Ashiwan Sivakumar **Research Statement, April 2017**
Email: asivakum@purdue.edu
`http://web.ics.purdue.edu/~asivakum/`

An eternal quest to provide better *Quality of Experience (QoE)* for Internet applications, has engendered decades of research in designing protocols and systems. Ensuring high QoE is imperative, since it has direct impact on user engagement and the revenue of many businesses. For instance, many studies have found that lowering end-to-end latencies of Web downloads is critical (e.g., Akamai found that if an e-commerce site takes >3sec to load, users tend to visit other sites[1] and Amazon found that 100ms of latency costs 1% in sales[2]). On the other hand, the need to ensure high QoE is constantly challenged by new trade-offs owing to the highly dynamic nature of the Internet Ecosystem (e.g. evolving communication technologies, changing application characteristics etc.). Consequently, this opens up new research opportunities in understanding the trade-offs and designing systems to improve user experience.

Motivated by the practical problem based on our observation that cellular downloads of top Alexa Web pages *have latencies in the order of seconds and are 6X* slower than wired downloads in desktops [9], the primary focus of my research is on the question – *How do we reduce this gap and ensure a responsive Web browsing experience?* To this end, I seek to adopt a principled approach, beginning with real-world measurements to gain insights into the challenges in solving the problem and the shortcomings of existing solutions. Based on the insights, I design and implement scalable software systems inspired by an awareness of *application characteristics (Web page download process)* and *metrics (page load metrics like onLoad())*. Further, I employ extensive empirical evaluations of all my designs with real workloads (top Web pages) under live network settings (LTE).

In recent years, we have witnessed several trends that pose challenges in ensuring a responsive Web browsing experience for users – (i) Web pages have become complex and rich interactive applications with content that is customized for user preferences, (ii) There is an explosive growth in Web activities on resource-constrained mobile devices such as smartphones and tablets connected over high latency cellular links (and poor links in developing regions). Rendering a Web page requires the mobile client to parse HTML, CSS and execute Javascripts (JS), resulting in a lot of HTTP transactions in the high latency cellular link. Thus, today's Web download process is ill-suited for cellular networks. While new protocols like SPDY (significant part of HTTP2 standard) seek to reduce network delays, studies have shown that complex object dependencies in real Web pages limit SPDY's potential to lower page load times. In an attempt to tackle these challenges, my thesis makes several contributions –
• PARCEL[9] modifies the page load process by judiciously refactoring browsing functionality between a proxy in the cellular network edge and a client (leveraging opportunities offered by cloud computing and edge computing). • In our prior work[7], we empirically showed naive cloud-heavy thin-client browsing approaches can hurt performance with user interaction; Subsequently, PARCEL employs *redundant execution*, where the proxy executes to identify objects and proactively pushes them to the client and the client executes again to support user interactions locally. Evaluations in live LTE show that PARCEL achieves, on an average, *2X* reduction in page load latencies over traditional browsers.
• In [8], we address the real-world scaling challenge associated with the computational overheads of execution-based proxies[9, 4] (dominated by Javascript execution) that must support millions of users, by developing a technique (similar to program slicing) to identify and execute only the JS code required to fetch objects and skipping other code.

Further, my other significant works reduce Web latencies by optimizing the other constituent layers such as CDNs (content-delivery networks) and back-end data storage layer which are relevant in non-cellular settings. Specifically, a page structure aware Web object placement and caching in CDN caches; an intelligent data replica configuration and placement in geo-distributed datastores to reduce percentiles of latencies under normal operation and failure.

In what follows, I will describe my thesis research, followed by future research directions that I wish to pursue.

Thesis Research:

Proxy-based redundant execution for low latency Web over cellular networks

In [9] we make a fresh attempt at improving the Web download process by cutting network latencies that dominate page load delays in cellular settings. We propose PARCEL, a new proxy-assisted mobile Web browsing system that lowers page load latencies and further reduces the radio energy usage. I started working on my thesis research as part of my internships in AT&T Labs. Many cloud-based mobile browsers exist in the market e.g. Amazon Silk, Opera Mini, Android Chrome Beta and Sky Fire. In [7], we empirically demonstrate that a cloud-heavy thin-client approach taken by some browsers,can hurt latencies with user interactions(e.g. clicks) because they may result in the click to be sent to a proxy in the cloud to execute the Javascript responsible to handle the click.

Consequently, PARCEL judiciously refactors browsing functionality between the mobile device and a proxy based on their respective strengths, in a manner distinct from traditional browsers and cloud-heavy approaches. The key idea is **Redundant Execution**, (i) proxy parses HTML, CSS and executes Javascripts (JS), to identify objects quickly and proactively pushes them to the client; (ii) client executes again to support interactive operations locally and avoid network communications to the extent possible; Further, the proxy supports cellular friendly data-transfers by flexibly pushing objects in a manner that balances latency and radio energy usage. In realizing PARCEL, we address some pragmatic challenges related to client customization of Web pages and HTTPS traffic by proposing to use a *personalized trusted proxy*. Since some URLs could depend on client parameters such as User-Agent, screen size, devicePixelRatio etc., the proxy emulates these settings and stores client cookies, cache state, to ensure URLs fetched by the proxy match that of the client. We validate PARCEL through extensive evaluations in live LTE network settings, and compare its performance to both a traditional browser, and an existing cloud-heavy browser. On average, PARCEL reduces page load latencies by 2X compared to traditional HTTP1.1 based browsers and supports energy-efficient interaction unlike cloud-heavy browsers. Moreover, I did a demonstration of a working prototype of Parcel proxy and client (Android) systems in the *2016 AT&T academic research summit.*

Scaling execution-based Web proxy design by reducing JS computation overheads

While redundant execution for Web proposed by systems like PARCEL[9] and Cumulus[4] is simple and effective in cutting network latencies, which dominate page load delays in cellular settings, a key concern is the scalability of the proxy which must execute JS for many concurrent users. Further, the concern is pertinent to the plethora of execution based split browser architectures (e.g., Opera Mini, SkyFire) as well. Deploying such a solution at scale, serving millions of users requires the execution overheads are economized (e.g. our measurements indicate 2000 users can be supported on a server with 32 cores and 128 GB RAM, translating to $2.5Million in CAPEX cost alone to support 1Million users, assuming server cost of $5000). Motivated to solve this practical scaling challenge, in [8] we take the first step in reducing the computational overheads at the proxy (dominated by JS execution). We solve the problem in the context of *redundant execution*[9, 4] focusing on a design where the proxy's execution is solely to push the needed objects and the client completely executes the page as normal. We develop a *light-weight* technique similar to program slicing to identify and execute in the proxy only the JS code necessary to identify and push the objects required for the client page load, while skipping other code. While JS program slicing is a hard and open research problem, we leverage the observation that it is acceptable to approximate the program slice in the proxy given the clients complete execution. We solve several key challenges in computing the slices and minimizing the associated overheads. Experiments with top Alexa mobile pages show the design can sustain, on average, 27% more user requests per second than a proxy performing fully redundant execution([9, 4]), while preserving, and sometimes enhancing, the latency benefits (20% latency savings for 15% of the pages). Further, our light-weight redundant execution approach provides, on an average 1.5X latency savings compared to SPDY and 1.2X compared to only pushing a subset of objects embedded in the main HTML.

Other Research Projects

With more Web pages opting for all-content delivery through CDNs, in[5] we show there is an opportunity to improve the end-user experience through careful management of Web objects in CDN caches taking page structure into account. The key idea is that the objects in a Web page having the largest impact on page latency should be served out of the closest caches in the hierarchy. We present and evaluate a family of algorithms for CDNs to identify important objects in a page and develop mechanisms to serve critical objects from the closest CDN cache, while balancing the traditional CDN concern to cache popular objects in the closest cache to minimize bandwidth penalties. Through extensive experimental evaluation we show latency reductions of over 100ms can be obtained for a lot of pages.

Besides Web performance, I have worked on improving the performance of geo-distributed applications deployed in the cloud. In order to meet the stringent SLAs, modern applications use geo-distributed key-value data stores for data replication seeking to lower latency and provide five 9's of availability. Given the scale of the applications and their dynamic workloads which are not exposed directly to data stores, configuring replication parameters of data stores is largely adhoc. In [6], we propose systematic frameworks for intelligent replica configuration and placement in geo-distributed data stores, considering the application data access patterns and consistency requirements for quorum-based data stores, optimizing percentiles of response times under normal operation and under a datacenter (DC) failure. In [3] we characterize the performance of geo-distributed multi-tier Web applications by deploying them on two commercial cloud platforms – Amazon EC2 and Microsoft Azure. The performance fluctuations were largely short-lived, impacting only some of the application components and motivated the need for a fine-grained application component level re-routing of requests to meet performance SLAs.

Future Research Directions:

Over the years, in contrast to the end-to-end arguments in system design, computer networks have taken up ever-growing list of functionalities beyond packet forwarding, such as flow monitoring, access control, compression, Web acceleration etc.; possibly enhancing the application QoE, beyond what is possible with a pure end-host implementation. Moreover, we are currently living in a *virtual world* where networks are being virtualized motivated by flexible manageability and cost savings. These trends will continue for the foreseeable future as evidenced by the 5G proposals, posing several challenges and opportunities for future research.

PARCEL[9] can be conceived as a user personalized Web acceleration functionality deployed in the edge network. While my thesis research is a first step in lowering Web page latencies, there are many recent proposals to run application code (e.g. for VR, AR applications) within the network (edge computing proposals & 5G drafts) to provide better user experience. There are many open challenges that need to be solved to realize the full potential of running application functions in the network. The first direction is around *orchestration and management* of such compute instances running application code or network functions. In PARCEL's context, How should the proxy instances be instantiated and the state maintained or migrated given users inexorably move in and out of the network? How often should the local client state (cache, cookies etc.) be shipped to the proxy, given clients might perform local interactions with a Web page or use other interfaces (Wi-Fi)? Another direction is how to provide the functionalities with the surge of *HTTPS usage* so that user privacy is protected. This problem requires broader collaboration with cryptography and privacy experts. A third direction is motivated by the question – *Can we systematically model application QoE (e.g. page load time) to validate system designs for a given network characteristic?* An approach is to first start with conducting user studies of different mobile applications to understand what is a good quality metric that reflects user-perceived performance. This opens up collaboration with HCI experts. Then correlate the QoE metric with network performance metrics such as delay, bandwidth, loss rate etc.

Video streaming is another killer application contributing to a significant fraction of mobile data traffic. With the advent of IPTVs, Facebook live and 360 videos the way in which videos are consumed is changing. While Facebook live demands a huge effort in designing scalable networks to handle unpredictable and peak load, designing better adaptive bit rate (ABR) algorithms for 360 videos may improve the user experience. Further, with the unique bandwidth characteristics of 5G networks (unpredictable with periods of high bandwidth), there is a necessity to predict bandwidth and perform aggressive buffering of video chunks when bandwidth is high. A longer term goal for my research is to enable many mission-critical applications on resource-constrained smart phones by exploring *light-weight redundant execution as a means* to boost their performance. Similar to parcel, a light-weight agent executes a piece of the application code inside the network with the aim of enhancing overall application QoE. There are many questions to explore – If we run these functionalities on general-purpose servers, can we measure the overhead on CPUs? How to make the network agent light-weight and scale to many users? Can hardware acceleration (using FPGAs or GPUs) be used to speedup some of these functionalities? Another direction I would like to pursue in the long run is to *apply program slicing techniques for network fault diagnosis and debugging.* It is paramount for network operators to automatically diagnose faults within the network and debug network functions, beyond providing good QoE. The scope of the problem is complicated by the dynamicity introduced by virtualization. While there is a rich body of research around critical slicing and delta debugging, applying such techniques to network fault diagnosis may introduce new challenges around how to be effective given the dynamicity in networks, inter-dependencies among multiple network functions etc.?

To conclude, given these trends in networking, and how users access applications on different devices, I believe there are lots of exciting challenges as well as opportunities to design, verify and manage scalable networked systems.

References

[1] End-Users' Web Experience Expectations Just Keep Getting Higher. `https://blogs.akamai.com/2012/11/end-users-web-experience-expectations-just-keep-getting-higher.html`.

[2] EATON, K. How One Second Could Cost Amazon 1.6 Billion In Sales, 2012. `https://www.fastcompany.com/1825005/how-one-second-could-cost-amazon-16-billion-sales`.

[3] Hajjat, M., Pn, S., Sivakumar, A., and Rao, S. Measuring and characterizing the performance of interactive multi-tier cloud applications. In *The 21st IEEE International Workshop on Local and Metropolitan Area Networks* (April 2015), pp. 1–6.

[4] Netravali, R., Sivaraman, A., Das, S., Goyal, A., Winstein, K., Mickens, J., and Balakrishnan, H. Mahimahi: Accurate record-and-replay for http. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)* (2015), pp. 417–429.

[5] Puzhavakath Narayanan, S., Nam, Y. S., Sivakumar, A., Chandrasekaran, B., Maggs, B., and Rao, S. Reducing latency through page-aware management of web objects by content delivery networks. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science* (New York, NY, USA, 2016), SIGMETRICS '16, ACM, pp. 89–100.

[6] Shankaranarayanan, P. N., Sivakumar, A., Rao, S., and Tawarmalani, M. Performance sensitive replication in geo-distributed cloud datastores. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (June 2014), pp. 240–251.

[7] Sivakumar, A., Gopalakrishnan, V., Lee, S., Rao, S., Sen, S., and Spatscheck, O. Cloud is not a silver bullet: A case study of cloud-based mobile browsing. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications* (New York, NY, USA, 2014), HotMobile '14, ACM, pp. 21:1–21:6.

[8] Sivakumar, A., Jiang, C., Nam, Y. S., Narayanan, S. P., Gopalakrishnan, V., Rao, S., Sen, S., Thottethodi, M., and Vijaykumar, T. Scalable proxy execution for low latency Web over cellular networks (Under submission).

[9] Sivakumar, A., Narayanan, S. P., Gopalakrishnan, V., Lee, S., Rao, S., Sen, S., and Spatscheck, O. PARCEL: Proxy Assisted BRowsing in Cellular networks for Energy and Latency reduction. In *Proceedings of the Tenth ACM Conference on Emerging Networking Experiments and Technologies* (2014), CoNEXT '14.