DREAM: A Distributed fRamework for customized dEployment of a vAriety of indexing engines over Million-node overlays

Evangelos Sakkopoulos
Computer Engineering and Informatics Department, University of Patras
sakkopul@ceid.upatras.gr

Spyros Sioutas
Department of Informatics, Ionian University
sioutas@ionio.gr

George Papaloukopoulos
Computer Engineering and Informatics Department, University of Patras
papalukg@ceid.upatras.gr

Alexandros Panaretos
Department of Informatics, Ionian University
alex@ionio.gr

Dimitrios Tsoumakos
Department of Informatics, Ionian University
dtsouma@ionio.gr

Anastasia Saltou
Computer Engineering and Informatics Department, University of Patras
saltou@ceid.upatras.gr

ABSTRACT

In this paper we present a distributed framework that supports customized deployment of a variety of indexing engines over million-node overlays. The key aim is to provide the appropriate integrated set of tools that allows numerous applications with large-scale, different requirements to evaluate and test the performance of various application protocols for very large scale deployments (multi million nodes - billions of keys). Using lightweight and efficient collection mechanisms, our system enables real-time registration of multiple measures, integrating support for real-life parameters such as node failure models and recovery strategies. Experiments have been performed at the PlanetLab network and at a typical research laboratory in order to verify scalability and show maximum re-usability of our setup.

1. INTRODUCTION

Our era can be characterized by an explosion in the production of data: Web 2.0, business processes, government regulations, etc, all produce increasing volumes of information that need to be stored, indexed and queried. This is especially true for social information that, in the form of blogs, wikis, social bookmarkings, etc, allows users to constantly evolve their ways to communicate. In this plethora of information sources, personalization is necessary to enhance user-experience and allow access to the most relevant parts of the data. It has been reported [5] that over 80% of users prefer the numerous personalization services that businesses and social sites offer.

Personalization often entails the formulation and maintenance of user profiles that are either implicitly logged or explicitly given by users. The desired scalability and efficacy to handle the advanced processing required are provided by distributed solutions as both academic and business innovation has already indicated.

Contemporary distributed systems involve very large populations of commodity nodes independent of the specific underlying architecture: BitTorrent has reported stats where over 20 million daily active users download over 400,000 torrents on average and the number of users is already 100+ millions. Other P2P-based networks like Live messenger report 300+ millions of monthly active users online and 25+ million nodes (peak online time) in the Skype chat network.

On the other hand, the processing and retrieval needs per application range vastly: Social and web advertising sites perform map-reduce based intensive processing to identify trends and mine useful information (e.g., [10]), while realtime applications or simple analytics tools require efficient point-range-aggregate queries over bulk personalized datasets (e.g., [4]). Thus, it is also imperative that, besides the required scalability, the platform should be modular enough to provide support for different types of storage/processing engines.

Personalized access may occur at either of the *three levels* of a distributed system [3]: Data description, selection (i.e., querying) and result presentation. In this work, we present a framework that is able to efficiently and scalably support millions of physical cooperating nodes running different protocols and diverse personalized datasets on different levels of granularity. As such, its contribution closely relates to the selection and result presentation levels of access.

In this paper, we present the DREAM framework that is able to deliver and to provide two key features:
First, DREAM supports and facilitates the deployment of very large-scale (*multiple million*) P2P node systems. It also includes simple, easy and organized tools to achieve the collection of numerous statistics and execution of web-scale concurrent queries.
Second, its modular design enables the use of multiple indexing/processing engines, allowing different applications or even groups of users within the same application to cus-

SAC'12 March 25-29, 2012, Riva del Garda, Italy.

tomize their processing needs.

The DREAM system can simulate 600,000+ nodes in a stand-alone environment and millions of nodes as it can be executed in a distributed mode within a researcher's laboratory. In this sense, it takes advantage of the available resources from multiple computers, that usually exist in a typical laboratory or a research network. DREAM brings forward appropriate tools and ready to use infrastructure in order to facilitate the researchers' tasks in-line with the framework. Multiple protocols have been simulated in order to show that it takes couple of hours to initialize a network of 100000 nodes at a middle range configured PC.

Distributed simulations with DREAM can be designed and delivered with the same framework in different environments but with the same single setup. We show that only a small scale laboratory environment (5 computing machines) is an adequate environment to deliver multi - million node simulations at ease. Furthermore, we present the roadmap and demonstrate the results to deliver a small scale laboratory setup to a successfully verified experiment of six (6) protocol evaluations at the PlanetLab platform (http://www.planet-lab.org) with limited effort given resources available.

2. RELATED WORK AND MOTIVATION

`Web Systems Personalization` Adaptive web [2] makes it possible to deliver personalized views or versions of a hypermedia document, improving usability and thus productivity. The user profile records information concerning the user and his knowledge state. User profiles can be distributedly stored across several servers.

`Web Services Personalization` The process of combination for fine-grained multidimensional user models with knowledge representation and management techniques for making Web Services knowledge-aware usually involves a number of Web Services distributed in a network of numerous nodes [7].

`Social Web` The Social Web provides an unpreceded environment to gather user activity and data. User data is distributed across different Web systems, and the aggregation of such user profile information is transformed into a distributed computation task. A user's profile can be stored in different network systems that formulate a world of nodes with partial knowledge of each other in accordance to a shared-nothing architecture [1]. A social network service is essentially user oriented and thus the user is a basic entity of the system. The users are associated with specific, unique IDs in the network (UID). Content is the other important entity of our model. Content is an abstraction of every type of data that can be indexed, retrieved, and stored in the system. Mainly, it refers to files that users own and wish to make available to the social network. For indexing purposes, each content item is represented by a set of terms (e.g. keywords) that describe it. We call this set of terms the *content profile*. Consider the (popular nowadays) cloud infrastructures for social content profiles. Each node in the cloud maintains a tuple with attributes: UID, OS, load, NodeId, time, profile-keyword e.t.c. Collectively, these makeup a relation, CloudNodes, and we wish to execute queries such as:
SELECT COUNT(UID)
FROM Cloudnodes
WHERE TIME=last week AND UID=10234 AND NodeId IN Facebook
in order to find how many times the user with identifier 10234 visited facebook during the last week.

In order to study the growing number of distributed services and user profiles, research is closely related to the development and usage of distrirbuted frameworks that allow numerous applications with large-scale, different requirements to evaluate and test the performance of various application protocols for very large scale deployments (multi million nodes - billions of keys). The analysis of Naicken [9] shows that P2P simulators do not usually support the creation of more than 10,000 nodes [8]. Only a limited number of cases conducts experiments with more nodes [6], however even then no more than a few 100,000 P2P nodes, mainly because of memory requirements. The presented framework, using lightweight and efficient collection mechanisms can easily reach $> 600,000$ P2P nodes in a stand alone PC and *multi-million* P2P node in a small network that every research laboratory has. P2P simulators fail to include real-time registration of multiple measures as well as inline features to simulate easily and seamlessly (a) randomized node departures and (b) node failures (and node failure scenarios) to depict more realistically real life conditions [9], [6].

3. BASIC ARCHITECTURE

In this section, the basic architectural features are presented in short. The figures following depict the main parts of the framework environment, how the packages are related and the design decisions that have been made. The modular architecture allows the researcher to develop quickly and using a clear approach every new protocol that one might want to test on large scale simulations.

3.1 Admin Tools

To support a GUI for multiple protocol implementations and to allow wide customization in testing scenarios and collection of metrics, a number of *administration tools* have been packaged into the simulation framework. *GUI* allows the high level researcher and protocol designer to perform protocol testing without involving source code at all. Furthermore GUI facilitates validation of protocols by independent researchers in an easy and straight forward UI functionality that incorporates collection and presentation of metrics. Admin tools have specifically been designed to support *reports* on a collection of wide variety of metrics including, protocol operation metrics, network balancing metrics, and even server metrics. Such metrics include frequency, maximum, minimum and average of: number of hops for all basic operations (lookup-insertion-deletion path length), number of messages per node peer (hotpoint-bottleneck detection), routing table length (routing size per node-peer) and additionally detection of network isolation (graph separation). All metrics can tested using a number of different distributions (e.g. normal, weibull, beta, uniform etc). Additionally, at a system level memory can also be managed in order to execute at low or larger volumes and furthermore execution time can also be logged. The framework is open for the protocol designer to introduce additional metrics if needed. Futhermore, XML rule based *configuration* is supported in order to form a large number of different protocol testing scenarios. It is possible to configure and schedule at once a single or multiple experimental scenarios with different number of protocol networks (number of nodes) at a single PC

or multiple PCs and servers distributedly.

3.2 Overlay Scalability

Apart from the fact that a number of packages facilitate development, a number of different protocols are available as sample code and executables in order to allow familiarization with the development and usage of the simulator: Chord, Baton*, Nested Balanced Distributed Tree (NBDT), NBDT*, R-NBDT* with advanced load distribution and ART [11]. Moreover, the respective abstract classes and programming steps are depicted also at a simplistic *dummy protocol* in order to guide the programmers that first use the simulation framework proposed. The framework is particularly designed to allow large scale experimental evaluation of node based protocols. The incoroporation of metrics collection methods into the framework minimizes the memory needed and frees up space for more nodes to be executed. Moreover, distributed scenarios allow multiple computers to participate in the same experiment increasing radically the number of nodes simulated. The simulator is organized in packages as shown in Figure 1(a).

4. NODE FAILURE AND DEPARTURE

The presented framework supports new operations and services so that it provides services for node failure and network recovery and for node departures and substitutions. Simulators up to now tend to limit themselves to support (a) import of nodes for the creation of overlay network and (b) searching, (c) import and deletion of keys in nodes.

The classes and packages of the framework include all the necessary code parts to facilitate the researcher in order to detect, control departures and execute simulation using them. The message passing environment is designed to be possible to detect during simulations whether the message recipient is online or not (i.e. it does not take for a fact that the nodes are always available as done in rest of simulators).

It is common place that sudden departure of nodes without notice can bring large scale problems to routing messages within a P2P network. Such sudden simultaneous departures of multiple nodes makes difficult to failure recovery routing strategies find an alternative path to avoid failure nodes. In such cases, sending messages to the same node more than one time is probable. In order to overcome such cases, the simulator infrastructure includes tools to store all intermediate nodes that a message visited in its path for the sake of the simulation study even in cases that a protocol does not need such information. As a result, all path can be stored and studied after each message is sent. Moreover, to support departure and node failures for any protocol, node selection strategies take into consideration exception lists for nodes that have failed while running any distribution for experiments. Simulator facilities allow passing failed node lists and/or departed node lists to preprocessing of distributions utilized to retrieve randomized node ids during experimental runs.

DREAM determines the state of each node based on the state of each thread implementing it (RUNNABLE, BLOCKED, TERMINATED). However, in order to handle self willing node departures and sudden retirement of nodes, the node has to pass through different states during simulation, independent from the thread state that implements it. These states are fixed in the class named PeerState, that was created as a part of network Overlay package, and they are following: (a) WORKING: online and ready, (b) CANDIDATE_SUBSTITUTE: node substitution by existing node or nodes at the network, (c) VOLUNTARILY_LEFT: node self-willing departure, (d) FAILED: abrupt node failure.

Next, it is crucial to monitor for overlay network partitioning after failure of successive nodes, which results in isolation of nodes. This happens when all routing pointers to nodes outside the isolated partition are broken. In other words, the minimum number of routing pointers that should break in order that a group of nodes is separated from the overlay network equals to S as follows:

$S = \sum contacts\ of\ all\ nodes\ of\ team - \sum internal\ contacts\ between\ nodes$

Furthermore, details on the statistics needed to monitor failures and departures of nodes are: (a) The number of steps (hops) to find the suitable overlay network point for additions in the overlay network. This calculation is realized with the infiltration of message JOIN_RESP from the Network Filter class. (b) Number of steps to find a substitute when a intermediary node wishes to withdraw itself. This calculation is realized with the infiltration of message REPLACEMENT_RESP from the Network Filter class. (c) Number of search queries, additions or deletions of keys that failed, either because the expected node has departed, or because the network has been partitioned and it is not possible to access the requested destination. This calculation is realized with the sum of messages QUERYFAILED_RES from the Network Filter class. In particular, a main target of this framework is to maintain robust collection and analysis of statistical data that results from the experiments at all new failure detection and overcome scenarios strategies. The importance of experimental analysis is of more major importance, because it confirms the theoretical analysis, it elects problems and potential omissions that had not been located during the theoretical study, and constitutes a proof for the proposal that is presented. Moreover, the more realistic the simulation is, the smaller the divergence of experimental results from the real life results.

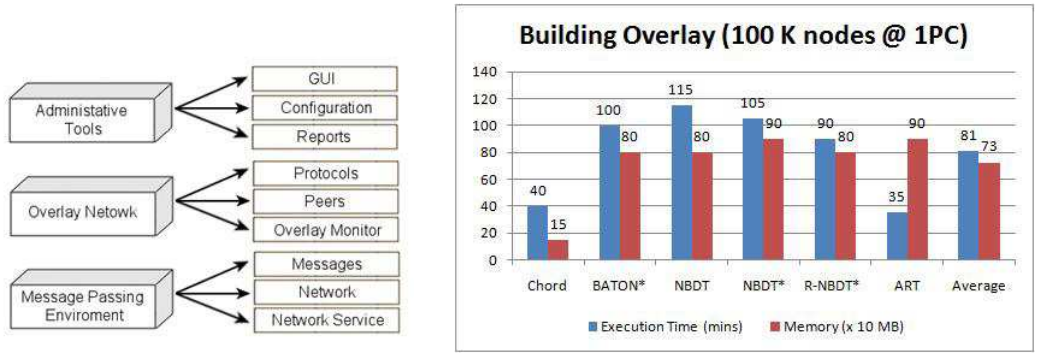
5. DREAM EXPERIMENTAL EVALUATION

In the following we present the large-scale simulations we executed in order to verify the efficiency of our novel framework.

5.1 Simulation Environment

A standalone experimental evaluation at a typical research laboratory computer can be performed either using the GUI or filling out XML configuration files with the necessary the parameters of execution.

First, we demonstrate the efficiency of our framework to deploy a number of distributed protocols and test their lookup performance. Using a single-PC configuration (Intel Core2 Duo CPU @ 3GHz, 3GB RAM) we simulate 100K node overlays with *six* different protocols: Chord, BATON*, NBDT, NBDT*, R-NBDT* and ART. Results that register the required time for overlay construction and memory requirements are presented in Figure 1(b). Firstly, we note that even a single typical PC can easily host wide scale experiments. Secondly, our framework easily host a wide variety of protocols and manages to very efficiently build a large overlay using minimal resources: At most 1 GB of memory is required for 100K overlay construction and full functionality. Execution times for this mode are also very small,



(a) Simulator main Packages

(b) Multiple Protocol Support: Execution time and memory requirements

Figure 1: Simulator modules and main performance measurements

ranging from 35min to at most 115min for NBDT.

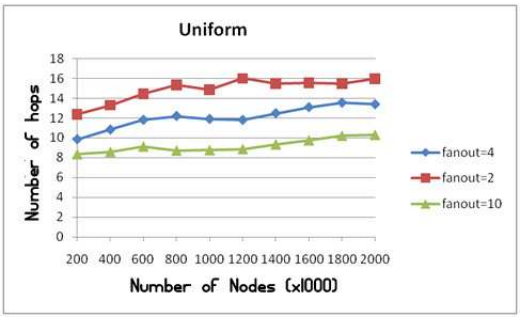
The distributed environment that was used for the remainder of the experiments consists of a total of 5 PCs (Intel(R) Xeon(R) CPU E5504 @ 2.00GHz, 8BG RAM) running over Ubuntu 9.10. To show the potential of the framework we evaluated the performance of a number of protocols. Exploiting the efficiency and its ability to function in a distributed environment, we executed large-scale simulations, altering the size of the network from 100,000 up to 2,000,000+ nodes with different flavours of protocols. Node failure and recovery scenarios have been evaluated as well as node departures and substitutions to test fault tolerance per protocol. Trials have been made with single, and 2-core up to 8-core processors and with 0.5 GB up to 16 GB of RAM memory verified that the simulator scaled without problems and did not need any changes of its configuration. This is also verified for experiments with 1 up to 12 PCs which is a number of PCs expected available at a typical University Lab (WAN experiments are also done on `50 Planetlab points`, please see section below). Furthermore the number of nodes simulated scaled near-linearly.

Figures 2 and 3 present results while experimenting with protocol Baton* for up to two million nodes. The performance of P2P protocols is greatly dependent on the average path length between two random network nodes. Figure 2(a) shows the path used in order to define insert and deletion of keys to nodes. Results verify that the cost is logarithmically increasing with the network population. Moreover, experiments show that query costs are decreasing with a fanout increase. In particular, the gain received from the fanout increase is larger when the network becomes more massive. This is expected as the larger the number of nodes, the smaller the tree height becomes with the fanout increase (in terms of rate). As a consequence, we have verified for overlays up to 2M nodes that the cost of search, insert and delete in Baton* protocol is $O(log_m N)$, where m is the fanout factor and N is the number of nodes. Note that original results presented in [8] are up to 10K nodes.

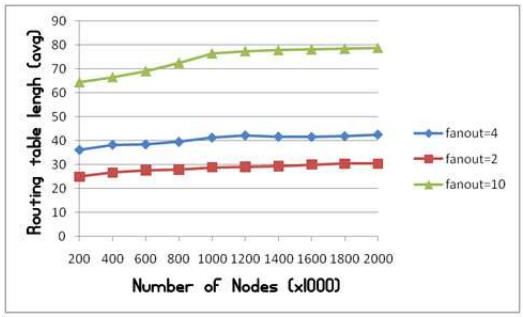
Figure 2(b) shows the average routing table length. It is clear that with an increase in the network size, the routing information that has to be maintained per node is also increasing: Increasing the number of nodes, more levels are created in the tree and therefore, the number of neighbours maintained at each node is increasing too. Moreover, a fanout increase leads to an increase of routing table length and, as a result, the cost of updating it. Experiments verify that faster search is possible using larger routing tables.

Figure 3(a) presents the load that each node receives, counting the number of messages received for any of the operations for node populations up to 2M nodes and for 3K operations. As shown, the maximum number of messages in the experiments reached 13 and this only happens for a single node. On the contrary, 30,590 nodes receive a single message. As a result, we note that the protocol successively balances the load among nodes. This also verifies that DREAM includes all the necessary features and tools to detect and research on load balancing techniques for P2P network protocols. As discussed in section 3, a list of collectable metrics are available to evaluate and experiment with protocols at application, protocol and system level. Figures 4(a) and 4(b) show a number of metrics for Chord, and ART, while figure 4(c) shows more metrics of Baton* in Planetlab.

We also show that our framework is able to collect connection-specific metrics that can be used to evaluate node-failure and recovery strategies on networks with multi million nodes. The problem that massive failures cause is the invalidation of links among them. As the search procedures have to overcome non-reachable connections, it is hard to choose a path that does not include failed nodes. Queries oscillate inside the overlay until the alternative path is determined. Thus an increase in node failures is expected to result in an increase to search costs. In our setup, a network of 1M nodes is initialized, with a 10% of randomly chosen nodes being assigned to leave abruptly (sudden node death), without rebuilding the network. The experiments continue increasing the percentage of nodes failing in the network by 1% at each round. In each step, the network is checked in order to verify that it is not partitioned into isolated areas that cannot communicate. All the experiments are repeated for fanout 2 up to 10 for Baton*. Figure 3(b) shows the average number of nodes that is expected to fail before the network is partitioned. The results verify that the network is resistant to failures when a quarter of the nodes fails for fanout=2. A

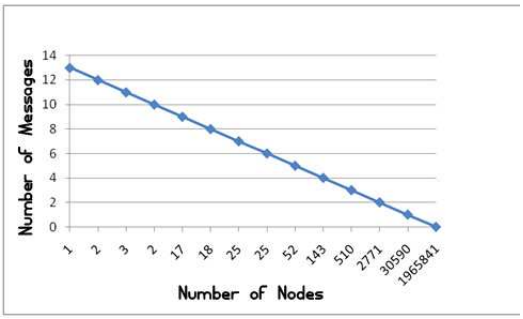


(a) Lookup cost

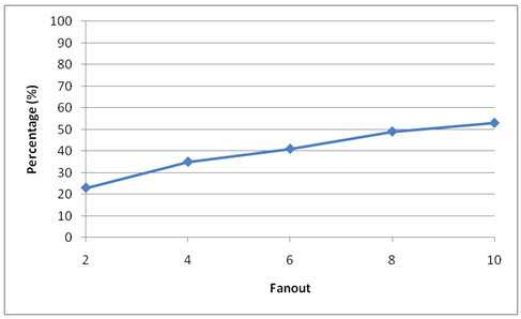


(b) Routing table length

Figure 2: Baton* Simulations with up to 2 Million P2P nodes (1)



(a) Messages per node



(b) failure percentage

Figure 3: Baton* Simulations with up to 2 Million P2P nodes (2)

fanout increase results in higher degrees of tolerance.

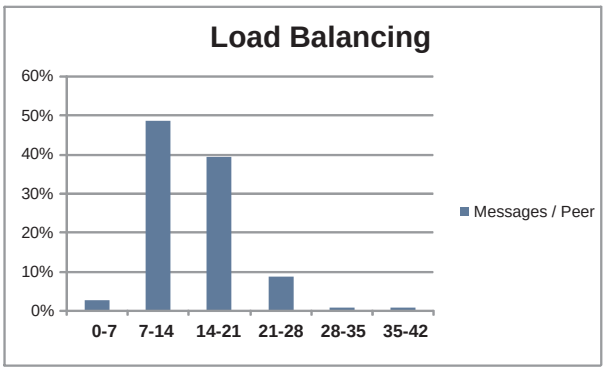
5.2 DREAM at the PlanetLab

PlanetLab is a distributed research networks infrastructure comprising 1091 computers, which are found in 505 different research locations. Each network participant allocates computers and is given the possibility to use resources from all the network to implement large scale experiments. The need for transparent transfer of experimental configuration and evaluation into large- scale infrastructures with limited changes is a common vision for researchers and research developers. DREAM has been verified to be easy to setup and execute at the PlanetLab network using the same experimental configuration as that inside a research lab. In order to avoid involvement of PlanetLab computers that are out of reach or overloaded the CoMon tool can be utilized that it provides statistics for PlanetLab available resources, both at nodes and slice level.

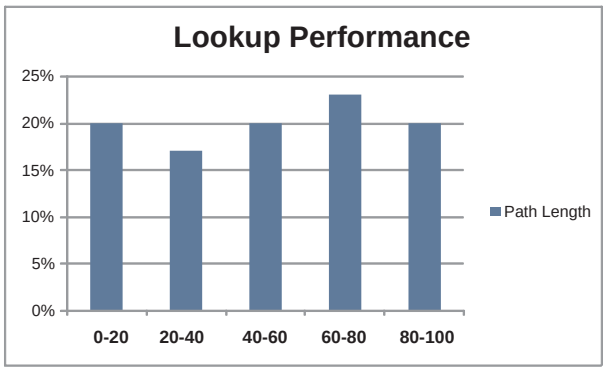
In the experimental simulation scenarios, 50 PlanetLab points were selected, based on their statistics according to the CoMon tool, so that they would not face severe problems of network interconnection. In each evaluation, node failure and recovery scenarios have been evaluated as well as node departures and substitution in order to test fault tolerance and robustness of each protocol tested. During experimentation at the PlanetLab, we observed that execution times are an order of magnitude larger than those required in our lab. There exist cases where 6,000 P2P nodes in PlanetLab take approximately 7.5 hours to return results. Slices of the PlanetLab are often over-loaded and large-scale experiments face enormous delays. Moreover, network communication among PlanetLab slices is an additional overhead that has to be taken into consideration when planning experiments. However, Planetlab is useful in order to verify that an engine under research is possible to work even under very large communication obstacles and most importantly in wide area networks. Figure 4(c), one notices how DREAM is executed for Baton* P2P at the planetlab nodes.

6. CONCLUSIONS AND FUTURE STEPS

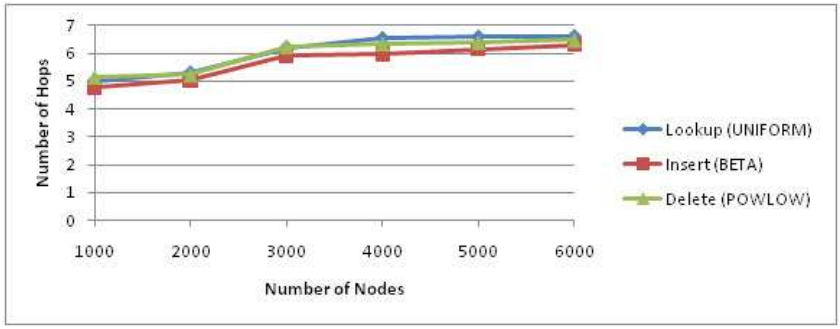
DREAM is presented as a novel framework to support simulation with multi million nodes. Contrary to the rest of P2P simulators, it supports the deployment of any distributed indexing and processing engine. Moreover it includes a robust framework to work easily on failure and recovery scenarios for the networks under design or research. Furthermore, robust statistics are supported and all operations are available through a simple GUI developed using Java tools. It is an ideal platform for multiple applications storing and quering in a fully customized way. Experimental evaluation in multi million node networks for the Baton*



(a) Chord 10K netwrk msg/node



(b) ART 10K netwrk pathlength



(c) Baton* Operation costs vs. network size in Planetlab

Figure 4: More metrics for Chord, ART and BATON*

protocol verified that the protocol indeed scales according to the theoretic analysis for up to 2 M nodes and this was possible to do with DREAM. Finally, the framework presented is easily used *as-is* within the PlanetLab, too. Future steps, include the HCI analysis of GUI simulators in order to detect common problems that P2P researchers face.

7. REFERENCES

[1] F. Abel, N. Henze, E. Herder, and D. Krause. Interweaving public user profiles on the web. In *User Modeling, Adaptation, and Personalization UMAP 2010, 6075 LNCS*, pages 16–27, 2010.

[2] P. Brusilovsky, A. Kobsa, and W. Nejdl. The adaptive web, methods and strategies of web personalization. In *4321 LNCS*, 2007.

[3] M. J. Carman and F. Crestani. Towards personalized distributed information retrieval. In *ACM SIGIR*, 2008.

[4] L. Chen, B. Cui, and H. Lu. Constrained skyline query processing against distributed data sites. *IEEE TKDE*, 23:204–217, 2011.

[5] Personalization reports. http://www.choicestream.com/who/news.php.

[6] J. Cowie, H. Liu, J. Liu, D. Nicol, and A. Ogielski. Towards realistic million-node internet simulation. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, June 28 - Jully 1, Las Vegas, Nevada, USA*, pages 2129–2135, 1999.

[7] A. Cuzzocrea. Combining multidimensional user models and knowledge representation and management techniques for making web services knowledge-aware. *Web Intelligence and Agent Systems: An international journal*, 4:289–312, 2006.

[8] H. V. Jagadish, B. C. Ooi, K.-L. Tan, Q. H. Vu, and R. Zhang. Speeding up search in peer-to-peer networks with a multi-way tree structure. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29*, pages 1–12, 2006.

[9] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers. The state of peer-to-peer simulators and simulations. *SIGCOMM Comput. Commun. Rev.*, 37(2):95–98, 2007.

[10] M. Shmueli-Scheuer, H. Roitman, D. Carmel, Y. Mass, and D. Konopnicki. Extracting user profiles from large scale data. In *MDAC*, 2010.

[11] S. Sioutas, G. Papaloukopoulos, E. Sakkopoulos, K. Tsichlas, Y. Manolopoulos, and P. Triantafillou. Brief announcement: Art–sub-logarithmic decentralized range query processing with probabilistic guarantees. In *PODC*, 2010.