

APRICOD: A Distributed Caching Middleware for Fast Content Discovery of Non-continuous Media Access *

Zhen Wei Zhao
NUS Graduate School for Integrative Sciences
and Engineering
National University of Singapore, Singapore
zhaozhenwei@nus.edu.sg

Wei Tsang Ooi
School of Computing
National University of Singapore, Singapore
ooiwt@comp.nus.edu.sg

ABSTRACT

We propose an access pattern-driven distributed caching middleware named APRICOD, which can be built on top of any existing content discovery system. APRICOD caters for fast and scalable content discovery in peer-to-peer media streaming systems, especially when user interactions that leads to non-continuous media access (such as random seek in video and teleportation in virtual environment) are present. APRICOD caches query results based on correlations among media objects accessed by users, reducing the content discovery time. Our evaluation using a VoD access trace shows that close to 70% of non-continuous access queries can be resolved with a single hop using APRICOD.

Categories and Subject Descriptors: H.2.4 [Systems]: Query processing; H.3.4 [Systems and Software]: Distributed systems

General Terms: Design, Performance

Keywords: Content discovery system, user interaction, caching, non-continuous media access, peer-to-peer

1. INTRODUCTION

Content discovery is a process by which a peer identifies where to retrieve a required object from. The requirement for content discovery time varies across different applications. File sharing applications can tolerate long content discovery time, while interactive media streaming applications such as VoD and networked virtual environment, require shorter lookup time to ensure a smooth user experience. For instance, De Silva *et al.* [4] show that over 95% of users can tolerate a delay up to only 1 second in progressive mesh streaming.

Approaches to content discovery can be classified into either *centralized*, *DHT-based*, *gossip-based*, or *cell-based* approaches. The centralized approach relies on a central server to index the content and to respond to content discovery queries. Media streaming systems such as Kangaroo [8] adopt this approach. With the centralized approach, queries take only one hop to resolve, but it is not scalable and has a single-point of failure.

*Area chair: Wu-chi Feng

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MM'11, November 28–December 1, 2011, Scottsdale, Arizona, USA.
Copyright 2011 ACM 978-1-4503-0616-4/11/11 ...\$10.00.

DHT is a scalable and distributed content discovery approach. It employs a structured overlay to route queries to their destination through intermediate nodes. Queries can be resolved within maximal $\log(N)$ hops, where N is the number of nodes in the DHT overlay. DHT is highly scalable, but is not suitable for content discovery in interactive media streaming applications due to its potentially long lookup time.

The gossip-based approach is widely used in media streaming systems. With this approach, peers periodically exchange data availability information with their neighbors. To support non-continuous access, gossip-based approach often groups peers into clusters according to their playback point. A peer maintains neighborhood relationship with not only peers in its own cluster, but also some peers in other clusters. If the peer jumps to another cluster due to non-continuous access, it reestablishes its new neighborhood through either its current neighbors or through a neighborhood discovery mechanism (such as through a centralized server or DHT). Different variations of this approach are adopted by systems such as InstantLeap [7] and RINDY [1]. Gossip-based approach works effectively when the number of clusters is small. When the number of clusters increases, however, the content discovery time and neighborhood maintenance overhead increase.

In the cell-based approach, resources are managed distributedly by selected nodes, usually called super peers or cell managers. Each cell manager behaves like a central server for the resource it manages. Hence, the content discovery problem is converted to the problem of finding the cell manager that manages the required resources. Unlike DHT, which routes queries based on the hashed key value, the cell-based approach directs peers to the right cell manager based on temporal or spatial adjacency relations among the cells. For instance, Cheng *et al.* [2] group vertices in a progressive mesh into cells and use this approach to enable content discovery in peer-to-peer progressive mesh streaming. Fast discovery of the right cell manager during non-continuous access, however, remains a problem.

Given the drawbacks of existing content discovery approaches, a fast and scalable content discovery system that supports non-continuous media access is needed. In this paper, we propose such a system called APRICOD. APRICOD is a distributed caching middleware that exploits the access correlations among media objects so as to shorten the content discovery time, especially for non-continuous media access.

We say object Obj_2 is correlated with Obj_1 if peers are likely to access Obj_2 after accessing Obj_1 . In media systems, the two basic sources of data correlation are spatial and temporal relations. For instance, in a virtual environment, two nearby objects are correlated spatially. Two consecutive video segments are correlated

temporally. Such obvious correlations are results of continuous access to media.

Other forms of correlations exist due to non-continuous access. Such correlations can be caused by reasons such as features in user interfaces (bookmarked playback point in a video or teleporters and landmarks in a virtual environment). These correlations can be exploited for provision of fast content discovery during non-continuous access. Existing systems, however, do not exploit such correlations. In particular, DHT-based approaches are unable to exploit these correlations since $\langle key, value \rangle$ pairs are looked up independently of each other. Existing gossip-based and cell-based systems only exploit those obvious correlations, which support continuous media accesses effectively, but incur longer latency for non-continuous access.

APRICOD is the first system that exploit the non-obvious correlations for content discovery. In APRICOD, the convolving correlations (including those non-obvious correlations) are mined in a black-box way, making our system application-independent. More specifically, we use the user interaction history to infer the correlations. The mined correlations are further used to facilitate content discovery. In contrast to the existing data-driven content discovery systems such as DHT and many others, APRICOD is user access pattern driven. To the best of our knowledge, we are the first to explore this new paradigm for content discovery.

We now describe the design philosophies of APRICOD, which lead to the improvements of APRICOD over the existing content discovery systems. Firstly, APRICOD is a caching middleware on top of an underlying content discovery system. It tries to resolve as many queries as possible with merely a single hop. Secondly, APRICOD resolves popular queries faster than unpopular ones. Thirdly, instead of viewing queries issued from different peers as independent from each other, we allow peers (even those offline ones) to collaborate by sharing their query information. Fourthly, APRICOD does not use a fixed overlay structure, rather it adapts to user access pattern by automatically changing its overlay structure over time. Finally, the APRICOD overlay is conceptually separate from either the data dissemination or the content discovery overlay, which makes it general and can be used for various applications and attached to different existing content discovery systems.

2. RELATED WORK

We now briefly discuss related work that considers non-continuous access in peer-to-peer media streaming.

Qiu *et al.* [7] design a content discovery scheme called InstantLeap for P2P VoD streaming. It divides peers into groups according to their playback point and each peer maintains connections to a portion of those groups. Content discovery is achieved through neighbor list exchange. Even though theoretically a constant number of hop is needed at high probability, it is at the cost of scalability because the number of connections a peer has to keep grows linearly with respect to the number of groups. Yang *et al.* [8] adopt a centralized approach to perform content discovery. As a result, it requires peers to continuously update the server about their data availability, which places significant amount of workload on the server when the peer population is large.

MOPAR [9] supports non-continuous access such as teleportation by using DHT to provide global connectivity. When peers teleport to another location, MOPAR discovers peers around the destination by querying the DHT. As a result, it is hard to ensure fast response to user interactions due to the potentially long lookup time of DHT. Cheng *et al.* [2] propose a content discovery scheme specifically for progressive mesh streaming by exploiting the parent-child relation among vertices. However, similar tech-

niques cannot be applied to other systems where such elegant hierarchical relation does not exist.

Several recent research work investigated into exploiting social relationship among users for content discovery. These systems assume that two users who are in the same social group are likely to be interested in the same set of files. Based on this assumption, Pouwelse *et al.* [6] develop Tribler, a social-based peer-to-peer file sharing system. Cheng *et al.* [3] develop NetTube, which exploits social relations to assist in streaming of YouTube videos. It is not clear, however, how social relationship relates to non-continuous access patterns in media.

3. SYSTEM MODEL

As a caching middleware, APRICOD should be used together with an underlying content discovery system that provides query resolution guarantees so as to ensure global connectivity. Fig. 1 shows two possible configurations, where APRICOD provides support for a centralized content discovery system and a DHT-based system respectively.

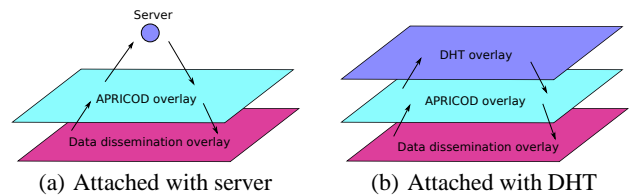


Figure 1: APRICOD used together with underlying content discovery systems. (Arrows indicate query resolution path. Queries are passed to the upper layer if they cannot be resolved by APRICOD.)

Fig. 1 shows that the APRICOD overlay is designed separately from the data dissemination and the content discovery overlay. All layers in Fig. 1 are conceptually independent of each other, even though in real implementation they may be combined together. The data dissemination layer issues query to the APRICOD layer. If APRICOD cannot resolve the query, it relays the query to the content discovery layer. Both configurations in Fig. 1 have their respective advantages and drawbacks: the system architecture shown in Fig. 1(a) trades scalability for shorter lookup time. It is the other way round for Fig. 1(b). Our work focuses on the APRICOD overlay and we treat the content discovery layer as a black box.

The key construct behind APRICOD is a resource map. All media objects accessible by users constitute a n -dimensional resource space, which is partitioned into small cells. For instance, in a VoD system, a video is partitioned in the temporal dimension. A cell is thus a video segment. In a virtual environment, the virtual space can be partitioned in the spatial dimension, where a cell is an area. Each cell is managed by a logical node called *cell manager*. Content discovery in APRICOD is achieved by exploiting correlations among media objects managed by these cell managers. Recall that object Obj_2 is correlated with Obj_1 if peers are likely to access Obj_2 after accessing Obj_1 . Accordingly, two cell managers are correlated if the objects they manage are correlated. Each cell manager maintains a small number of links pointing to their most correlated cell managers. Links can be either fixed links, which corresponds to the obvious correlations, or dynamic links, which corresponds to non-obvious correlations and are updated on the fly. These cell managers and the links they maintain constitute the resource map.

Suppose a peer is currently accessing resources managed by a

cell manager H . The peer attaches to the cell manager H , and uses it as the query resolver to perform content discovery. Every cell manager maintains a content provider list for objects it manages. Therefore, if the peer queries for a list of content providers for objects managed by H , H would be able to answer the query directly. If the peer starts accessing objects in a cell that is out of the responsibility of H , the peer should find the corresponding cell manager K who is in charge of that cell first and then request a list of content providers from K . To discover K , the peer queries cell manager H as well, because it is very likely that H has a link pointing to K given the way we construct the resource map. If so, the query is a hit and resolves in just a single hop.

When H does not have a link to K , it resorts to the content discovery layer to find K and return K 's IP address to the peer. In addition, H should establish a dynamic link pointing to K , since the query indicates that K is correlated to H . Even though establishing the dynamic link does not help the querying peer, it will help future peers issuing the same query. In other words, H caches the query result. As the peer moves to access resources managed by another cell manager, it changes the cell manager it is attaching to and repeats the same querying process as described above.

We can observe that the resource map is constructed incrementally: a new dynamic link is established if a query misses. This construction process is indeed a learning process, which learns the correlations among cell managers, reflecting user access patterns. Due to the capacity limitation at each cell manager, we may want to limit the number of links a cell manager can maintain. If the capacity limit is reached, an existing dynamic link should be replaced. State-of-the-art cache replacement policies (such as LRU/ k and LFU) can be used to determine which dynamic link to replace.

If the peer has cached enough content managed by a cell manager, it registers itself as a content provider to that cell manager for those media objects it has cached. Accordingly, deregistration should be carried out when deleting the content from its cache.

Note that cell managers are logical nodes, which can be mapped to different peers in real implementation. Cell managers employ appropriate replication scheme (and thus logically never fail) to preserve the learnt correlations in the system.

4. EVALUATION

We evaluate APRICOD by collecting traces from a VoD flash player, which log the seeking behavior of 54 users while watching a lecture video over one semester. The 857s video is quantized into segments with equal length of 10s. Each segment is managed by a cell manager. Whenever the playhead moves from segment S_A to segment S_B , a query is issued to M_A (the cell manager in charge of segment S_A) to discover M_B . Each cell manager maintains two fixed links pointing to its neighbors, i.e., the cell managers that manage the previous segment and next segment in temporal order in the video. Apart from that, each cell manager also maintains n dynamic links pointing to non-neighboring cell managers ($n = 10$ by default), using LFU as the cache replacement policy.

First, we examine the correlations among non-neighboring segments since that is the fundamental assumption APRICOD is built on. Correlations among neighboring segments are obvious, so we do not examine them. Fig. 2 shows the correlations among non-neighboring segments mined from the trace. The X-axis denotes the top x portion of most popular non-continuous access links, and the Y-axis denotes the portion of queries attracted by the top x portion of most popular non-continuous access links. Meanwhile, the green *dotted line* denotes random user access pattern. The red *solid line* denotes user access pattern observed in the trace. Within the huge non-continuous access link space, only a small portion of

them get accessed. From the figure, we can conclude that there is significant amount of correlations even among non-neighboring segments. By knowing a small number of non-continuous access links, we would be able to answer a significant portion of non-continuous access queries.

Next we use the trace data to demonstrate how APRICOD evolves over time and to show how APRICOD works. For clarity, we limit the number of dynamic links to 5 and focus on the cell managers in charge of segments 60, 61, 62, 63 and 64 only (shown as blue nodes). Fig. 5 shows a serial of snapshots of the system states. Outgoing links of black nodes are not shown in Fig. 5. Neighbor links are not shown as well since they are always present. Each link is labeled with its access count. The thicker the link is, the more often it is accessed, indicating a stronger correlation.

When the simulation starts (Fig. 5(a)), there are no dynamic links in the system. After receiving 700 queries, dynamic link starts appearing. From Fig. 5(b) to 5(c) we start observing link replacement. For instance, the links $62 \rightarrow 55$ and $62 \rightarrow 57$ in Fig. 5(b) are replaced by $62 \rightarrow 59$ and $62 \rightarrow 69$ in Fig. 5(c). As more queries are received, the dynamic links adapt automatically to the user access pattern. One phenomenon that can be easily observed from Fig. 5(a) to 5(f) is that those links that indicate strong correlations tend to be kept over time. Note that access count of each link may not increase monotonically from Fig. 5(a) to 5(f). For instance, the link $62 \rightarrow 65$ has an access count of 3 in Fig. 5(e), but only 1 in Fig. 5(f). This is because the link is replaced and later added back between the 2800th and 3500th query. When adding back a previously replaced link, its access count restarts from 1.

Finally, the trace is fed into an APRICOD simulator to examine the caching efficiency. We group every 50 queries into a round according to the order they are issued, and then compute the hit rate of each round. Fig. 3 plots the CDF of query hit rate, including both continuous access queries and non-continuous access queries. The green line denotes the query hit rate when only neighboring links (fixed links) exist, handling only continuous access queries. The red line shows the hit rate when we have 10 dynamic links in addition to the fixed links to handle non-continuous access queries. Fig. 3 shows that the median hit rate increases from about 0.6 to 0.85 with the dynamic links introduced by APRICOD. The improvement is achieved by exploiting the correlations among non-neighboring segments as shown in Fig. 2.

Since neighboring links will always be present, continuous access queries will always hit with peer failure as an exception. Thereby, we focus our evaluation below on non-continuous access queries. Fig. 4 shows the hit rate of non-continuous access queries over time. We can observe the learning process from the curve. At the beginning of the simulation, the hit rate of non-continuous access queries is almost 0. After receiving 15 rounds of queries (each round consists of 50 queries), the hit rate increases to 0.7. We observe 1464 non-continuous access queries with 781 hits and 683 misses. Majority of the missed queries (668 out of 683) are due to the fact that they are seen for the first time. This observation suggests that we can get a higher query hit rate by training the system with more data.

We also evaluated APRICOD in a peer-to-peer virtual environment using traces from Second Life [5]. Similar improvement is observed. We omit the details here due to space constraint.

5. CONCLUSION

In this work, we proposed a general distributed content discovery caching middleware named APRICOD, aiming at shortening the content discovery time for non-continuous media access. APRICOD exploits the correlations among media objects, and is general

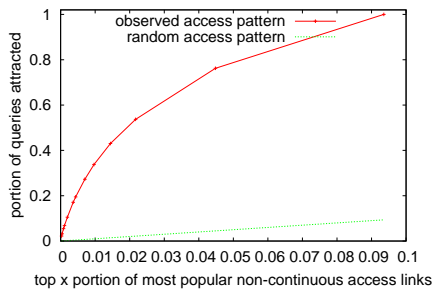


Figure 2: Correlations among non-neighboring segments observed in the trace

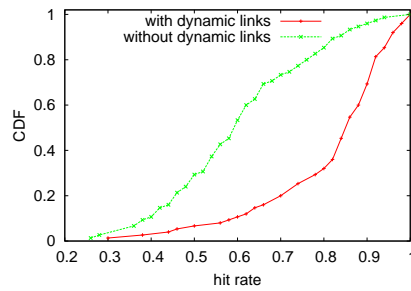


Figure 3: CDF of query hit rate

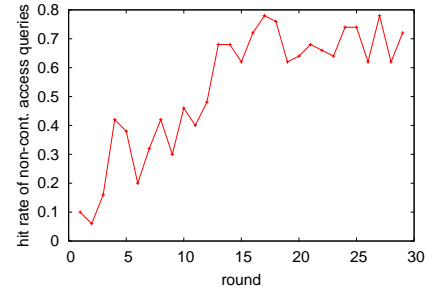


Figure 4: Non-continuous access query hit rate over time

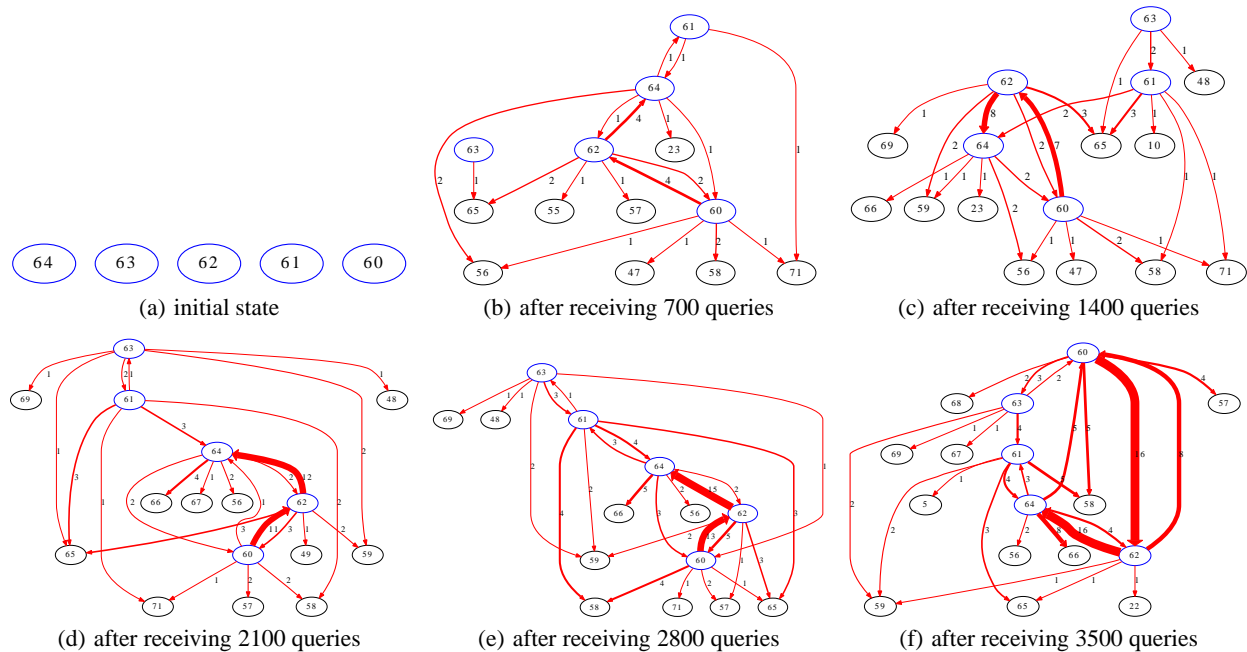


Figure 5: Evolution of dynamic links over time

enough to be used for various P2P media streaming applications such as VoD, virtual environment, zoomable video, and many others. The evaluation of APRICOD by user traces collected from a VoD player shows that it is able to shorten the content discovery time for a significant portion of non-continuous access queries to just one hop.

Acknowledgments: This research is supported by NExT Search Center, funded by the Singapore National Research Foundation & Interactive Digital Media R&D Program Office, MDA under the research grant WBS:R-252-300-001-490.

6. REFERENCES

- [1] B. Cheng, H. Jin, and X. Liao. Supporting VCR Functions in P2P VoD Services Using Ring-Assisted Overlays. In *ICC '07*, pages 1698–1703, Glasgow, Scotland, UK, June 2007.
- [2] W. Cheng, D. Liu, and W. T. Ooi. Peer-assisted view-dependent progressive mesh streaming. In *MM '09*, pages 441–450, Beijing, China, 2009.
- [3] X. Cheng and J. Liu. NetTube: Exploring Social Networks for Peer-to-Peer Short Video Sharing. In *INFOCOM '09*, pages 1152–1160, Rio de Janeiro, Brazil, Apr. 2009.
- [4] R. N. De Silva, W. Cheng, W. T. Ooi, and S. Zhao. Towards understanding user tolerance to network latency and data rate in remote viewing of progressive meshes. In *NOSSDAV '10*, pages 123–128, Amsterdam, Netherlands, 2010.
- [5] H. Liang, R. De Silva, W. Ooi, and M. Motani. Avatar mobility in user-created networked virtual worlds: measurements, analysis, and implications. *Multimedia Tools and Applications*, 45:163–190, 2009.
- [6] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips. TRIBLER: a social-based peer-to-peer system: Research Articles. *Concurr. Comput. : Pract. Exper.*, 20:127–138, February 2008.
- [7] X. Qiu, C. Wu, X. Lin, and F. C. Lau. InstantLeap: fast neighbor discovery in P2P VoD streaming. In *NOSSDAV '09*, pages 19–24, Williamsburg, VA, 2009.
- [8] X. Yang, M. Gjoka, P. Chhabra, A. Markopoulou, and P. Rodriguez. Kangaroo: video seeking in P2P systems. In *IPTPS'09*, Boston, MA, 2009.
- [9] A. P. Yu and S. T. Vuong. MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games. In *NOSSDAV '05*, pages 99–104, Stevenson, Washington, 2005.