The Sky is the Limit: Cloud-Assisted Autonomous Driving via Service Tiers

Alexander Krentsel Peter Schafhalter Joseph E Gonzalez
Sylvia Ratnasamy Scott Shenker Ion Stoica

1 INTRODUCTION

Autonomous driving is a transformative AI application with the potential to save lives by eliminating human error from driving [1], provide mobility to millions impacted by disabilities [7], and increase economic productivity by improving traffic flow [13]. While industry has made remarkable progress towards deploying autonomous vehicles (AVs) [2], key challenges remain in ensuring high performance and coordination of the components comprising the autonomous driving system.

AV driving systems are typically structured as a pipeline [12, 18] containing the following components [4, 10], as shown in Fig. 1:

- *Perception* runs machine learning models to extract information from the sensor readings generated by cameras and LiDARs.
- *Localization* uses GPS and high-definition mapping to compute the vehicle's precise location.
- *Prediction* forecasts how nearby objects (e.g., other vehicles, pedestrians) will behave and move.
- *Planning* generates a safe and comfortable motion plan for the vehicle to take.
- *Control* converts a motion plan into steering, acceleration, and braking commands.

These components are made up of modules implemented with machine learning as well as traditional algorithms, and must coordinate to perform safe and comfortable driving maneuvers. This design describes a compound AI system [15], and has evolved this way due to the close interaction with human agents and regulation which requires them to be highly explainable and debug-able [3, 10].

Unlike compound AI systems for language models, which operate at scale and target throughput and statistical Service-Level Objectives (SLOs), AVs must meet a higher bar for reliability and performance and are latency-optimized. The strict target latency SLO decomposes into deadlines for individual components [9] that must be met for safe operation. For reliability, the system runs entirely on the car with no external dependencies, though they typically have network access through cellular connections.

We observe that these modules, with explicit dependencies and target SLOs, have similarities to cloud-based microservice architectures, and investigate the implications of viewing components in AVs as services. This allows us to reason about components in terms of SLOs, configurations, and contracts which provides clear APIs and guarantees across components. This ensures a degree of minimum performance and enables modular development to accelerate improvements.

Taking implications of modularity to the extreme, we can imagine multiple variants of these services that provide different performance guarantees (e.g. model specialized for city vs. model specialized for cloud, low-latency models for quick decision-making). If we accept that AVs are a collection of services with many different beneficial configurations, we could optimize the overall pipeline by selecting the best possible configuration. Unfortunately, we are unable to fully exploit this capability today because AVs are constrained by fixed, scarce hardware resources [12]. To solve this, we innovate on the execution environment; in particular we propose that AVs can take advantage of cloud-based execution to use more powerful GPUs that enable more beneficial services, run models faster, and deploy updates more easily.

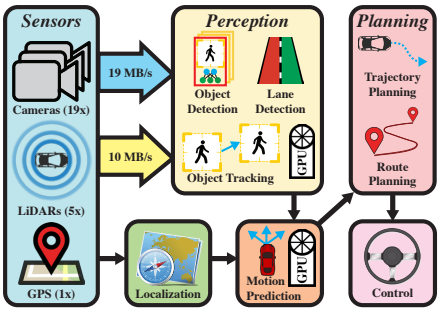


Figure 1: *A view of the components in an AV control system. One Object Detection module in the perception component is outlined in red.*

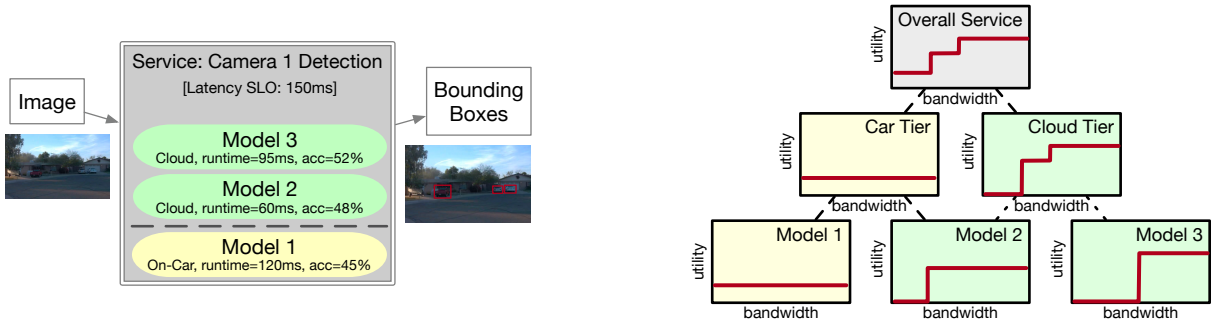
These decisions have far-reaching implications. First, cloud execution requires managing an additional constrained resource, network bandwidth, which is not plentiful enough to support remote execution of all services. Second, optimal per-service configurations may result in sub-optimal system-wide performance (i.e., overall driving system is stuck in a local minimum). Finally, cloud execution introduces new sources of unreliability that must be managed to maintain reliable overall performance. We propose a tiered service architecture which configures services by maximizing the benefit each service contributes to the overall AV while meeting the resource constraints.

2 A TIERED APPROACH

Several components or "services" in the AV pipeline have multiple possible configurations that provide the same functionality but with different tradeoffs in runtime, accuracy, compute requirements, and input fidelity. For example, the EfficientDet [16] family of object detectors solve the same object detection task but range in their input size, accuracy, and runtime across 8 model variants, ED0 through ED7[1]. We define a "configuration" to be the selected model variant used to satisfy the service. In Fig. 2a, we show this view for the Object Detector service shown in red in Fig. 1.

Existing AVs are designed with statically configured components. A single model variant is selected for each component for execution on pre-allocated resources to ensure a consistent and reliable execution environment. While prior work has explored re-configuring components to maximize safety [9], these possibilities are fundamentally limited by the fixed hardware available on the vehicle.

[1]This has also become the dominant paradigm with foundation model releases, multiple size models (e.g. 7B, 13B, 70B) that provide the same service [5, 8, 17].



(a) *The multiple model variants available for the detection service for the front camera. Each service would have its own view.*

(b) *The utility curves derived for each model, which compose upward to form per-tier utility curves and finally an overall service utility curve.*

Figure 2: *A view of a single tiered service, with the on-car tier in yellow and cloud tier in green. An AV system is then made up of dozens of such services.*

In contrast, viewing AVs as collections of services offers an opportunity to selectively expand the execution environment beyond the hardware available on a vehicle (e.g., using the cloud). The additional cloud "tier" of execution environment enables transparent access to powerful cloud resources for more compute at the cost of worse reliability and additional latency. Previous work has observed that typical AV models such as object detection can run 3-4x faster on cloud hardware as compared to AV hardware [14], indicating that the cloud can run more powerful models while offsetting network latency in order to meet latency SLOs. To guard against the unreliability of the network, we always execute a model on-vehicle, and greedily use more accurate results from the cloud if they arrive in time.

The challenge is that bandwidth to the cloud is limited and highly variable. As a result, the optimal decision of which services to execute in which tier, and which model to use in that tier, is a dynamic optimization problem depending on the available cellular uplink bandwidth at runtime and each service's expected accuracy gain. Executing a service remotely incurs variable additional network latency, but benefits from lower execution latency by running on cloud hardware. For the cloud tier, the total latency incurred is

$$l_{end-to-end} = t_{model\_runtime} + l_{RTT} + B * S_{input} \quad (1)$$

where l_{RTT} is ping round-trip time, B is bandwidth, and S_{input} is input size. Unlike allocating compute resources which are static on the car for AV control systems, *bandwidth* is a scarce resource that is highly dynamic; 5G deployments support bandwidths of up to hundreds of Mbps, but with high variance based on number of users on the network, proximity to tower, interference, etc. Evaluating public speedtest data from mobile devices, we find mobile device bandwidth in the US in 2024 to range from 5Mbps - 100 Mbps[2] with ping latencies of 15-75ms.

With this bandwidth range, we cannot afford to offload all services. A standard JPG used as input to the EfficientDetector family of models [16] ranges from ranges from 0.5Mb (ED0, 512x512) to 4.3Mb (ED7, 1536x1536). With an optimistic bandwidth of 100Mbps and ping of 20ms, and a service target latency SLO of 150ms, the network portion of uploading these JPGs adds 24ms (ED0) - 63ms (ED7) of additional latency, leaving 126ms - 87ms respectively for model runtime in the cloud if offloading just one service. Sharing that bandwidth amongst three off-loaded services would consume 32ms - 149ms. Therefore, we clearly cannot offload all 20+ detector services, let alone all services in the AV system, and likewise equally sharing bandwidth across all services would result in *no* individual service being able to use the cloud tier.

To solve this problem, we build on the idea of utility curves for bandwidth allocation introduced and used in past work [6, 11], which encode how much utility an application gets per unit of additional bandwidth. To adapt these to our use-case, we observe that each model must return results within the target latency, and has a fixed runtime and input size. The runtime, input size, target latency together can be used to solve Eq. (1) for a minimum bandwidth allocation required to achieve the target latency. Thus the utility of this model is 0 below this threshold, and some positive value u_i above this threshold. We choose this u_i to be the accuracy of the model. This can optionally be weighted by some set of constants α_i across all models and services. We show this in Fig. 2b.

These utility curves can be composed upwards as discussed in [11] to obtain a utility curve for each tier, for the service, and for the system overall. We then use the utility curves to allocate bandwidth in a max-min fair way [6] across the services, except we modify the final allocation step within each service, choosing to give all that service's bandwidth to the single best model that will get utility from it, rather than sharing at utility-ratios as is done in the original utility curve literature. This allows the overall system to achieve higher accuracy across services by prioritizing running those services on the cloud that benefit the overall system the most.

3 FUTURE DIRECTIONS

We see a number of open questions to follow this work. First, we observe an opportunity to co-optimize offload in cases where inputs are shared between multiple services. For example, if Service A takes Camera 1 and 2, and Service B takes Camera 2 and 3 as input, then allocating more bandwidth to Service A's inputs makes it "cheaper" to upgrade Service B. This should be reflected in the derived utility curves. Second, understanding how improvements to intermediate service quality affects overall vehicle safety is an open problem and may influence the optimal composition of utility curves. Finally, cars sharing the same cell tower may benefit unevenly from additional bandwidth, and intelligently allocating this bandwidth among vehicles can maximize fleet-wide safety.

[2]This value is configurable to be considerably higher; 4G/5G networks today offer 5-10x higher (up to 500Mbps+) downlink bandwidth than upload due to usage patterns. The underlying spectrum could easily be reallocated to favor higher upload bandwidth.

REFERENCES

[1] Automated Vehicles for Safety. https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety.

[2] Scaling waymo one safely across four cities this year. https://waymo.com/blog/2024/03/scaling-waymo-one-safely-across-four-cities-this-year/, March 2024.

[3] National Highway Traffic Safety Administration. Collision between vehicle controlled by developmental automated driving system and pedestrian. https://www.ntsb.gov/investigations/accidentreports/reports/har1903.pdf, mar 2018.

[4] Autoware. Autoware User's Manual - Document Version 1.1. https://tinyurl.com/2v2jkk9n.

[5] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are Few-Shot learners. May 2020.

[6] Zhiruo Cao and E W Zegura. Utility max-min: an application-oriented bandwidth allocation scheme. In *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, volume 2, pages 793–801 vol.2. IEEE, 1999.

[7] Henry Claypool, Amitai Bin-Nun, and Jeffrey Gerlach. Self-Driving Cars: The Impact on People with Disabilities. *Newton, MA: Ruderman Family Foundation*, 2017.

[8] Gemini Team. Gemini: A family of highly capable multimodal models. December 2023.

[9] Ionel Gog, Sukrit Kalra, Peter Schafhalter, Joseph E Gonzalez, and Ion Stoica. D3: A Dynamic Deadline-Driven approach for Building Autonomous Vehicles. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 453–471, 2022.

[10] Ionel Gog, Sukrit Kalra, Peter Schafhalter, Matthew A. Wright, Joseph E. Gonzalez, and Ion Stoica. Pylot: A Modular Platform for Exploring Latency-Accuracy Tradeoffs in Autonomous Vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 8806–8813. IEEE, 2021.

[11] Alok Kumar, Sushant Jain, Uday Naik, Anand Raghuraman, Nikhil Kasinadhuni, Enrique Cauich Zermeno, C Stephen Gunn, Jing Ai, Björn Carlin, Mihai Amarandei-Stavila, Mathieu Robin, Aspi Siganporia, Stephen Stuart, and Amin Vahdat. BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 1–14, New York, NY, USA, August 2015. Association for Computing Machinery.

[12] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E. Haque, Lingjia Tang, and Jason Mars. The Architectural Implications of Autonomous Driving: Constraints and Acceleration. In *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 751–766, 2018.

[13] National Highway Traffic Safety Administration. Traffic Safety Facts (2017 Data). https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812687.

[14] Peter Schafhalter, Sukrit Kalra, Le Xu, Joseph E. Gonzalez, and Ion Stoica. Leveraging cloud computing to make autonomous vehicles safer, 2023.

[15] Daniel Seita and Matei Zaharia, Omar Khattab, Lingjiao Chen, Jared Quincy Davis, Heather Miller, Chris Potts, James Zou, Michael Carbin, Jonathan Frankle, Naveen Rao, Ali Ghodsi. The shift from models to compound AI systems. https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/. Accessed: 2024-2-20.

[16] Mingxing Tan, Ruoming Pang, and Quoc V. Le. EfficientDet: Scalable and Efficient Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[17] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. February 2023.

[18] Nicolo Valigi. Lessons learned building a self-driving car on ROS. In *ROSCon Madrid 2018*, Mountain View, CA, September 2018. Open Robotics.