We thank the reviewers for their interest in the paper and their constructive feedback. In the following, we address the outstanding comments and outline our plan for improving the paper.

Comparison to Facebook's Horizon platform (Reviewer 3). Facebook's Horizon platform plays an orthogonal and complementary role to Park. In Figure 1, we show where Horizon and Park reside in a typical RL workflow diagram. As shown, Horizon implements several RL algorithms and exposes APIs to developers to integrate RL training and maintenance into their systems. By contrast, Park is a research platform that provides a common interface to connect to a wide range of computer system environments. In fact, Park's primary contribution is this environment suite that can serve as benchmarks for RL research in computer systems. In principle, a developer can use Horizon to try out existing RL algorithms or develop new RL algorithms for the system environments in Park. We will add this comparison to Horizon to the introduction (where we compare with OpenAI Gym) to better position the Park platform.

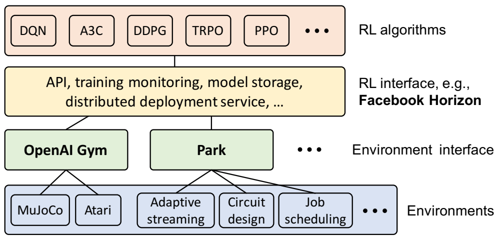

Figure 1: RL workflow. Park uses a common interface to connect to twelve computer system environments.

"Correctness" of system MDP (Reviewer 3). Thank you for raising this point. Formulating the MDP is an important, problem-specific step in applying RL to any system. In fact, some of Park's systems have multiple possible MDP formulations, which can affect the efficiency of RL training (e.g., for the Tensorflow device placement problem, an "incremental improvement" formulation (Addanki et al., 2019) is more efficient than the "one-shot" MDP (Mirhoseini et al., 2018)). While the main contribution of Park is not defining the MDP for each system environment, we thought carefully to pick an appropriate MDP formulation for each problem. Our guiding principle was to provide the RL agent with all the information and actions available to existing baselines schemes in that environment, such that the agent could at least express existing human-engineered policies. In most cases, the MDP formulations are straightforward and their correctness is self-explanatory. However, some are more subtle (e.g., the Spark scheduling and TF device placement), and in these cases we adopt the formulations from prior work. In the final version of the paper, we will discuss the MDP formulation for each system in more detail in the appendix, pointing out any subtleties with reference to prior work.

How extensible is Park's interface? (Reviewer 3). We will use lines of code added for each system to quantify how much work it takes to add new environments with Park's interface. For example, the adaptive video streaming environment uses 324 additional lines of code for Park interaction. To add a new environment, as we commented in §4, the developer only needs to specify the state and action space, and make them accessible through RPCs. We will add the additional lines of code for each environment as a reference in the appendix.

Details of baseline schemes (Reviewer 3). In our experiment, we compare against existing heuristics specifically designed for each environment. In the appendix, we describe the details for the compared heuristics, the overview of their implementations and the intuitions for how they work. Due to space limit, we only provide the names of these baseline schemes in Figure 4. We will add a forward pointer to the appendix in the experiment section for the details.

More detailed discussions on the experimental results (Reviewer 2 and 3). We agree with Reviewer 2 about trimming §4 (mostly the "Real System Interaction Loop" part) and adding more details from the appendix to §5. As suggested, we will highlight and comment on the results where the RL policies are not stable. We will also describe our model tuning strategy, a form of grid search, which is performed separately for each environment. We believe that the instability observed in some of the environments are due to the fundamental challenges discussed in the paper, not poorly tuned training procedure. For example, the policy in Figure 4(h) is unable to converge smoothly partially due to the variance caused by the cache arrival input sequence (i.e., the challenge in §3.2). Also, to clarify, the horizontal lines in Figure 4 correspond to fixed, non-learning baseline schemes in each system. These policies are not trained. We include their performance level as a constant line for reference in Figure 4.

Standardized interface for testing environments (Reviewer 1). In our experiments, we test the RL agent on unseen scenarios by loading a different set of traces. For example, the adaptive video streaming environment loads unseen network traces and the Tensorflow device placement environment tests with unseen models (i.e., unseen computation graphs). The users have the flexibility to create different scenarios to test the robustness of a learning model. However, we agree that standardizing the training/testing scenarios (e.g., fixing which traces to use for training/testing) would provide a better common ground for comparison. We will provide flags to launch default training and testing scenarios in our implementation.

Elaborate on future directions (Reviewer 1 and 3). In the conclusion section, we will highlight the future research direction by summarizing the challenges discussed in §3. In addition, the rightmost column of Table 2 outlined the challenges for each environment. We will make them stand out more when we discuss the table in §4.

Adding reference (Reviewer 2). We will add the citation for adaptive video streaming. Thanks for pointing this out!