Configuring the OS for Tomorrow's Robots

Madhav Tummala, Daehyeok Kim, Joydeep Biswas and Aditya Akella
The University of Texas at Austin

I. Introduction

With the advent of autonomous mobile service robots (AMSRs), there is a growing interest in the simultaneous execution of diverse applications. However, configuring the OS in such a scenario presents a few challenges: (1) the optimal configuration for an application depends on the environment it is running in, (2) some configuration knobs are system-wide and cause conflicts, (3) configurations for applications can have complex and counter-intuitive effects on each other, and (4) the globally optimal configuration requires a well-defined relationship between the performance metrics of individual applications. In this paper, we elaborate on the above challenges with empirical results showing the interdependencies between configurations and discuss a potential solution for automating the search.

II. Experimental setup

We choose two applications: (1) Navigation application (NAV), which performs path planning and obstacle avoidance based on lidar scans, and (2) YOLO application (YOLO) [1], which does object detection from a raw image stream. For ease of exploration, we run them each in their own cgroups [1] to simulate our UT-Automata F1tenth Car [2]. To supply inputs to the respective topics we playback stored LIDAR recordings and video streams. We will particularly discuss four configuration knobs: (1) CPU shares of YOLO's cgroups, (2) memory soft limit of YOLO's cgroups, (3) real-time scheduling (deadline policy) for NAV process, (4) Transparent Huge Pages (THP) for the entire system.

For performance metrics, for NAV - we consider 99^{th} percentile of the processing time (collecting LIDAR scans + running path planning) as it is a time-sensitive application. For YOLO, we consider the mean frame processing time and add up both of them for the global performance metric.

III. Preliminary results

Table I shows some interesting configuration choices. Running the applications in a challenging environment like a hallway full of objects adversely affects performance metrics compared to our chosen baseline, a general hallway. Thus, different configurations may be required to maintain SLOs depending on the environment. Shifting the NAV process to the RT scheduler greatly improves its metric (distribution plot in Fig 1). We find that setting a period (for deadline policy) less than 50ms provides no additional improvement (as path planning is programmed to run every 50ms). On the other hand, reducing runtime and deadline parameters to 35ms and increasing CPU shares for YOLO improves YOLO's performance without sacrificing NAV's performance. Surprisingly, setting a soft memlimit (row 5) on YOLO enhances NAV's performance, demonstrating complex interactions. Enabling THP, a system-wide knob, slightly improves the performance of YOLO, but doesn't have much effect on NAV.

[1] A Linux kernel feature that helps control resources for a set of processes

TABLE I
NAV Loop Time and YOLO Frame Time

Configuration	99th PCTL Loop (ms)		YOLO Frame (ms)	
	Mean	Std Dev	Mean	Std Dev
Dense environment	133.490	18.738	2.640	0.022
Baseline	111.751	9.987	2.535	0.013
+NAV rt sched	36.161	0.438	2.658	0.025
+YOLO cpushares	36.560	0.468	2.420	0.011
+YOLO memlimit	35.833	0.527	2.554	0.020
+THP enabled	35.856	0.453	2.441	0.018

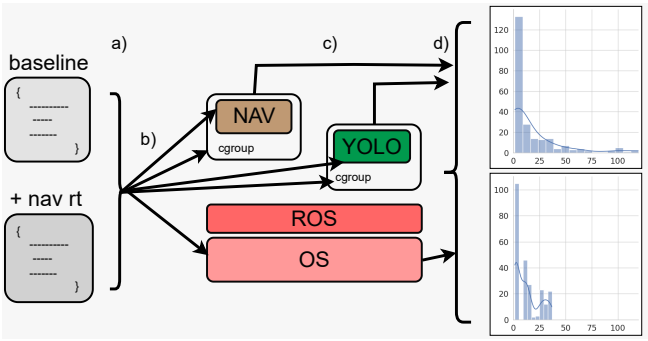

Fig. 1. (a) Configurations being applied. (b) Applied at different levels: process, cgroups, system-wide. (c) Collecting stats and traces for calculating performance metrics. (d) Frequency(y-axis) dist of the NAV loop time (ms)

IV. Future work

In future work, we plan to employ learning techniques to systematically find optimal OS configurations. For example, reinforcement learning appears well-suited for this purpose, given easy drop-in replacements for state changes (configuration as the state) and rewards (changes in performance metrics). Alternatively, the collected data from systematic exploration in this paper can be utilized for supervised learning. By capturing the concept of application signatures and input signatures, this approach can be applied to any mix of applications being deployed.

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.
[2] P. Atreya, H. Karnan, K. S. Sikand, X. Xiao, S. Rabiee, and J. Biswas, "High-speed accurate robot control using learned forward kinodynamics and non-linear least squares optimization," in *IROS 2022, Kyoto, Japan, October 23-27, 2022*. IEEE, 2022, pp. 11 789–11 795.