Q_i, is defined as the number of access nodes in range from node N_i. In order to allow SSPA to break free from suboptimal solutions, a random node is selected from the n best candidate nodes (with n a configurable upper bound) and the best placement after multiple (i.e. ten) runs is used. For instance, in the ring of rings topology of Fig. 1(b), n should be at least 6, to allow SSPA to choose a non-primary ring node as first server location.

Results for the mesh of trees topology are summarized in the table of Fig. 3. For this rather simple topology, the SSPA heuristic produces optimal results. Results from the simulations on the ring of rings topology are depicted in Fig. 4. Here, the SSPA heuristic adds slightly more server locations than the optimal result. Results are still close to optimal, however. Furthermore, each SSPA iteration takes less than a minute on modern PC hardware[1], as opposed to hours to days—depending on the parameters—for the BILP implementation to produce a single result.

7 Striping & Resilience

High availability is an important feature of a network storage system. In a world where harddisks are failure prone and recordable optical media only have limited archival lifespan [19], high availability and guaranteed data retention are major selling points for a high speed network storage service. In order to protect data from node failures, two alternative approaches are considered: whole-file replication and erasure code (or block) replication [20]. In case of whole-file replication, a user's data is replicated on $R > 1$ server locations. The user's data is available, as long as at least one of the R replicas is online. As a drawback, whole-file replication requires R times the storage capacity of storing each data item once.

[1] The simulations were executed on an AMD64 3000+ system with 512 MB of memory, running Linux and the Sun J2SE 5.0 Runtime Environment.

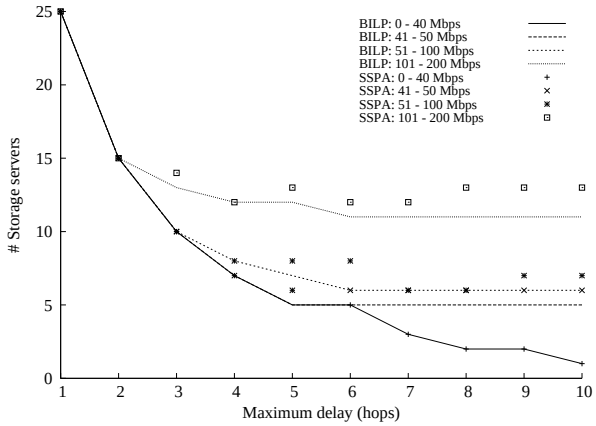


Fig. 4: Simulation results for the ring of rings topology, depicted in Fig. 1(b). Lines represent the optimal BILP results, while dots represent the SSPA results.

Furthermore, a user's bandwidth requirements must be guaranteed to each of the R replica locations in order to guarantee service quality in case of node failure.

In case of block replication, a data item is split into $b > 1$ blocks. Erasure coding is then applied to these b data fragments, producing $R > b$ fragments of the same size as before [21]. The essential property of erasure coding is that any b of the R coded fragments are sufficient to reconstruct the original data item. Each of the R coded fragments are stored at a different server location. A data item is available as long as at least b of the R data fragments are online. Block replication only requires $\frac{R}{b}$ times the storage capacity of storing each data item once. Furthermore, each of the R replica server locations only has to guarantee $\frac{1}{b}$ of a user's bandwidth requirements, at the cost of greater complexity. Note that whole-file replication can be represented as a special case of block replication, where $b = 1$.

Lin, Chiu and Lee [20] showed that the benefit of erasure code (block) replication depends on the node availability, relative to the storage overhead S ($S = R$ for whole-file replication and $S = \frac{R}{b}$ for block replication). Defining μ as the node availability, the file availability is given by:

$$A = \sum_{i=b}^{R} \binom{R}{i} \mu^i (1-\mu)^{R-i} \tag{11}$$

A thorough analysis of the file availability, using different scenarios for both replication schemes, shows that whole-file replication might actually perform better for the same storage overhead S when node availability is low [20].

Striping can be defined as a special case of block replication, where $R = b$. A file is split into b blocks, each block stored at a different server location. Striping provides no resilience—once one of the b server locations goes offline, the file can no longer retrieved—but allows to better distribute the load of the storage

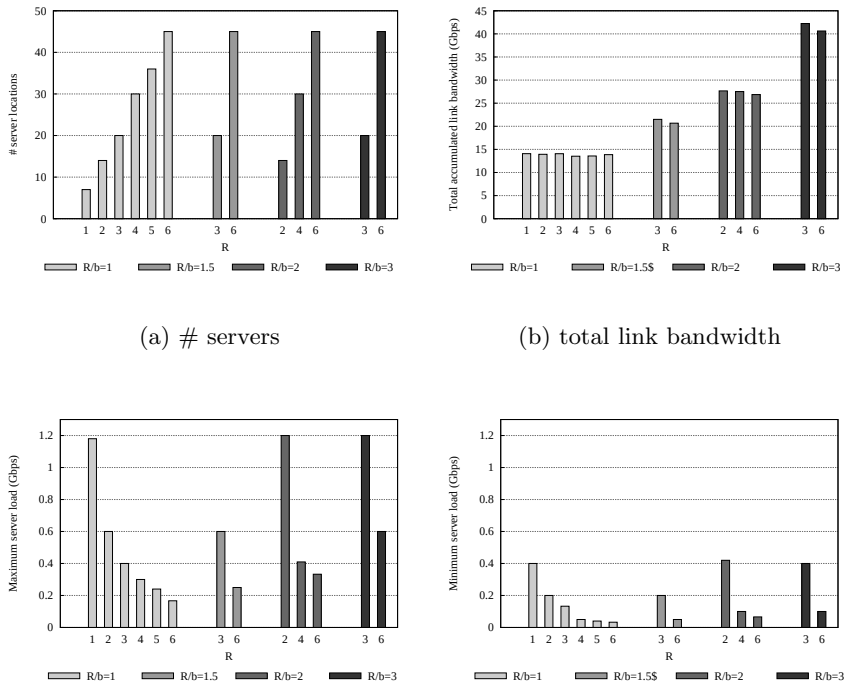
system. With the above definitions of $b \geq 1$ and $R \geq b$, the BILP problem description from Section 4 now includes striping and resilience.

For solving the storage server placement problem, supporting striping and resilience, the SSPA heuristic from Section 5 is extended. Phase one operates roughly the same, but only assigns $\frac{1}{b}$ of a user's storage and bandwidth requirements to a single server location. A user U_u is only removed from the set of users U when he has R server locations assigned. Furthermore, the link bandwidth of the helper edges E_{A_jT} is initialized at $\frac{D_j}{b}$, as a single server location provides at most $\frac{1}{b}$ of the total required bandwidth from access node A_j. In phase two, an additional check should be made whether a closer server location on the path, from the current server location on that path to an access node, does not already serve (some of) the users on that access node, before relocating these users to that closer server location.

For illustrating the effect of replication and striping on the storage server placement problem, the ring of rings topology from Fig. 1(b) was used. In order to keep the simulations interpretable, read operations are assumed to be dominating the demand. First, server locations are assumed to have unlimited capacity, in order to analyze network limitations and SSPA's load distribution. After that, the number of simultaneous users per server location is limited by installing a maximum read capacity. The quality metric Q_i for selecting the best candidate server location N_i was constructed in such a way that the nodes N_i with the highest number of access nodes in range and the lowest average delay to those access nodes are the most likely to be chosen. Again, for avoiding local optima, a location is randomly selected from the n best candidates ($n > 6$, if multiple nodes have equal quality, they are all considered). The figures show the best result after five runs. Again, the hop-count was used as a delay measure. Simulation results include the total number of server locations, the total accumulated link bandwidth (the sum of the bandwidth usage on all links in the network), and bandwidth requirements at the least and the most loaded server location.

Assuming unlimited server capacity, simulations were run for varying service requirements: maximum delay, read bandwidth per access node, number of replica locations (R) and number of blocks (b) per data item. Simulations results show that for a fixed storage overhead $\frac{R}{b}$, the total accumulated link bandwidth (hence the bandwidth cost for deploying the service) is constant for increasing R, when the network has sufficient bandwidth available (small fluctuations are explained by the heuristic nature of SSPA). Furthermore, as one would expect, for fixed $\frac{R}{b}$ the total number of server locations increases and the maximum server load decreases for increasing R. This is illustrated in Fig. 5 for a maximum delay of 4 hops and a read bandwidth of 20 Mbps per access node.

As SSPA seeks to minimize the number of server locations, it will tend to maximize the usage of each server location. As a side-effect, load is not evenly distributed among the available server locations. This is clearly visible in Fig. 5(c) and 5(d): e.g. in the case where each data item is only stored at one server location ($\frac{R}{b} = 1$ and $R = 1$), the maximum loaded server location provides almost 25% of the 5 Gbps total required bandwidth, while the least loaded server lo-



(a) # servers

(b) total link bandwidth

(c) maximum server load

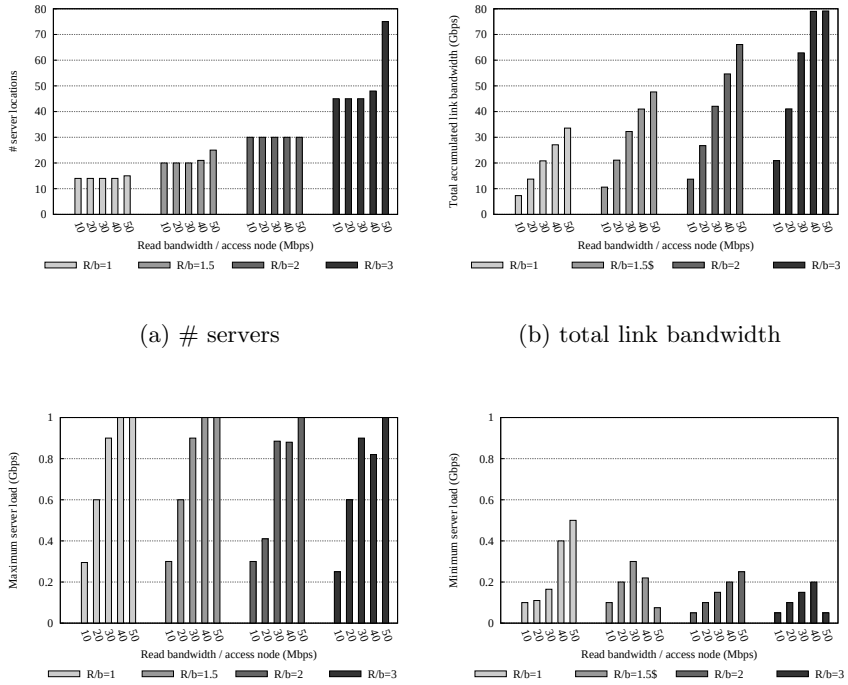
(d) minimum server load

Fig. 5: Number of server locations, total link bandwidth, maximum and minimum server location read bandwidth for varying $\frac{R}{b}$ and increasing R. Service requirements are a maximum delay of 4 hops and a read bandwidth of 20 Mbps per access node. Server capacity is assumed unlimited.

cation only provides 400 Mbps. Work is in progress to design a third phase for SSPA, providing better balancing of the load over the available server locations, using a modified minimum cost flow algorithm—from the server locations remaining after phase two to the access nodes—with dynamic costs for avoiding saturated links and heavily loaded locations, i.e. costs $\sim \frac{1}{1-load(\%)}$ [3].

For the remainder of the simulations, server location read capacity is restricted to 1 Gbps[2]. Furthermore, the maximum delay is set to 4 hops and the number of fragments per data item is set to $b = 2$. Fig. 6 summarizes simulation results for varying storage overhead $\frac{R}{b}$ and increasing read bandwidth demand per access node. From these simulations it is clear that SSPA effectively tries to use servers at each location to their maximum capabilities before adding a new server location. Total accumulated link bandwidth increases linearly with

[2] Note that in Fig. 5, the maximum server location read bandwidth is 1.2 Gbps.



(a) # servers

(b) total link bandwidth

(c) maximum server load

(d) minimum server load

Fig. 6: Number of server locations, total link bandwidth, maximum and minimum server location bandwidth for varying $\frac{R}{b}$ and increasing read bandwidth requirements per access node. A maximum delay of 4 hops is required. The number of data fragments per file $b = 2$ and server locations have a maximum read capacity of 1 Gbps.

increasing demand per access node and increasing storage overhead. Furthermore, doubling the storage overhead roughly doubles the number of server locations. Both observations only hold as long as the access and aggregation network has sufficient bandwidth available. Note that for a read bandwidth demand of 50 Mbps per access node and a storage overhead $\frac{R}{b} = 3$, which implies 6 server locations per data item for $b = 2$, SSPA is forced to add multiple storage server locations at the access nodes, as the secondary ring bandwidth becomes the limiting factor.

8 Conclusion

The deployment of transparent personalized services requires the complete decoupling of devices for accessing the service and the supporting personal data

storage. This paper studied the deployment of a personal content storage service, supporting created content, in the access and aggregation network, minimizing the number of server locations, while guaranteeing fast access from any access node. Both network restrictions—link bandwidth and maximum delay—and server restrictions—read and write performance, storage capacity and a maximum number of simultaneous users—are taken into account.

First it was assumed that servers do not fail and storing each data item once is sufficient. This problem was first formulated as a Binary Integer Linear Programming (BILP) problem. Next, a heuristic Storage Server Placement Algorithm (SSPA) was presented, installing servers one by one at the best candidate locations, until all users are served. Two typical access and aggregation network topologies were used for evaluating the algorithm: a mesh of trees, used in DSL deployment, and a ring of rings, used for cable Internet access deployment. Simulations, comparing SSPA's placement results to the optimal placement, achieved using a CPLEX branch and bound implementation of the BILP formulation, showed the heuristic algorithm produces close to optimal results.

As high availability is an important feature of online personal content storage systems, while real-world servers do fail, both the BILP formulation and the SSPA heuristic were extended to support resilience and striping. The extended storage server placement problem was used to simulate some resilience and striping scenarios in the aforementioned ring of rings topology. As one would expect, simulation results showed that doubling the storage overhead roughly doubles the number of required server locations and the total accumulated link bandwidth, as long as the access and aggregation network has sufficient bandwidth available. For all use cases presented in this paper, simulation took less than a minute per parameter combination on modern PC hardware, showing that SSPA is very efficient for optimizing server placement in personal content storage deployments. The presented SSPA simulator was already used for evaluating a business case for deploying a digital media library service in the access and aggregation network of the Belgian DSL operator Belgacom [22].

Acknowledgment

This work is partially funded by the IBBT GBO project PeCMan.

References

1. Baya, V., Gauntt, J.: The Rise of Lifestyle Media: Achieving Success in the Digital Convergence Era. Technical report, PriceWaterHouseCoopers (2006)
2. Stevens, T., Vlaeminck, K., Van de Meerssche, W., De Turck, F., Dhoedt, B., Demeester, P.: Deployment of Service-Aware Access Networks through IPv6. In: 8th Int. Conf. on Telecommunications. (2005)
3. Wauters, T., Coppens, J., Turck, F.D., Dhoedt, B., Demeester, P.: Replica Placement in Ring Based Content Delivery Networks. Journal of Computer Communications **29**(16) (2006) 3313–3326

4. Vlaeminck, K., De Turck, F., Dhoedt, B., Demeester, P.: Deploying Digital Media Libraries in Multi-Service Access Networks. In: 8th IEEE Int. Symposium on Multimedia. (2006) 105–112
5. Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.: Progress in Linear Programming-Based Algorithms for Integer Programming: An Exposition. INFORMS Journal on Computing **12**(1) (2000) 2–23
6. Callaghan, B., Pawlowski, B., Staubach, P.: RFC1813: NFS version 3 protocol specification (June 1995)
7. Howard, J.H., Kazar, M.L., Menees, S.G., Nichols, D.A., Satyanarayanan, M., Sidebotham, R.N., West, M.J.: Scale and Performance in a Distributed File System. ACM Trans. Comput. Syst. **6**(1) (1988) 51–81
8. Mummert, L.B., Ebling, M.R., Satyanarayanan, M.: Exploiting Weak Connectivity for Mobile File Access. In: 15th ACM Symposium on Operating Systems Principles, New York, NY, USA, ACM Press (1995) 143–155
9. Cluster File System Inc.: Lustre: A Scalable, High-Performance File System. Technical report (November 2002)
10. Jones, T., Koniges, A., Yates, R.K.: Performance of the IBM General Parallel File System. In: IEEE Int. Parallel & Distributed Processing Symposium. (May 2000) 673–683
11. Ghemawat, S., Gobioff, H., Leung, S.T.: The Google file system. In: 19th ACM Symposium on Operating Systems Principles, New York, NY, USA, ACM Press (2003) 29–43
12. Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Wells, C., Zhao, B.: OceanStore: an Architecture for Global-Scale Persistent Storage. SIGARCH Comput. Archit. News **28**(5) (2000) 190–201
13. Bolosky, W.J., Douceur, J.R., Ely, D., Theimer, M.: Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs. SIGMETRICS Perform. Eval. Rev. **28**(1) (June 2000) 34–43
14. Saito, Y., Karamanolis, C., Karlsson, M., Mahalingam, M.: Taming Aggressive Replication in the Pangaea Wide-Area File System. SIGOPS Oper. Syst. Rev. **36**(SI) (2002) 15–30
15. Noble, B., Fleis, B., Kim, M.: A Case for Fluid Replication. In: Network Storage Symposium, Seattle, WA, USA (October 1999)
16. Qiu, L., Padmanabhan, V.N., Voelker, G.M.: On the Placement of Web Server Replicas. In: INFOCOM. (2001) 1587–1596
17. Casier, K., Verbrugge, S., Coller, D., Pickavet, M., Demeester, P.: Using Aspect-oriented Programming for Event-handling in a Telecom Research Softwave Library. In: 8th Int. Conf. on Software Reuse. (2004)
18. ILOG: CPLEX. http://www.ilog.com/products/cplex/
19. Vitale, T.: Digital Imaging in Conservation: File Storage. AIC News **31**(1) (January 2006)
20. Lin, W.K., Chiu, D.M., Lee, Y.B.: Erasue Code Replication Revisisted. In: 4th Int. Conf. on Peer-to-Peer Computing. (August 2004) 90–97
21. Pless, V.: Introduction to the Theory of Error-Correcting Codes. 3 edn. Wiley-Interscience (1998)
22. Van Ooteghem, J., Vlaeminck, K., Colle, D., Pickavet, M., De Turck, F., Dhoedt, B., Demeester, P.: Business Case for Deploying Digital Media Libraries in Multi-Service Access Networks. In: 2007 Int. Conf. on Business and Information, Tokyo, Japan (July 2007)