Poster: Accelerating Applications in the Fast-moving Devices with Proactive Provisioning

HyunJong Lee†, Hee Won Lee§, Moo-Ryong Ra§, Yu Xiang§, and Jason Flinn†
† University of Michigan § AT&T Labs - Research

An increasing number of applications that leverage built-in sensors is hosted on new types of devices such as vehicles and drones. Example applications are augmented reality Head-Up-Display in a connected car [2], scout drone that chases a target suspect among the crowd [3], autonomous fleet drones [1], and so on. These futuristic applications in various genres of devices are latency-sensitive and require a significant computation power to process a tremendous amount of data from built-in sensors [8]. However, the capabilities of processors shipped with these devices are limited. A natural solution is offloading to an edge surrogate,[1] running on a 'nearby' edge-cloud that offers more computation capacity at low latency.

The key to retain low latency in edge-cloud is placing the surrogate in the nearest edge node. If an end-device moves to a new location and connects to a new nearby edge node while the corresponding surrogate stays at the previous edge node, user-perceiving latency increases to communicate over WAN [9, 10]. Provisioning the surrogate from one to another edge nodes requires migrating the state as applications are often stateful.

Prior systems [5, 6, 12] have thoroughly investigated *just-in-time* provisioning mechanisms. They reduce the total provisioning time in order to migrate and resume a surrogate instantenously when the end-device associates with a new edge node to retain low latency. Because they target applications hosted on devices that are mostly used in stationary or slow-moving scenarios (e.g., smartphones, tablets), they passively migrate a surrogate's state on demand (e.g., watching VR video at home and later on, resuming to watch at the airport). Thus, prior systems primarily focus on reducing the transfer size since the size of a surrogate's state dictates the total provisioning time. For applications in those devices, the downtime – an application becomes unresponsive – has negligible effect on user-experience; applications in fast-moving devices should consistently deliver low latency for a good user-experience. Hence, both overall latency and downtime have significant impact on user-exprience.

Our proposed solution to reducing both overall latency and downtime for applications in the fast-moving devices is by proactively provisioning the state of an actively running surrogate from current to new edge nodes that are predicted to associate in the near future by end-device. Rather than migrating the state after end-device switches to an edge node, i.e., just-in-time, as done by prior systems [5, 6, 12], we will make predictions on next edge nodes and proatively provision forecasted next nodes while the end-device is still using the surrogate in the current edge node. Since the provisioning stage takes a place before an end-device switches to a next node and the state of the surrogate is already present and is ready to resume on the node that the end-device moves onto, proactive provisioning is effective at masking the state transfer time and reducing the downtime.

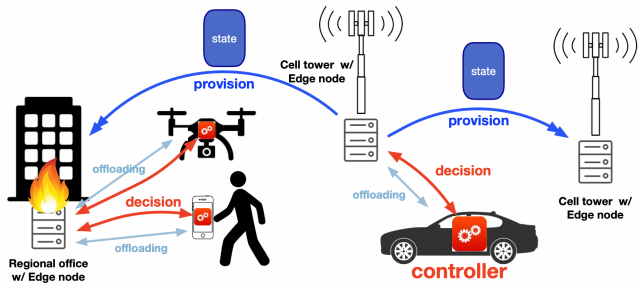

Figure 1: Interaction between various types of devices and edge-clouds located at roadside cellular towers and regional gateway office.

A proactive provisioning system must make predictions in many variables such as next edge node, total provisioning time, expected latency improvement, unlike previous just-in-time systems [5, 6, 12] that passively migrate on-demand when an end-device associates with a new edge node. With inaccurate prediction, proactive provisioning may consume extra resources for no performance improvement. For instance, when an end-device deviates from the predicted route and associates with an unprovisioned edge node, not only the application experiences high downtime to provision on-demand but resources previously used to provision (e.g., bandwidth, storage) are wasted. So previous edge provisioning systems either provision just-in-time [5, 6, 12] or target specific use-cases [7, 11, 13].

Worse, each prior system implements its own provisioning decision logic that targets specific performance metrics. Table 1 summarizes the control domains covered by each of them. We argue that the control plane should be decoupled from mechanisms to build a pragrmatic and flexible edge provisioning system. Furthermore, to balance performance improvement and resource usage, a system requires an intelligent provisioning *controller* that systematically incorporates various dimensions (i.e., where, when, and how) at run-time to make the provisioning decision proactively. Figure 1 shows the control plane decoupled from existing mechanism planes.

Prediction of the future 'nearby' edge nodes: A primary goal for migrating the surrogate to a nearby edge node is to retain low latency by alleviating network tail latency and jitter. When devices move to a new location, we should make a provisioning decision. For applications hosted on devices that often stay with

[1] A surrogate is a virtual machine (VM) or container that executes an application's offloaded computation on behalf.

MobiSys '19, June 17–21, 2019, Seoul, Republic of Korea

ACM ISBN 978-1-4503-6661-8/19/06.
https://doi.org/10.1145/3307334.3328649

System	Target Performance Metric	Target Use-cases	Controls			
			where	when	whom	how
JIT [6]	Total provisioning time	International traveler stepping off his flight	✗	✗	✗	✓
VM Handoff [5]	Total migration time	Unexpected flash crowd may overload small cloudlet	✗	✗	✗	✓
Service Handoff [12]	Total transfer time	Container migration with thin top writable layer	✗	✗	✗	✓
Steel [13]	Operation cost	Hybrid cloud to minimize monetrary cost	✓	✗	✓	✗
Paradrop [11]	Resource utilization	IoT devices that require computation power	✓	✗	✓	✗

Table 1: Comparison of prior edge provisioning systems with their control domain.

an edge node for a long period of time, the best strategy is always migrating to the nearest edge node since latency improvement is larger than the cost of migrating. For applications hosted on fast-moving devices, however, this strategy can be very costly since these devices rapidly move in and out of edge nodes' coverage and often associate with an edge node for a short period of time. Thus, for these devices, selectively provisioning edge nodes, which can improve application-perceiving latency is crucial to avoid resource waste.

Intuitively, proactive provisioning requires high prediction accuracy for the future location of devices to find which edge node the devices will encounter in the future. While the context-based approach [14] for predicting next edge nodes located in cellular towers offers a good accuracy, it requires building a specific prediction model for each device and use-case because a mobility pattern varies by types of a device and user's context. So, rather than considering the physical location of edge nodes, we plan to estimate end-to-end latency between current and predicted edge nodes located along the forecasted projectory that we will infer from previous and current location data at run-time. Then, we can selectively provision edge nodes that may cause high user-pereceiving latency when communicated over WAN. This strategy allows us to skip provisioning edge nodes that improve latency at the margin.

Making a prediction at run-time inherently comes at the uncertainty that may degrade application performance. So, we plan to explicitly incorporate uncertianity in our next edge node prediction. Rather than proactively provisioning one edge node at a time, we will compute the confidence for each candidate nodes and explicitly calculate expected latency gain and resource usage. Then, we can redundantly provision ones that have the expected benefit that surpasses the cost to proactively provision.

When to start provisioning: Given that applications in fast-moving devices are constantly used, starting to transfer the surrogate's state as early as application starts offers both low latency and low downtime. Yet, throughout provisioning stage, the surrogate at current edge node is actively used and the surrogate's state continues to change. So, it makes previously transferred state useless and requires re-transmitting the updated state until the end-device switches to a next provisioned edge node.

Instead, we will estimate the total provisioning time and predict the time left untill moving to a candidate edge node. In this way, we can avoid resource waste by starting to provision as late as possible so that the stage completes at the same time the end-device switches to the provisioned edge node. Figure 2 shows the potential benefit of this approach by starting to provision at different time periods ahead, when running AR Heads-Up-Display application [2] for 10 minutes. Two leftmost lines are employing proactive provisioning

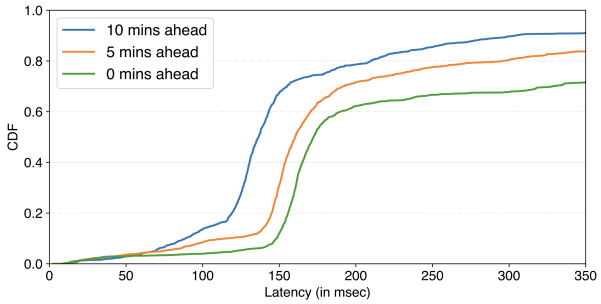

Figure 2: Latency CDF for starting proactive provisioning at three different time points ahead. Downtime is equal to the distribution after 350 ms.

and the rightmost line is a result of just-in-itme provisioning. While the leftmost line improves overall application-perceiving latency and reduces downtime the most, it also cosumes resources the most. On the other hand, the rightmost line results in high downtime but consumes minimal resources. Our initial design systematically explores the tradeoff between expected latency improvement and excessive resource usage by controlling where, when, and how dimension at run-time to support efficient proative provisioning on top of prior systems [4–6] as a controller module.

REFERENCES

[1] Intel drone light show breaks guinness world records title at olympic winter games pyeongchang 2018. https://tinyurl.com/y4xbwv2y.

[2] Porsche, Hyundai invest in WayRay to make augmented-reality HUDs. https://www.cnet.com/roadshow/news/wayray-augmented-reality-hud-investment.

[3] These police drones are watching you. https://www.pogo.org/analysis/2018/09/these-police-drones-are-watching-you/.

[4] K. Ha, Y. Abe, Z. Chen, W. Hu, B. Amos, P. Pillai, and M. Satyanarayanan. Adaptive VM handoff across cloudlets. Technical report CMU-cs-15-113, 2015.

[5] K. Ha, Y. Abe, T. Eiszler, Z. Chen, W. Hu, B. Amos, R. Upadhyaya, P. Pillai, and M. Satyanarayanan. You can teach elephants to dance: Agile vm handoff for edge computing. In *Proceedings of the 2nd IEEE/ACM Symposium on Edge Computing (SEC)*, 2017.

[6] K. Ha, P. Pillai, G. Lewis, S. Simanta, S. Clinch, N. Davies, and M. Satyanarayanan. The impact of mobile multimedia applications on data center consolidation. In *Proceedings of IEEE International Conference on Cloud Engineering (IC2E)*, pages 166–176, 2013.

[7] M. Jang, H. Lee, K. Schwan, and K. Bhardwaj. SOUL: an edge-cloud system for mobile applications in a sensor-rich world. In *Proceedings of the 1st IEEE/ACM Symposium on Edge Computing (SEC)*, 2016.

[8] H. Lee and J. Flinn. Poster abstract: Reducing tail response time of vehicular applications. In *Proceedings of the 1st IEEE/ACM Symposium on Edge Computing (SEC)*, 2016.

[9] H. Lee and J. Flinn. Poster: Redundancy aided vehicular networking. In *Proceedings of the 15th International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2017.

[10] H. Lee, J. Flinn, and B. Tonshal. RAVEN: Improving interactive latency for the connected car. In *Proceedings of the 24th International Conference on Mobile Computing and Networking (MobiCom)*, 2018.

[11] P. Liu, D. Willis, and S. Banerjee. Paradrop: Enabling lightweight multi-tenancy at the network's extreme edge. In *Proceedings of the 1st IEEE/ACM Symposium on Edge Computing (SEC)*, 2016.

[12] L. Ma, S. Yi, and Q. Li. Efficient service handoff across edge servers via docker container migration. In *Proceedings of the 2nd IEEE/ACM Symposium on Edge Computing (SEC)*, 2017.

[13] S. A. Noghabi, J. Kolb, P. Bodik, and E. Cuervo. Steel: Simplified development and deployment of edge-cloud applications. In *Proceedings of the 10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2018.

[14] J. Paek, K.-H. Kim, J. P. Singh, and R. Govindan. Energy-efficient positioning for smartphones using Cell-ID sequence matching. In *Proceedings of the 9th International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2011.