Closing the Generation Gap: Satisfying Servers, Providers, and Visitors of Hyper-Content *

Jay A. Patel and Indranil Gupta
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{jaypatel,indy}@cs.uiuc.edu

Abstract

We consider the problem of building a robust and efficient infrastructure for hyper-content. Existing server-only architectures are susceptible to flash crowds, while other peer-to-peer (p2p) solutions may not yet be economically viable because of their voluntary model. We start with the assumption that any infrastructure has to satisfy three types of key players - the content provider, the server, and the visitor. Our solution allows the provider to specify a pricing function, optimizes resource usage and avoids flash crowds at the server, and keeps response times low for the visitor. The solution adapts at run-time in order to satisfy these requirements. With low visitor concurrency, our solution reduces to a server-only solution. With rising concurrency, more and more hyper-content gets distributed and shared out among visitors and providers. This leads us to believe that the "right" infrastructure for hyper-content may lie somewhere in between the current two extremes of server-only and p2p.

1 Introduction

Increasingly, web content is being published by decentralized "content providers", e.g., blogs that are published by individual web users. The current web infrastructure, relying on servers to host this content, is often unable to withstand the performance burdens imposed by popular content, e.g., flash crowds can cripple a server [7]. On the other hand, peer to peer (p2p) solutions [9, 16], solve the performance problem to an extent, but since they are based on a "volunteer" model, they may not be economically viable today.

Our work in this paper leads us to believe that the most pragmatic solution may lie somewhere in between these two extremes of server-only and p2p infrastructures, and in fact involve some elements of both. Consider a broad characterization of web content called "hyper-content", defined as small to medium-sized individual documents that are updated frequently, and are often generated dynamically from other pre-specified static documents. Hyper-content documents are interactive, possibly embedded with dependent documents or with references (links).

Any infrastructure for hyper-content is required to satisfy three types of players: the content *provider*, the *visitor* of the content, and the *server* which hosts the content. A pragmatic solution reduces the economic cost for the provider (in dollars), produces quick response times for the visitor, and utilizes the limited server resources in an efficient manner that avoids the effect of flash crowds and high concurrency.

We are building a hyper-content infrastructure that follows the line of thinking laid out above. In this paper, we first formulate a *pricing scheme* that the provider can agree with, and then include mechanisms (1) to provide reasonable response times to visitors, as well as (2) to avoid flash crowds at the server. We are able to build a solution by leveraging off our previously designed *Overhaul* [17] protocol that is aimed at circumventing flash crowds.

In order to ground our arguments, we first use web site logs (gathered from the UIUC Dept. of Comp. Sci. web site) to study and verify certain relevant and often-assumed characteristics of today's Web infrastructure (Section 2). This leads us to define *concurrency* level as a primary parameter to consider while detailing our price function (Section 3). Finally, we outline a simple solution realizing this approach (Sections 4, 5).

Other Related Work Many solutions proposed for reducing costs of content distribution have focused on power consumption [4, 6]. Efforts on reducing bandwidth costs have been limited to large

*This research was partly supported by National Science Foundation Grant ITR-0427089

multimedia-type files [1], which are order(s) of magnitude larger than hyper-content documents. Solutions [14, 12] aimed at reducing response time to visitor have been shown to have limited scalability [11]. Work on improving server performance [15, 10] has come at expense of provider costs.

2 Basics

This section first formally defines hyper-content, both for this paper and for future work. Then, through analysis of data gathered from the UIUC website logs, we discuss the relation between concurrency on one hand and server bandwidth utilization, server throughput, and client response times on the other hand.

Hyper-Content Formally Defined A hyper-content collection is composed of a D set of *documents*. A *session* is a stream of individual requests for one or more documents, which expires after t_{out} time of inactivity (i.e., no further requests). A visitor establishes a new session by requesting more documents after a session expires. Each session is composed of the visitor (v_j), the starting time (t_{begin}), the completion time (t_{end}), and the set of transferred documents ($D_i \subset D$).

Increasing Bandwidth Utilization *Concurrency* can be defined as the number of discrete document transfers (either started, ongoing, or completed) within a time period t_{period}. Server *load* is a special case of concurrency calculated using a very small value ϵ for t_{period}. For purposes of brevity, we use the notation C_i, where $i \geq 0$, to denote the concurrency level.

It may seem intuitive that as concurrency increases, the amount of utilized bandwidth also increases. As it forms the basis of our principle argument, statistical analysis of trace data help further justify the corollary. Figure 1 demonstrates that there is a tight linear correlation between bandwidth utilization and the concurrency. This data is based on traces from the UIUC Dept. of Comp. Sci. web site for the year 2004. A function $\mathcal{F}$ can be defined (independently for each content collection) such that: $\mathcal{F}(C_p) = B_q$. For example, $\mathcal{F}(C_p) = 4.53KB * C_p$ (with asymptotic std. error of 3.334%) for the aforementioned web site. This signifies that the amortized bandwidth utilization is approximately 5KB per document (per visitor).

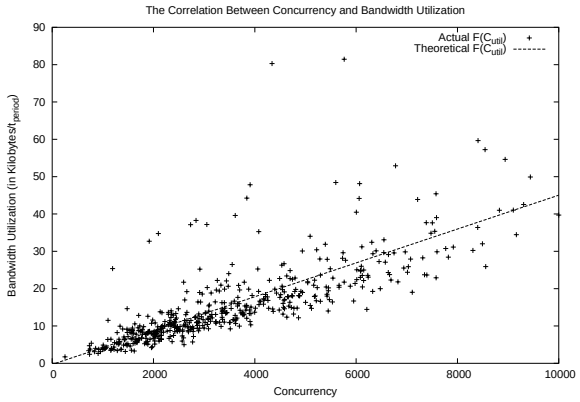

Figure 1: **A randomized sample from access logs of the UIUC Dept. of Comp. Sci. web site for year 2004. The correlation between concurrency (with $t_{period} = 3600$s) and utilized bandwidth can be duely noticed.**

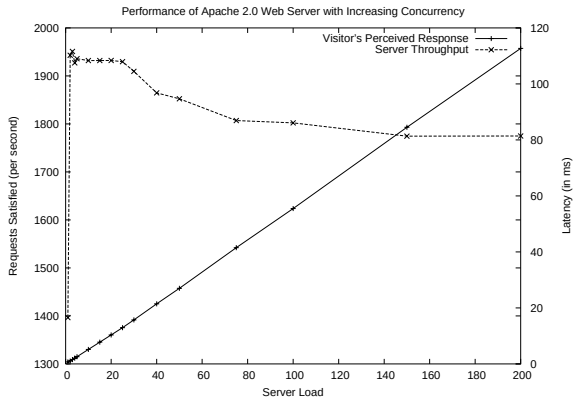

Figure 2: **The performance of a typical server degrades with increasing concurrency. The server throughput decreases and the server response (to visitors) increases.**

Performance Degradation Servers exhibit a degradation in performance after reaching peak resource utilization, a common characteristic of multiprogramming. In Figure 2, experiments performed on the Apache web server transferring a small 2KB file demonstrate that as the concurrency level increases, the server performance degrades because of suboptimal resource utilization to handle additional connections. As a result, server throughput decreases and the content delivery time increases.

3 Basis for a New Framework

The major problems associated with traditional server-client paradigm include increased bandwidth utilization (from the provider's perspective), performance degradation (from server's perspective)

and increased transfer latencies (from the visitor's perspective) as content popularity increases (Section 2). In this section, we provide goals for a new framework that provides a balance between the needs of the all involved parties.

Limiting Provider's Costs In our model, the following axioms are assumed to apply to the economics of bandwidth purchase:

1. A provider purchases B_{norm} bandwidth (possibly "guaranteed" in the SLA [18]) from a *service provider* who has a maximal capacity of B_{max} bandwidth, where $B_{max} \geq B_{norm}$.
2. A non-decreasing pricing function $cost(B_{util})$ is used to calculate costs of utilized bandwidth. The *base cost* of the bandwidth is $cost(B_{norm})$, even if $B_{util} \leq B_{norm}$.
3. A service provider can not provide bandwidth greater than $B_{max}, therefore cost(B_{max} + \epsilon) = \infty$.

Given the assumptions above, a service provider sells excess bandwidth ($B_{excess} = B_{max} - \sum B_{norm}$, i.e., unsold bandwidth) at a nominal cost to customers requiring bandwidth greater than B_{norm}. As the supply of B_{excess} is finite, the cost of additional bandwidth increases with demand, especially if multiple customers are involved. An enterprising service provider can provide additional bandwidth by "borrowing" underutilized portions of B_{norm} from other customers, if the penalties for violating guaranteed service (as agreed upon in a SLA) are offset by the benefits. However, such additional bandwidth will incur a substantial cost premium (which increases costs rapidly) if multiple SLAs are violated.

Based on the above mentioned cost model, the marginal cost of supporting additional concurrent users is: given $p > q \geq 0$,

$$\mathcal{MC}_{p,q} = \frac{cost(\mathcal{F}(C_p)) - cost(\mathcal{F}(C_q))}{C_p - C_q} \qquad (1)$$

The traditional server-client paradigm leads to an increasing marginal cost (Figure 3), i.e., the $cost(B_{util})$ plot is concave. In contrast, our proposed distribution framework provides guarantees on the upper-bound, $mc_{threshold}$, of the marginal cost. Mathematically, the framework satisfies:

$$\frac{d\left(cost\left(\mathcal{F}(C)\right)\right)}{dC} \leq mc_{threshold} \qquad (2)$$

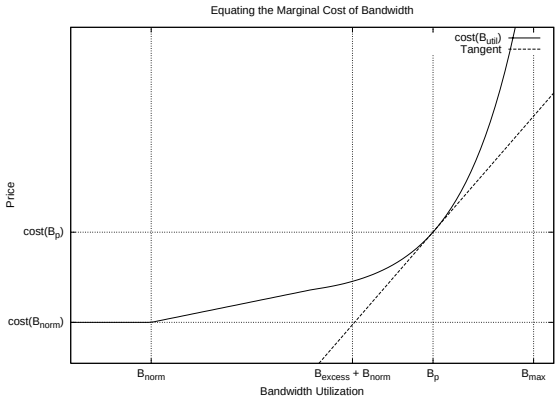

Figure 3: **The pricing of bandwidth is plotted by $cost(B_{util})$. At B_p the marginal cost of bandwidth is equal to the slope of the tangent.**

The maximum concurrency level for which Equation 2 holds true is denoted as $C_{s\text{-}threshold}$.

Visitor's Perspective Let us denote the maximum amount of time a consumer is willing to wait (with a high probability) for the content as t_{wait}. Based on aforementioned service degradation (see Section 2, a server can not guarantee the content delivery time of $\leq t_{wait}$ above a certain concurrency threshold of $C_{c\text{-}threshold}$.

Our proposed distribution framework aims to promote the harmony between the requirements of both the providers and visitors by using the server-client model only for concurrency levels below:

$$C_{max} = min(C_{s\text{-}threshold}, C_{c\text{-}threshold}) \qquad (3)$$

At higher concurrencies, a shift to a more p2p-style content sharing protocol is required. We elaborate in Section 4.

4 A Distributed Implementation

An increasing number of Internet users are subscribing to broadband connections. Due to the increased up-line capacities, p2p resource sharing with Napster, Gnutella, Kazaa, etc. has gained widespread acceptance. In a similar manner, our proposed framework alleviates the responsibilities of distributing hyper-content solely from the provider and shares it amongst the visitors.

Overhaul Hyper-content has a natural property of being delivered in "chunks": initially the anchor document is transferred, followed by the dependencies. We have previously designed Overhaul

[17], an HTTP extension, in which chunking is applied on the individual document itself. A server (when under flash crowd load) divides a document in to c chunks and maintains a list of the last m visitors (per document). The visitors are sent a single chunk (distributed sequentially) of the requested document along with an addendum header, $h_{overhaul}$, containing verification signatures for the c chunks and the initial membership list.

The visitors form an a p2p overlay on the fly, based on the initial membership list, where they exchange chunks and discover new peers (as visitors join in). No distributed hash table is used. Moreover, peers discover and fetch dependent documents autonomously, without server intervention. Experimental results show that the Overhaul extension thoroughly outperforms traditional server-client paradigm under high concurrency.

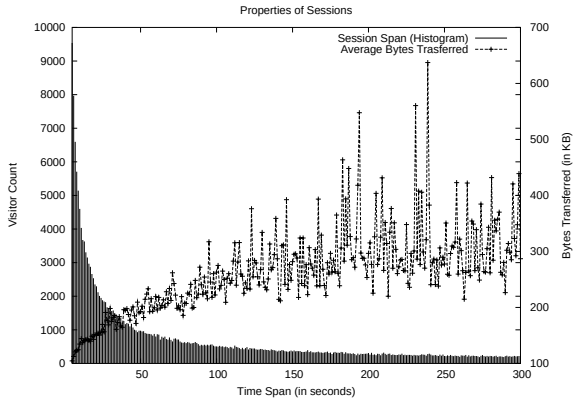

Figure 4: **Traces from UIUC Dept. of Comp. Sci web site show that a substantial number of visitors have long session time span. Additionally, longer time spans imply more document transfers.**

Exploiting Sessions The session time span is defined as $t_{span} = t_{end} - t_{begin}$. Additionally, during a session, a visitor may remain inactive (for up to t_{out} time) before either, requesting an additional document, or, terminating the session. To improve performance (by reducing churn [5]), each visitor is forced (by protocol and server policy) to remain an active member of the p2p overlay for at least $average(t_{span}) + t_{out}$. A non-compliant client is barred from accessing further content. For example, for the UIUC Dept. of Comp. Sci. web site, the $average(t_{span})$ is 79.44s, and the average data requested by a visitor is 146KB. Hence, each visitor need only have an average up-link capacity of 2KB/s (conservatively ignoring server contribution, and assuming $t_{out} = 0$sec).

Optimal Bandwidth Utilization A server utilizing a chunking p2p protocol will drastically reduce bandwidth utilization. A document utilizes only $1/c^{th}$ the bandwidth (ignoring the size of $h_{overhaul}$) of a server-client protocol. However, maximum savings are achieved when the server sends only a single chunk of a document to each visitor (per session). The optimal bandwidth utilization is:

$$B_{util}^{p2p} = \frac{B_{util}}{c * average(|D_i|)} \quad (4)$$

A Hybrid Approach Due to p2p economies of scale, a visitor might not be able to fetch required documents within t_{wait} (from the overlay) at low concurrency levels. However, at a certain concurrency level, C_{opt}, it becomes viable for a visitor to fetch the required content within t_{wait}. Therefore, we propose a hybrid approach that utilizes the traditional server-client paradigm until concurrency level approaches C_{opt}. After which, the distribution switches to an Overhaul-based p2p protocol.

Cost Guarantees Using the hybrid approach, the bandwidth utilization for concurrency levels up to $C_{opt} * c * average(|D_i|)$ (Equation 4) will remain below $\mathcal{F}(B_{opt})$. However, during a flash crowd, the utilized bandwidth may exceed the aforementioned limit. If this occurs, the chunk count is incremented to induce a secondary drop in bandwidth utilization. A multiplicative increase in chunk count is performed for each such saturation point: $c_{i+1} = base * c_i$ (given that $c_0 = c$, $base > 0$). This limits the amount of bandwidth a content provider must purchase.

Meeting Our Goals Depending on the characteristics of the content collection, there exist two possibilities regarding the value of C_{opt}:

1. $C_{opt} \leq C_{max}$: The proposed distribution framework is viable for the content collection.
2. $C_{opt} > C_{max}$: At concurrent level C_{opt}, either the marginal cost exceeds $mc_{threshold}$ or the document fetch time exceeds t_{wait}. Either the provider or the visitor may relax requirements to make the framework viable. Alternatively, further optimizations can reduce the value of C_{opt}. We describe a few in Section 5.

5 Further Considerations

Enforcing Fair Exchange Previous studies [8] on p2p resource sharing systems show that a majority of peers make locally optimal decisions. However, due to the presence of a centralized authority (the server) in our framework, we instill greater incentives for client participation, for example, by barring visitors from accessing further content. BitTorrent already utilizes reputation schemes [2] to instill a fair exchange policy.

Dynamic Content The proposed distribution framework can handle limited types of dynamically generated content. The server can use the traditional server-client paradigm for dynamically personalized content and utilize the distributed protocol for static documents.

Publish-Subscribe Recently, numerous schemes (RSS, ATOM) have been introduced to periodically check for content updates. As a result, many providers have introduced a policy of minimum time between access because of additional bandwidth utilized by such schemes. With the proposed distribution framework: (1) the server can *push* updates to random visitors during times of low concurrency, i.e., $\leq C_{opt}$, and (2) regular visitors can form a large, semi-permanent peer group (with a low churn rate) that utilizes either a gossip-style protocol [3] or a structured voting system [13] to share updates.

Assembling Automatic Associations The p2p overlay establishes long-term trust relationships amongst visitors. For example, regular visitors to a web site are likely to have common "surfing" interests. To the best of our knowledge, such relationship amongst consumers of hyper-content has not been previously exploited or researched.

During periods of low concurrency, the bandwidth will be underutilized. At such times, a server may optionally act as a *fabricated peer* for other providers' p2p distribution overlays. This creates mutually beneficial communities of providers which aid to universally lower C_{opt}.

6 Conclusion

In this paper, we verify relevant and often-assumed characteristics of today's web infrastructure (Section 2). Furthermore, we discover *concurrency* level as a primary parameter to consider while detailing the pricing costs of a provider (Section 3). Finally, we present a distribution framework that combines the traditional server-client paradigm and a new p2p protocol based on document chunking (Section 4) to satisfy the needs all three players: the providers, the visitor and the server.

References

[1] L. Cherkasova and J. Lee. FastReplica: Efficient large file distribution within content delivery networks. In *USITS*, 2003.

[2] B. Cohen. Incentives build robustness in BitTorrent. In *Workshop on Econ. of P2P Systems*, 2003.

[3] A. Demers et al. Epidemic algorithms for replicated database maintenance. In *Proc. PODC*, pages 1–12, 1987.

[4] A. Weissel et al. Cooperative I/O: a novel I/O semantics for energy-aware applications. In *Proc. OSDI*, pages 117–129, 2002.

[5] D. Liben-Nowell et al. Analysis of the evolution of peer-to-peer systems. In *Proc. PODC*, pages 233–242, 2002.

[6] E. Carrera et al. Conserving disk energy in network servers. In *Proc. ICS*, pages 86–97, 2003.

[7] J. Jung et al. Flash crowds and denial of service attacks. In *Proc. WWW*, pages 293–304, 2002.

[8] K. P. Gummadi et al. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proc. SOSP*, pages 214–329, 2003.

[9] M. Freedman et al. Democratizing content publication with Coral. In *Proc. NSDI*, pages 239–252, 2004.

[10] M. Welsh et al. SEDA: An architecture for well-conditioned, scalable internet services. In *Proc. SOSP*, pages 230–243, 2001.

[11] M. Wolman et al. On the scale and performance of cooperative web proxy caching. In *Proc. SOSP*, pages 16–31, 1999.

[12] P. Linga et al. A churn-resistant peer-to-peer web caching system. In *Workshop on Survivable and Self-Regenerative Systems*, 2003.

[13] P. Maniatis et al. Preserving peer replicas by rate-limited sampled voting. In *Proc. SOSP*, pages 44–59, 2003.

[14] S. Iyer et al. Squirrel: A decentralized peer-to-peer web cache. In *Proc. PODC*, pages 213–222, 2002.

[15] T. Brecht et al. accept()able strategies for improving web server performance. In *Proc. USENIX*, pages 227–240, 2004.

[16] T. Stading et al. Peer-to-peer caching schemes to address flash crowds. In *Proc. IPTPS*, pages 203–213, 2002.

[17] J. Patel and I. Gupta. Overhaul: Extending HTTP to combat flash crowds. In *Proc. WCW*, pages 34–43, 2004.

[18] E. Wustenhoff. Service level agreement in the data center. *Sun BluePrints OnLine*, 2002.