# PASS It ON (PASSION): An Adaptive Online Load-Balancing Algorithm for Distributed Range-Query specialized Systems

Ioannis Konstantinou, Dimitrios Tsoumakos and Nectarios Koziris

Computing Systems Laboratory
School of Electrical and Computer Engineering
National Technical University of Athens
{ikons, dtsouma, nkoziris}@cslab.ece.ntua.gr

## 1 Introduction

A basic requirement for every P2P system is fault-tolerance. Since the primary objective is resource location and sharing, we require that this basic operation takes place in a reliable manner. In a variety of situations with skewed data accesses (e.g., [1] , etc) the demand for content can become overwhelming for certain serving peers, forcing them to reject connections. In many cases, these skewed distributions take extreme forms: *Flash crowds*, regularly documented surges in the popularity of certain content, are also known to cause severe congestion and degradation of service [2]. Data replication techniques is one commonly utilized solution to remedy these situations. Nevertheless, there are cases in which the requested resources cannot be arbitrarily replicated. Distributed data-structures that support range-queries is such an example: The keys are stored in the network nodes so that a natural order is preserved. These structures can be very useful in a variety of situations: On-line games , web servers , data-warehousing , etc. In such cases, adaptive and on-line load-balancing schemes must be employed in order to avoid resource unavailability and performance in a variety of workloads [3,4].

**Our contribution** In this work, we present *PASSION*, an on-line, adaptive load balancing algorithm that operates on distributed range-partitioned data structures. Our algorithm operates in a completely decentralized manner and requires no kind of global coordination. Its goal is, through key exchange between neighboring nodes according to their current load and individual thresholds, to counterbalance the inequality in load that affects performance. Each peer, upon sensing an overload situation relative to its individual threshold, requests help and proactively sheds a suitable part of its load to its neighbors. Load moves in a "wave-like" fashion from more to less loaded regions of the structure adaptively.

## 2 PASSION

The main idea behind *PASSION* is the following: When the current load of a node exceeds its self-imposed threshold $thresh_i$, the node sends a HELPREQUEST message containing its current load to one of its neighbours. The recipient node takes over a portion of the overloaded node's key range. This procedure is performed online, that is, nodes continue to serve requests during the key transfer. The recipient then estimates his new load and if this is above its local threshold, it initiates a new HELPREQUEST towards another neighbouring node. The procedure continues *TTL* hops away at most or until all nodes have successfully shed their load below their thresholds.

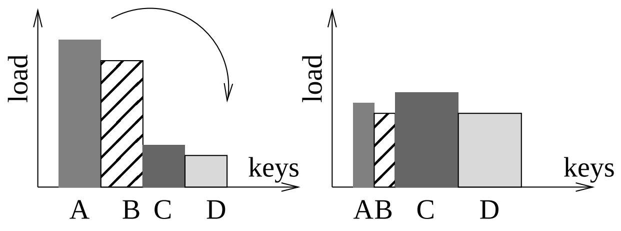

**Fig. 1.** PASSION example

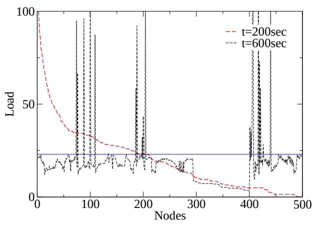

**Fig. 2.** Before balance, $\theta = 1$

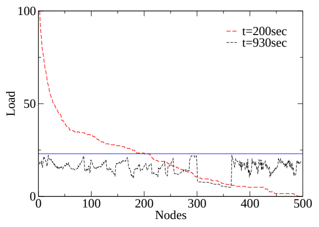

**Fig. 3.** After balance, $\theta = 1$

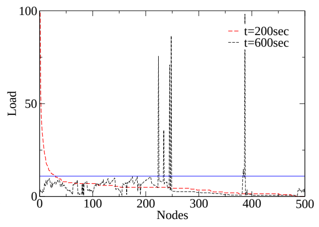

**Fig. 4.** Before balance, $\theta = 1.4$

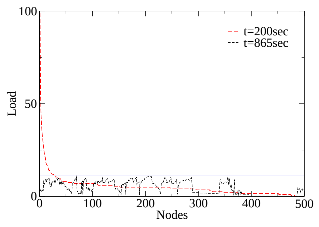

**Fig. 5.** After balance, $\theta = 1.4$

In order to calculate the portion of load that the overloaded node needs to shed, we introduce the *overThres* threshold, where $overThresh_i > thresh_i$. If the *splitter*'s load is above the *overThresh*, then only a fraction *a* of the extra load is accepted. Otherwise, the *splitter*'s excessive load is fully accepted. Like the simple *thresh*, *overThresh* is a local per-node setting. Its purpose is to smooth out the key/load exchanges between sequential *PASSION* executions.

## 3 Initial Results

We present an initial simulation-based evaluation of our method. We assume a network size of 500 nodes, all of which are randomly chosen to initiate queries at any given time. More specific, we apply passion on our simulator with load generated by: a zipfian distribution for $\theta = 1$ and $\theta = 1.4$ (see Figures 2–5).

## References

1. Cha, M., Kwak, H., Rodriguez, P., Ahn, Y., Moon, S.: I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In: IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement. (2007)
2. Jung, J., Krishnamurthy, B., Rabinovich, M.: Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In: WWW. (2002)
3. Karger, D.R., Ruhl, M.: Simple efficient load-balancing algorithms for peer-to-peer systems. Theory of Computing Systems **39** (November 2006) 787–804
4. Ganesan, P., Bawa, M., Garcia-Molina, H.: Online balancing of range-partitioned data with applications to peer-to-peer systems. Proceedings of the Thirtieth international conference on Very large data bases - Volume 30 (2004) 444–455