CherryPick

Adaptively Unearthing the Best Cloud Configuration for Big Data Analytics

Paper authors: O. Alipourfard, H. Harry Liu, J. Chen, S. Venkataraman, M. Yu, M. Zhang

Venue: NSDI 2017

Presentation by Andreea Zaharia (az396) | R244 | 08/11/2021

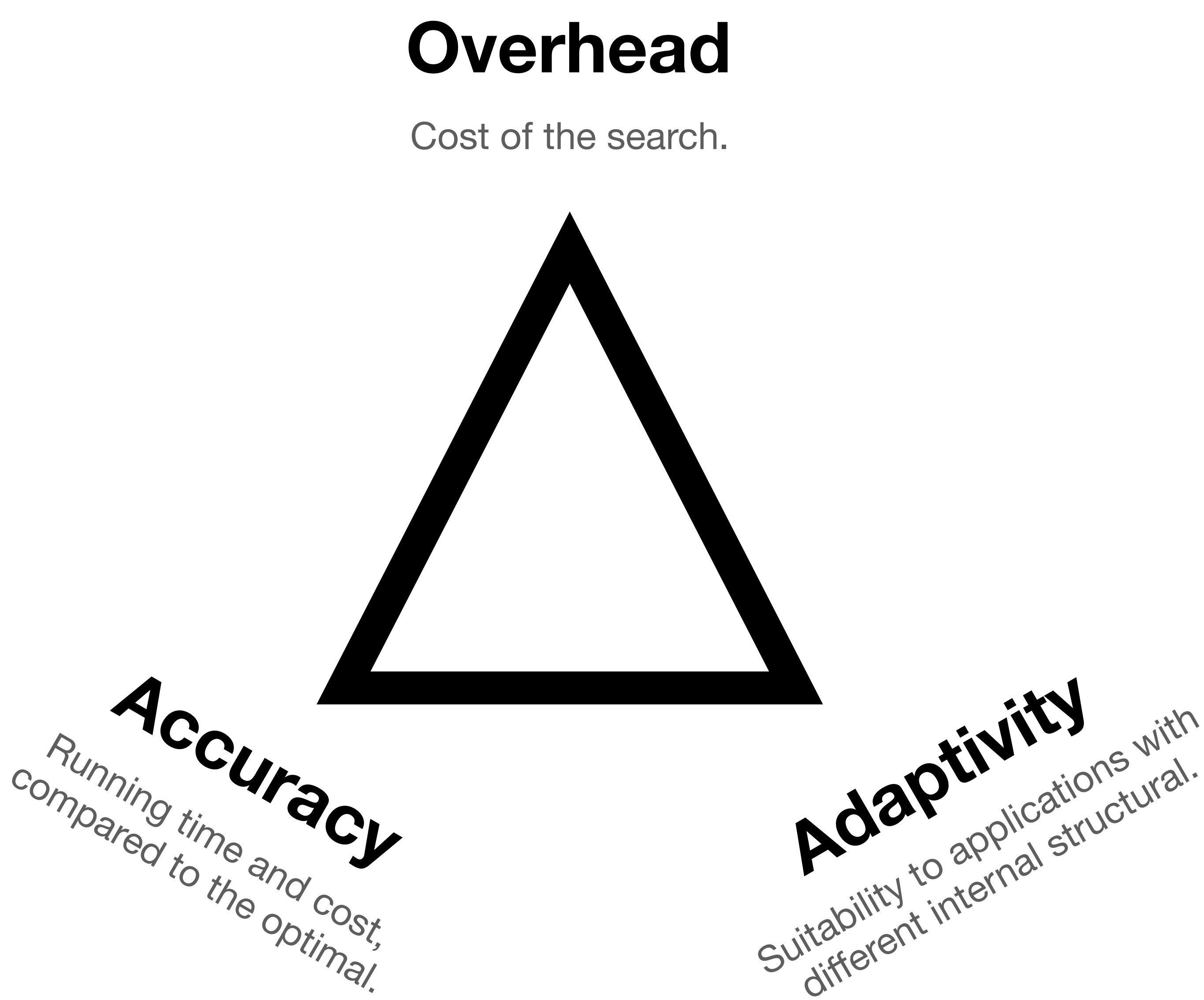
Background

Motivation

- **Choices:** cloud providers, machine types, cluster size.
- **Good config** —> saves time & space —> higher quality service.
- **Bad cloud config** —> up to 12x higher cost and 3x running time.
- **Complementary** to work on optimising application configs.
- **Recurring jobs** would benefit the most…
    - … and up to 40% of analytics jobs are recurring!

Background

Challenges and prior work



- Prior work failed to simultaneously solve all three challenges.
- Searching approaches: e.g. coordinate descent, random search.
- Modelling approaches: e.g. Ernest.

Design

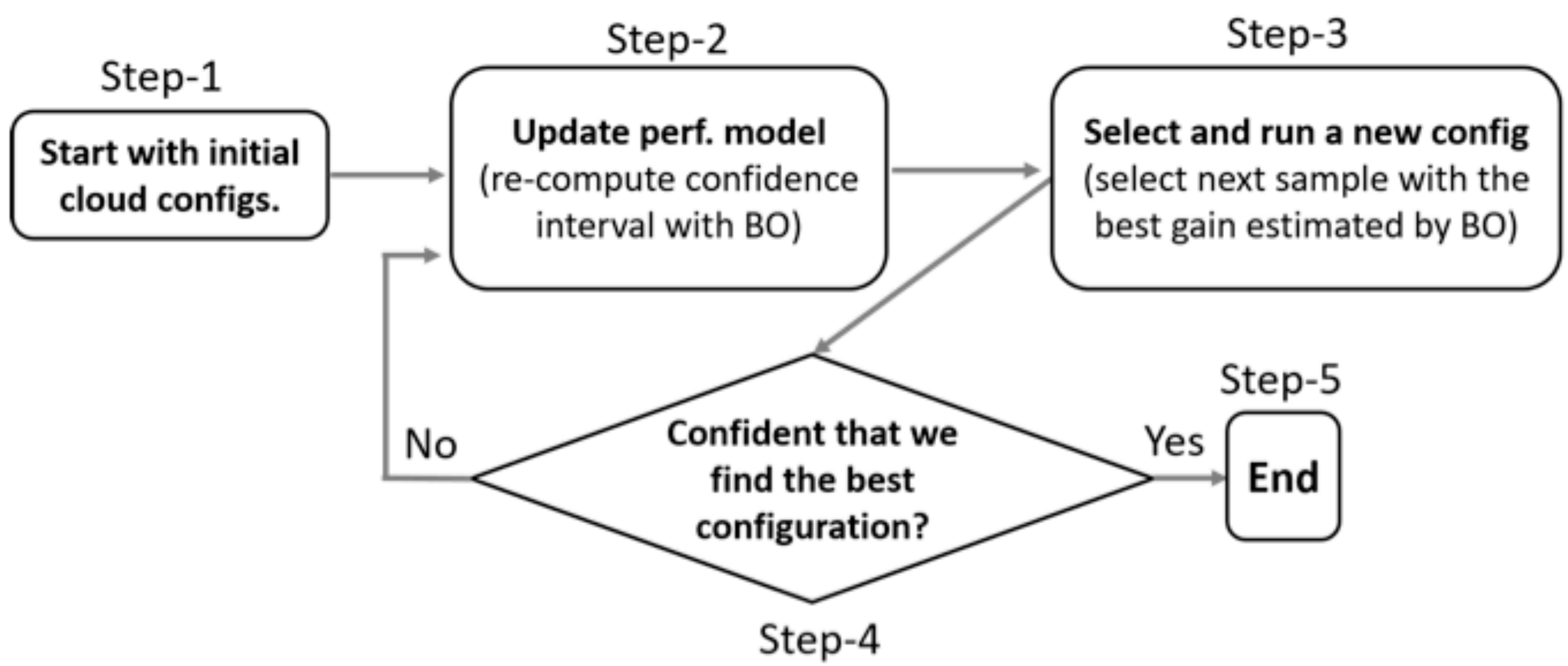
Key ideas

- **Cloud configuration:** number of VMs, CPU count & speed/core, RAM/core, disk count & speed, network cap of the VM.
- **Performance model:** accurate enough to distinguish the near-optimal configs from the rest.
- **Bayesian Optimisation:** for black-box functions; non-parametric

Design

Workflow

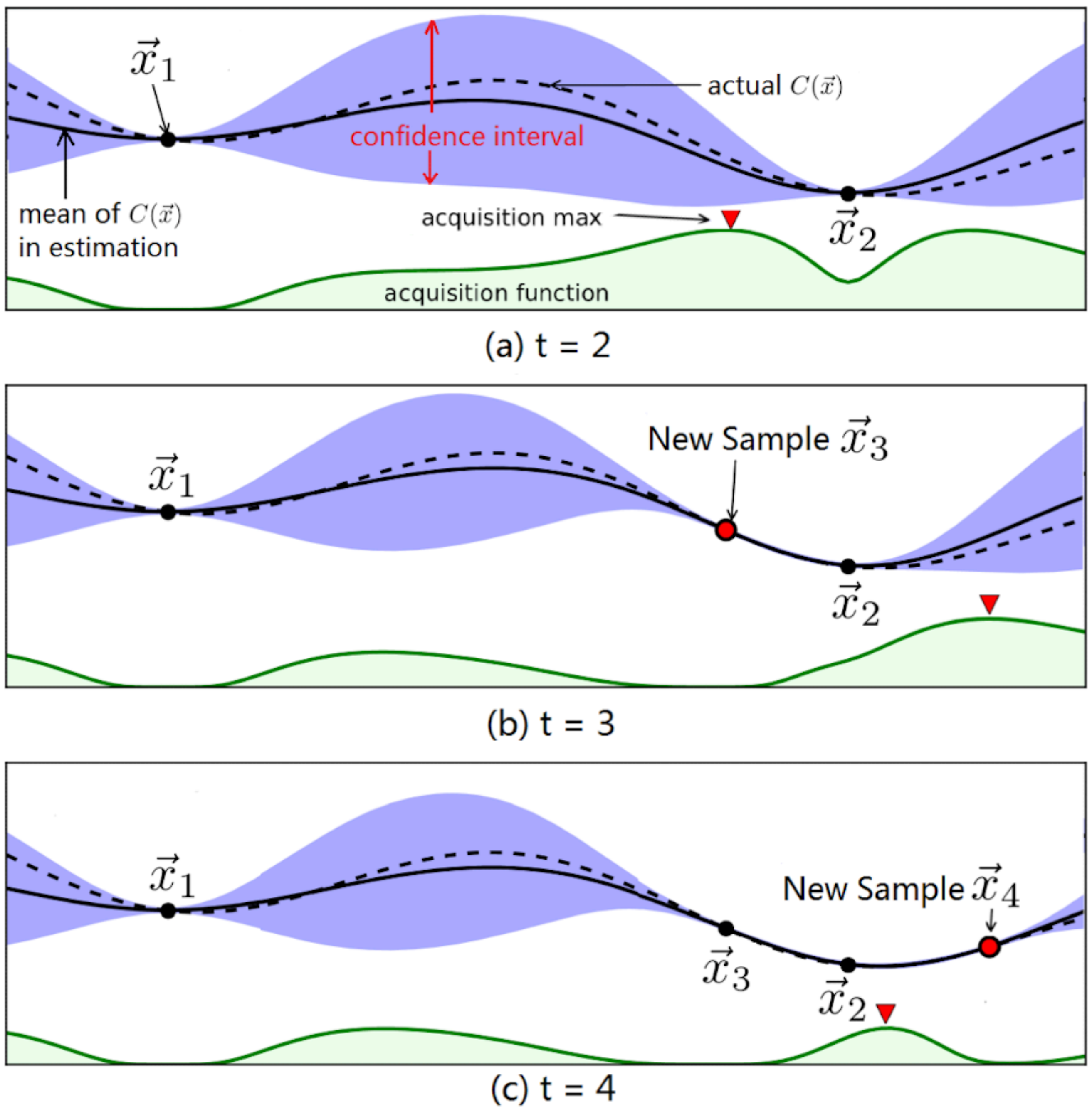
- Iterative and dynamic workflow:
  - Pick the next cloud config, by the performance model.
  - Run the config and update the model.



Design

Bayesian Optimisation

- **Prior**: models performance and cost of a config; GP.
- **Acquisition:** ranks and chooses the next config.
- **Posterior:** confidence interval of cost and runtime.



Design

Noise handling

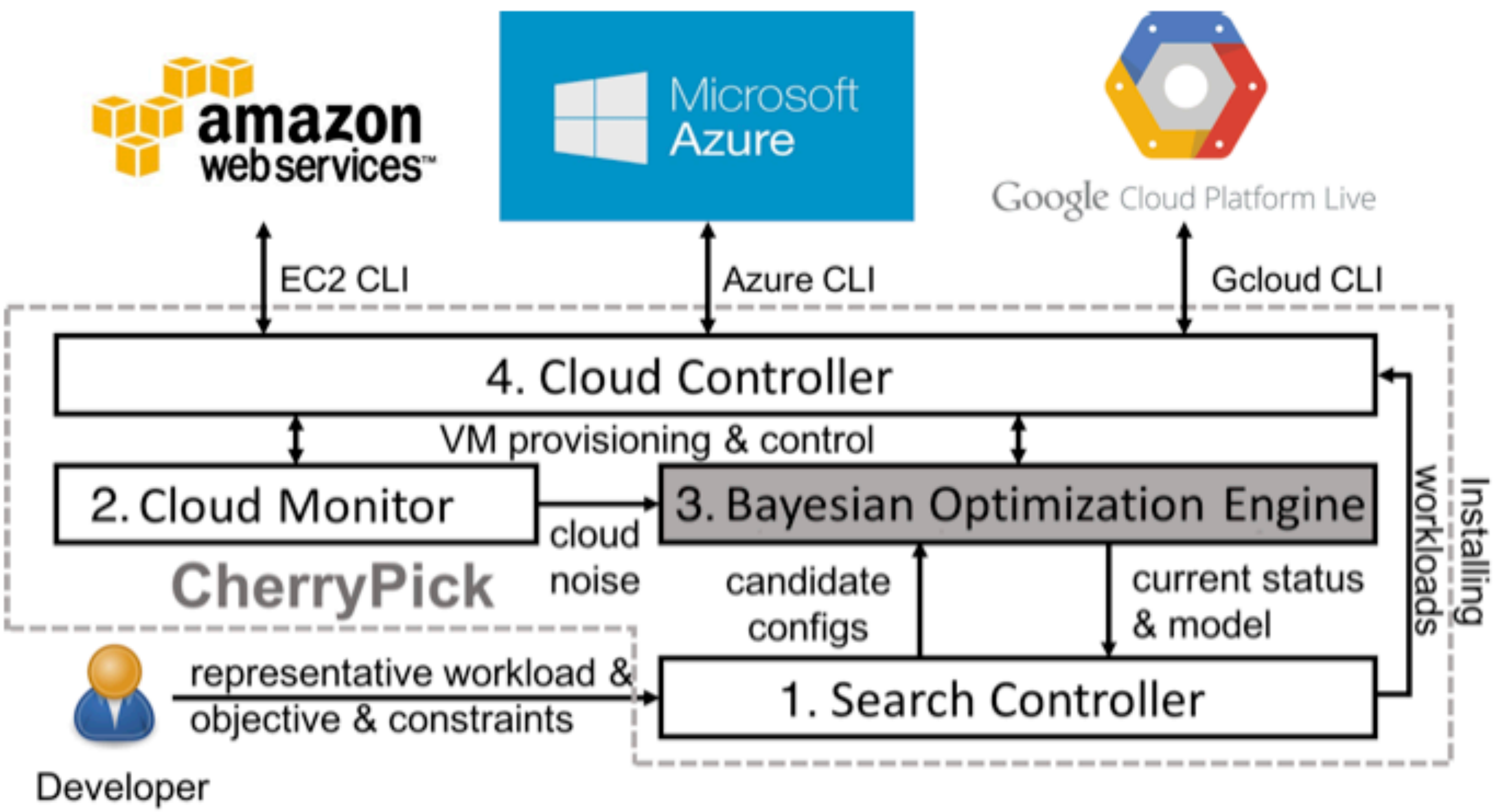
- BO is great at handling additive noise…
- … but noise in the cloud is multiplicative.

- Idea is to minimise the logarithm of the cost function instead:

$$\begin{aligned} \underset{\vec{x}}{\text{minimize}} \quad & \log C(\vec{x}) = \log P(\vec{x}) + \log T(\vec{x}) \\ \text{subject to} \quad & \log T(\vec{x}) \leq \log \mathscr{T}_{max} \end{aligned}$$

Implementation

Architecture



Evaluation

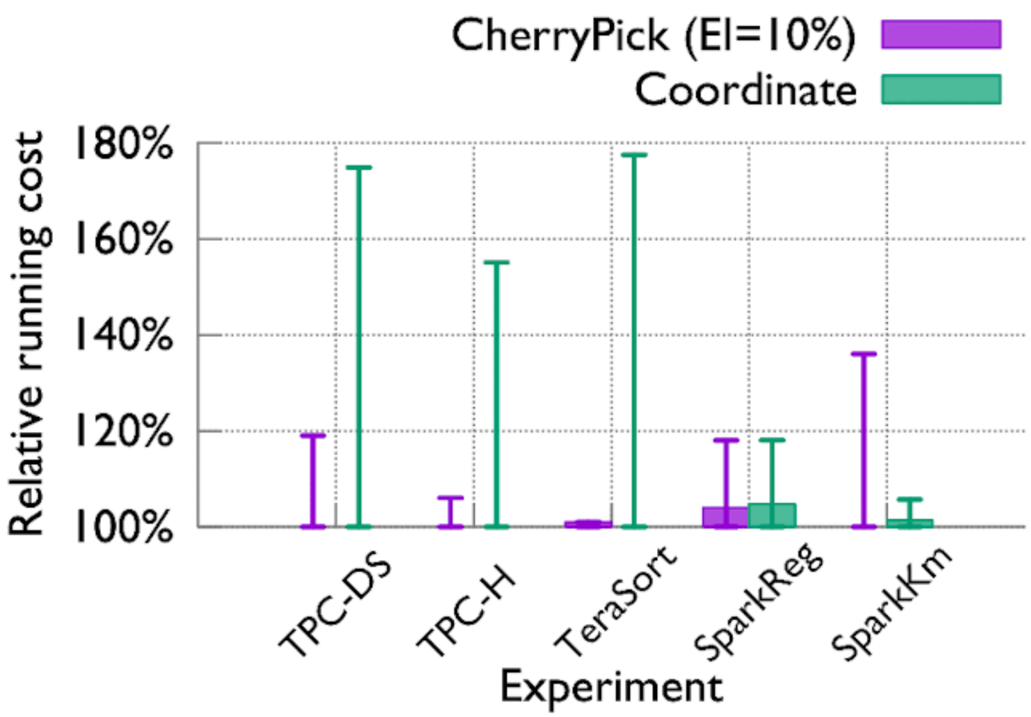
Experiment summary

- **Input:** five popular analytical jobs.
    - 66 reasonable configurations, of four families in Amazon EC2.
- **Objective:** minimise cost, under running time constraints.

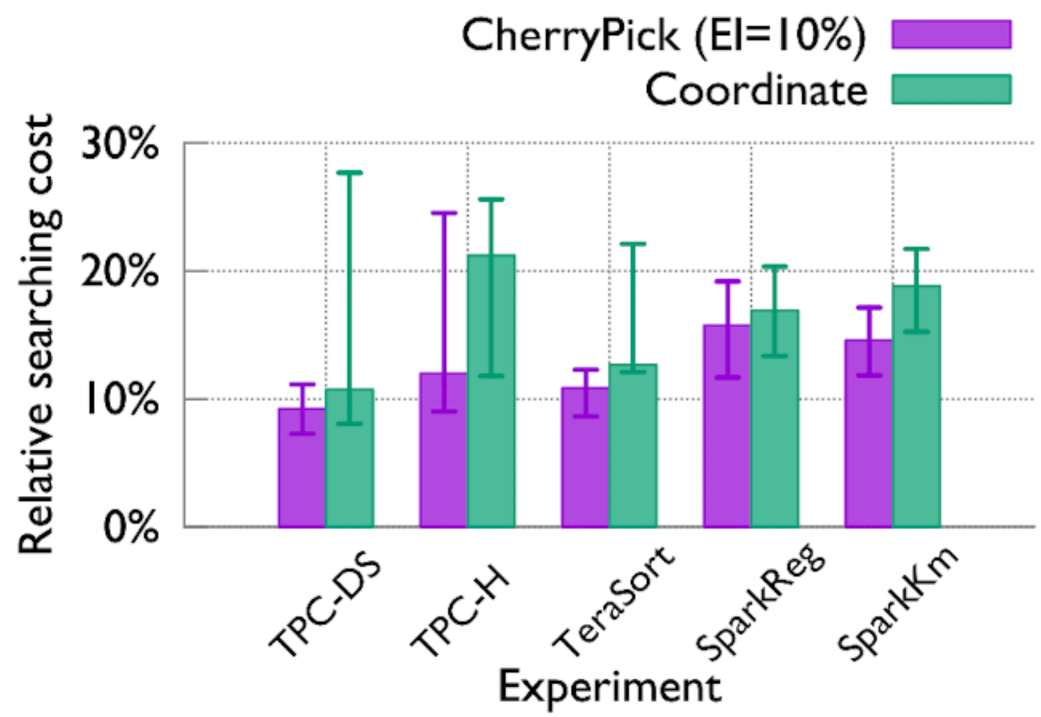
- **Results:**
    - 45-90% to pick optimal, otherwise finds a solution within 5%.
    - Alternatives take up 75% more time and 45% more overhead.

Evaluation

Experiment results



(a) Running cost



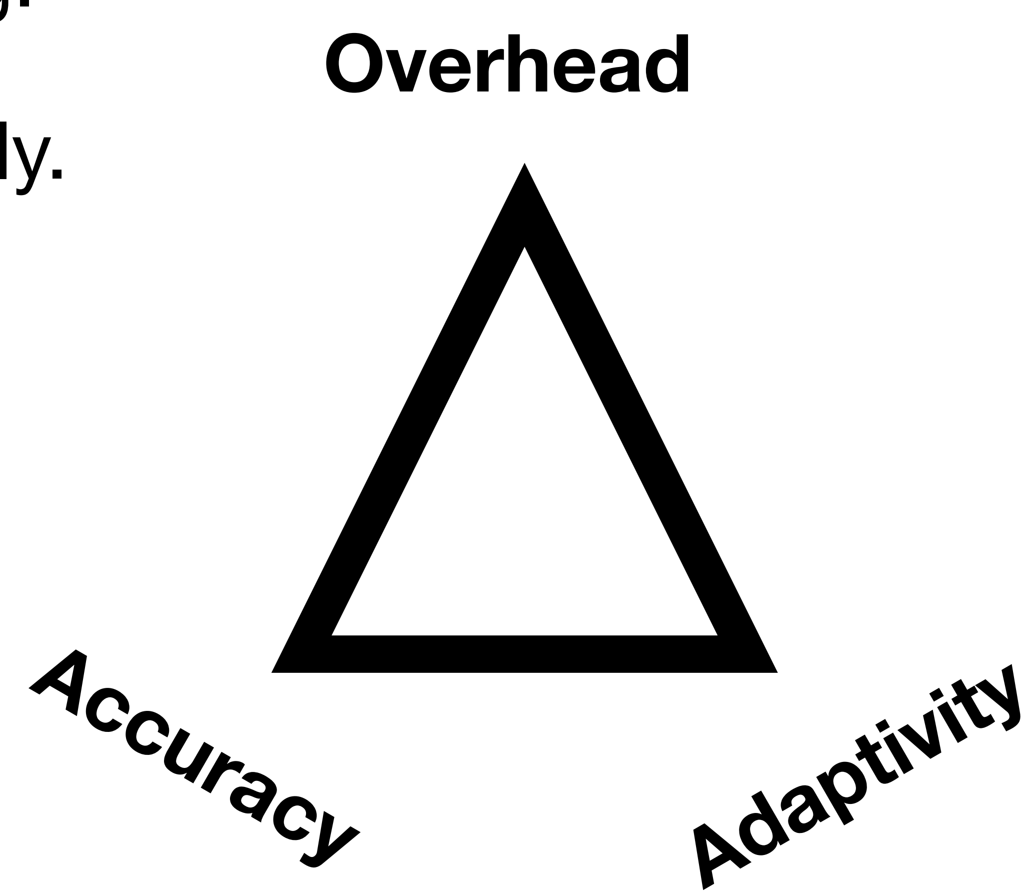
(b) Search cost

Figure 7: Comparing *CherryPick* with coordinate descent. The bars show 10th and 90th percentile.

Contribution

Differences to prior work and novelty points

- CherryPick achieves all three goals:
  - High accuracy: modelling only top ranking configs.
  - High adaptivity: black-box modelling.
  - Low overhead: searching interactively.



Other comments

Criticism

- “45-90% chance to find the optimal” — does not mean much…
- **Representative workloads** are needed for CherryPick to work.
  - Difficult to find. The paper brushes off this limitation.
- **The prior** is set to GP and cannot be modified by the user.
  - Disables improvements by application specific knowledge.
- Can it always converge to a near optimal solution?

Questions?