Alohamora: Reviving HTTP/2 Push and Preload by Adapting Policies On-the-Fly

Nikhil Kansal

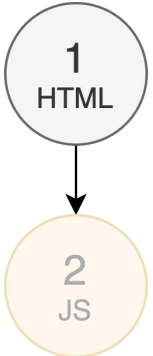
Murali Ramanujam

Ravi Netravali

UCLA

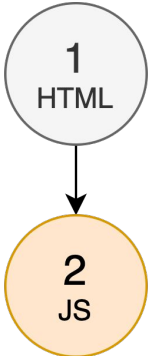
Serial dependency chains in page loads

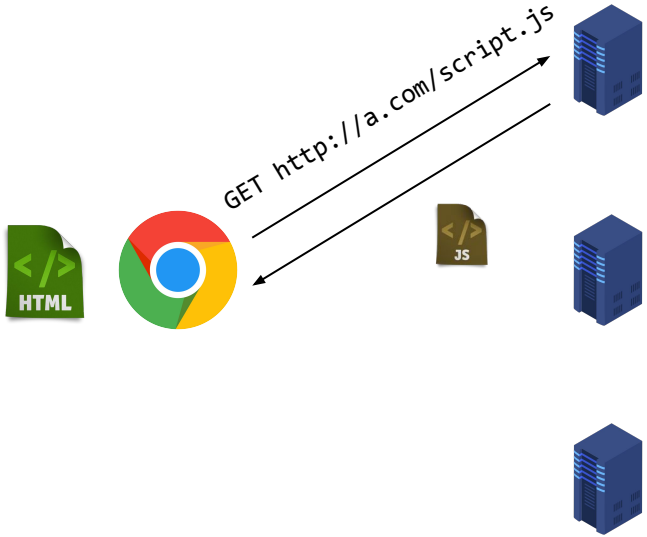
Parse page and discover dependencies



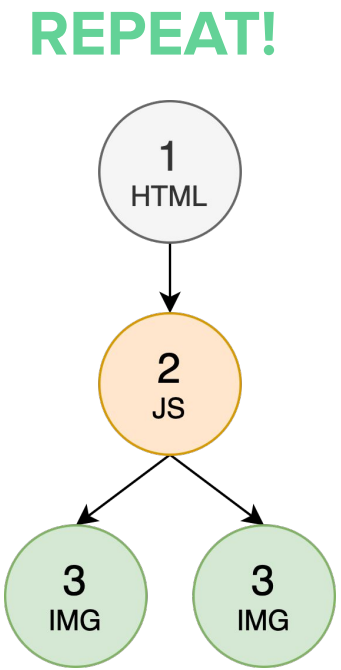
Serial dependency chains in page loads

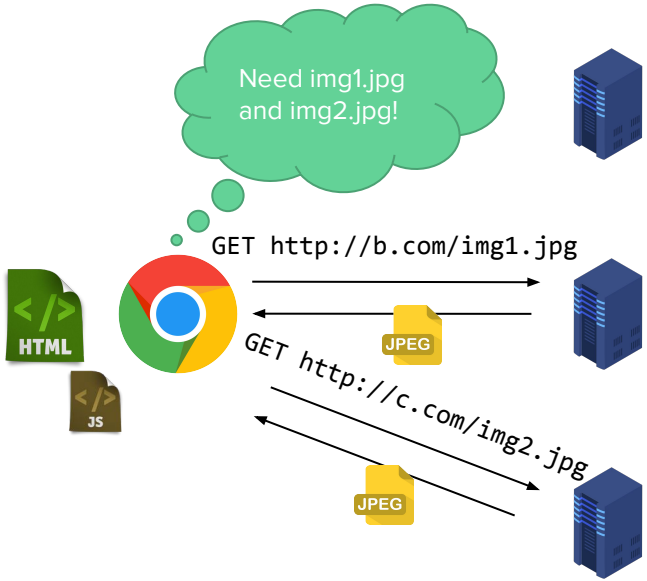
Load dependencies from remote server



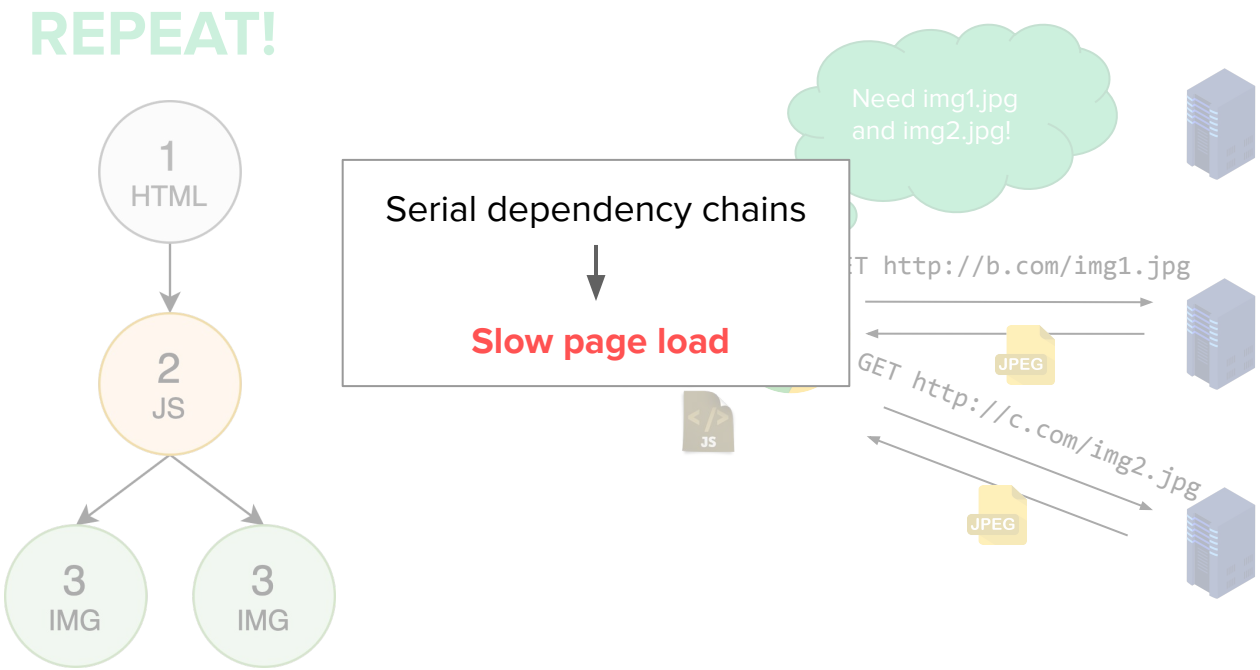


Serial dependency chains in page loads



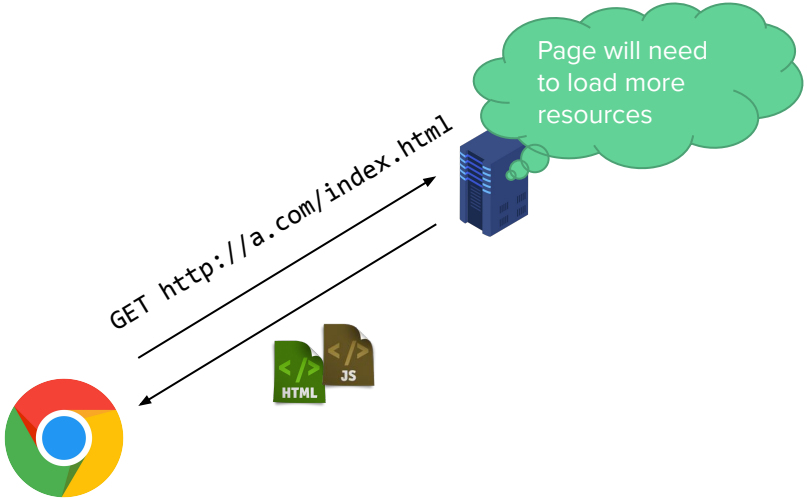


Serial dependency chains in page loads



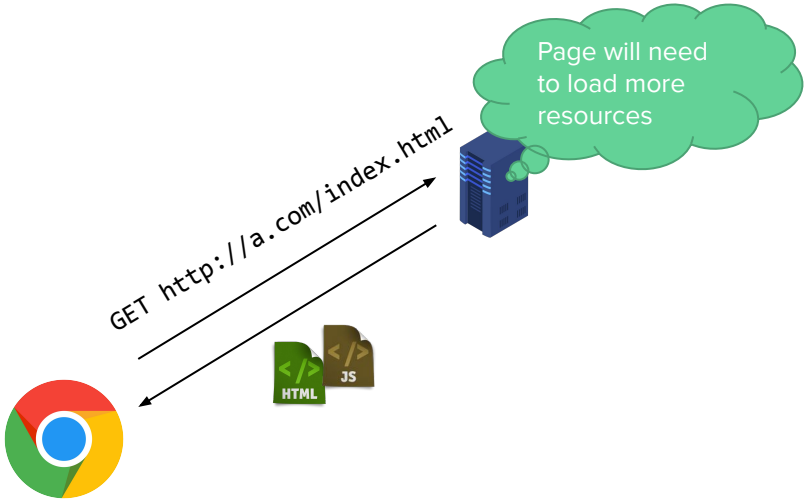
Page loads with HTTP/2 push + preload

Push

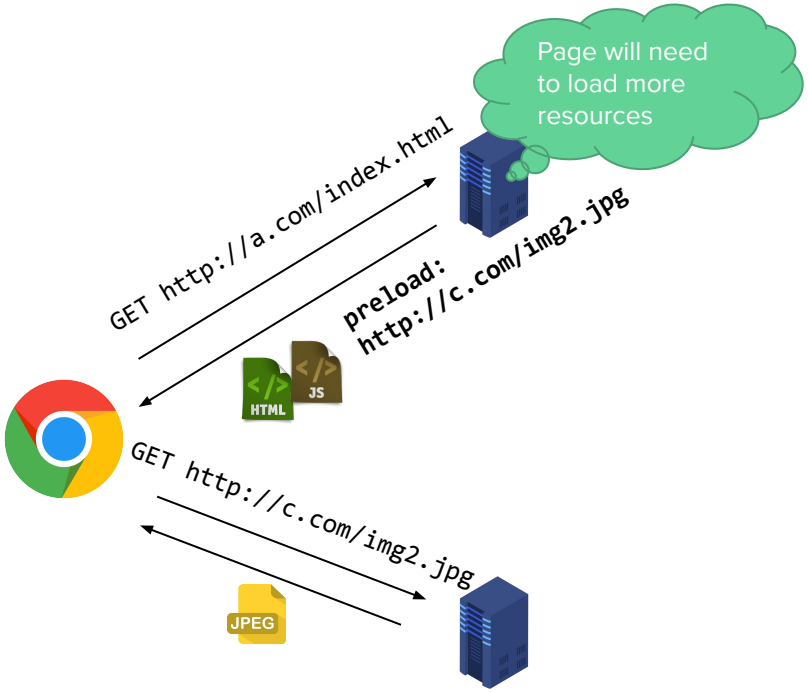


Page loads with HTTP/2 push + preload

Push

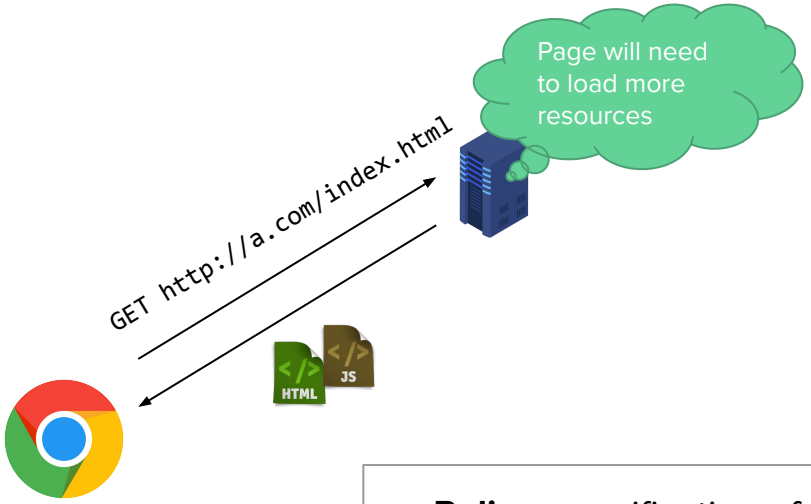


Push + Preload



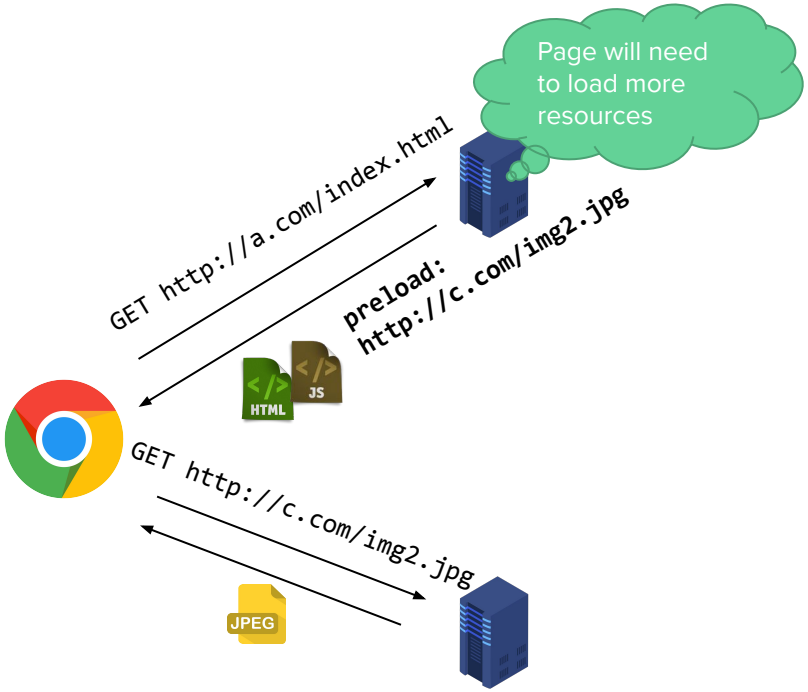
Page loads with HTTP/2 push + preload

Push



Policy: specification of which objects to push/preload from which parents

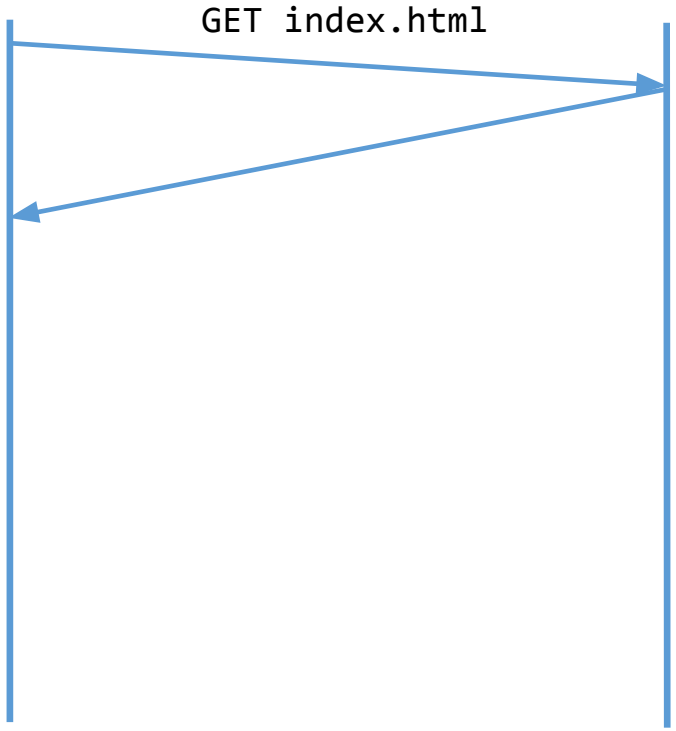
Push + Preload



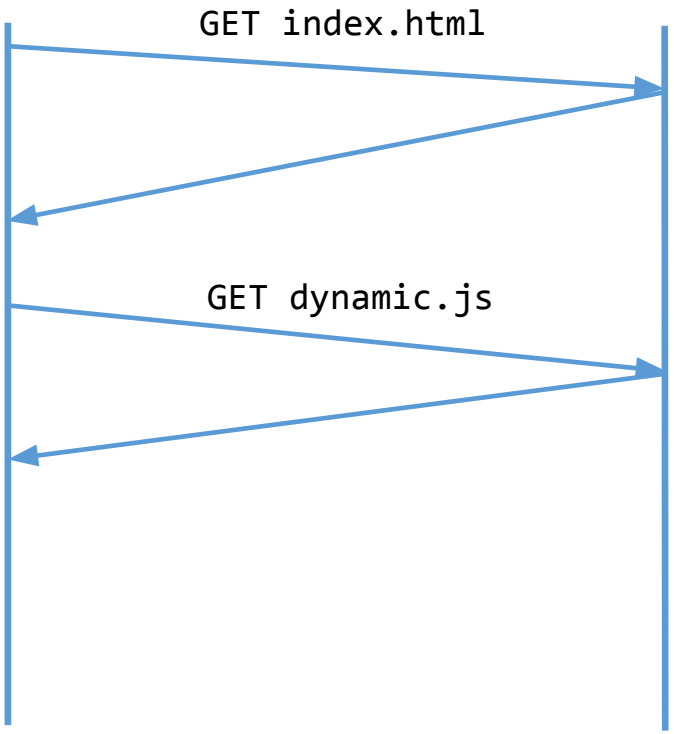
Static policies don't generalize

Static policies don't generalize

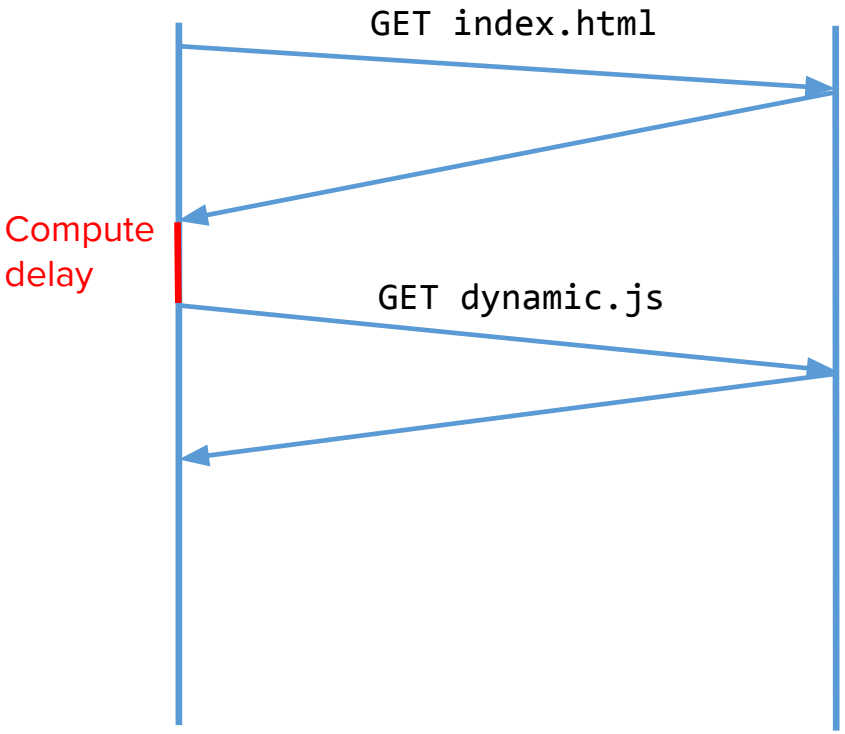
Static policies don't generalize



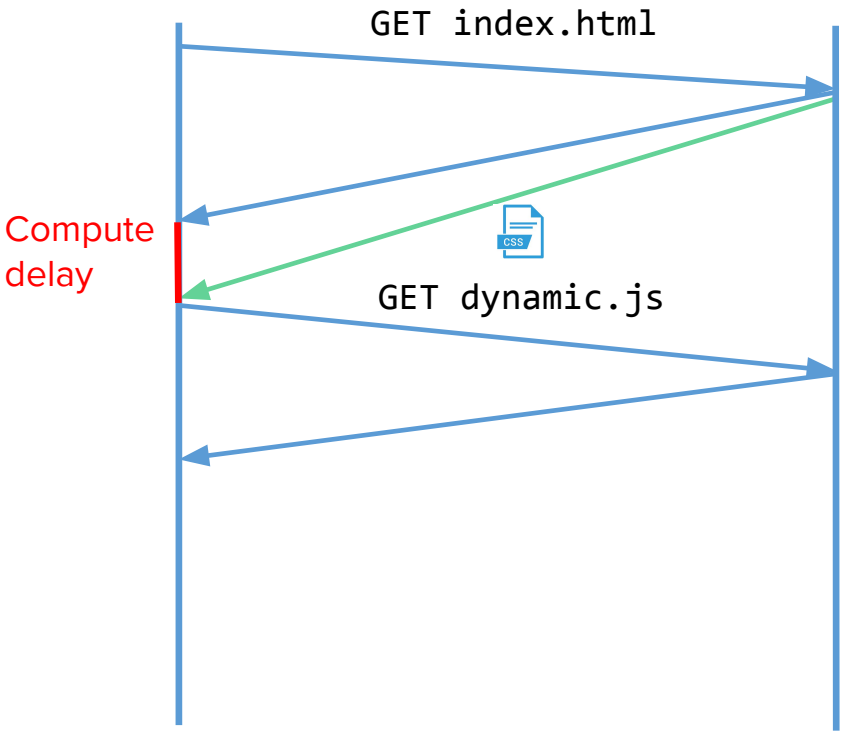
Static policies don't generalize



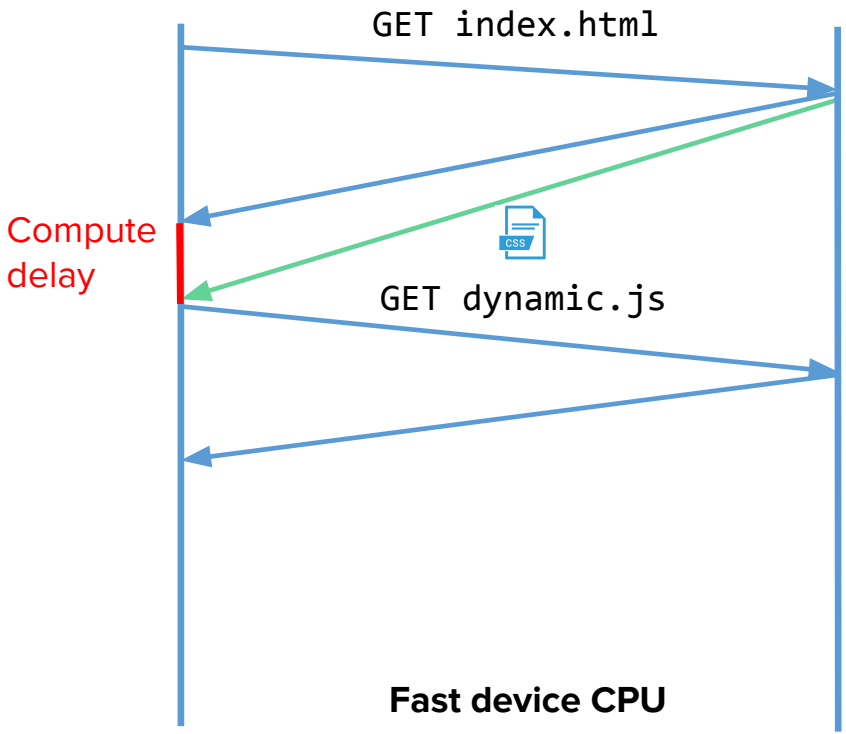
Static policies don't generalize

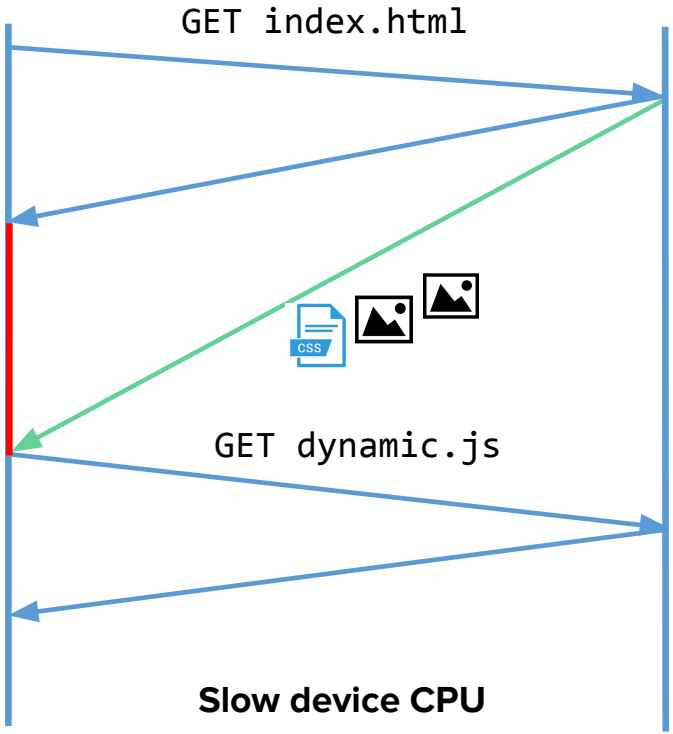


Static policies don't generalize



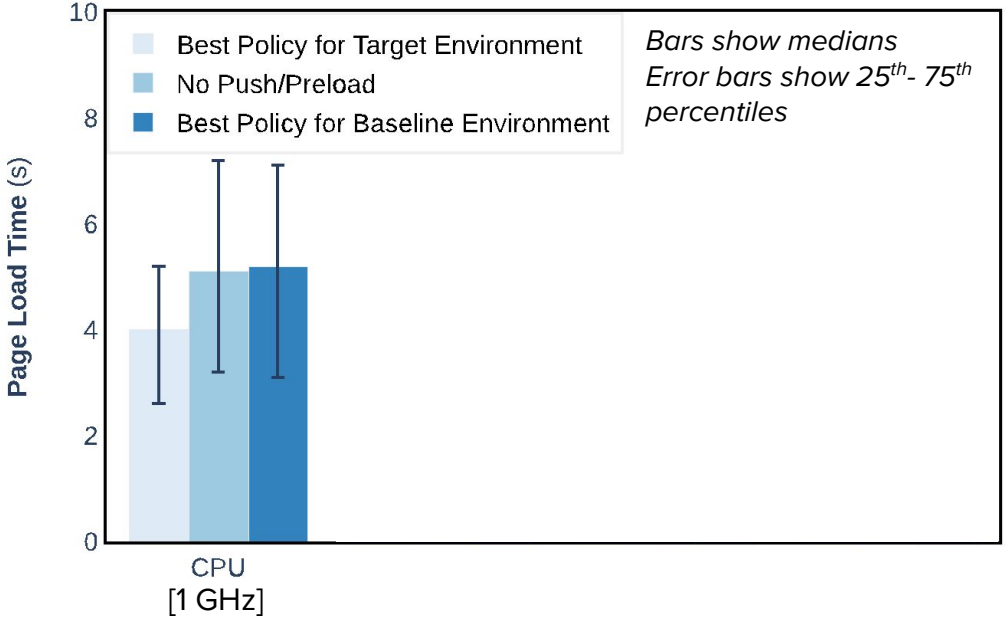
Static policies don't generalize





Static policies don't generalize

Found best policy for each environment and applied it to a different environment



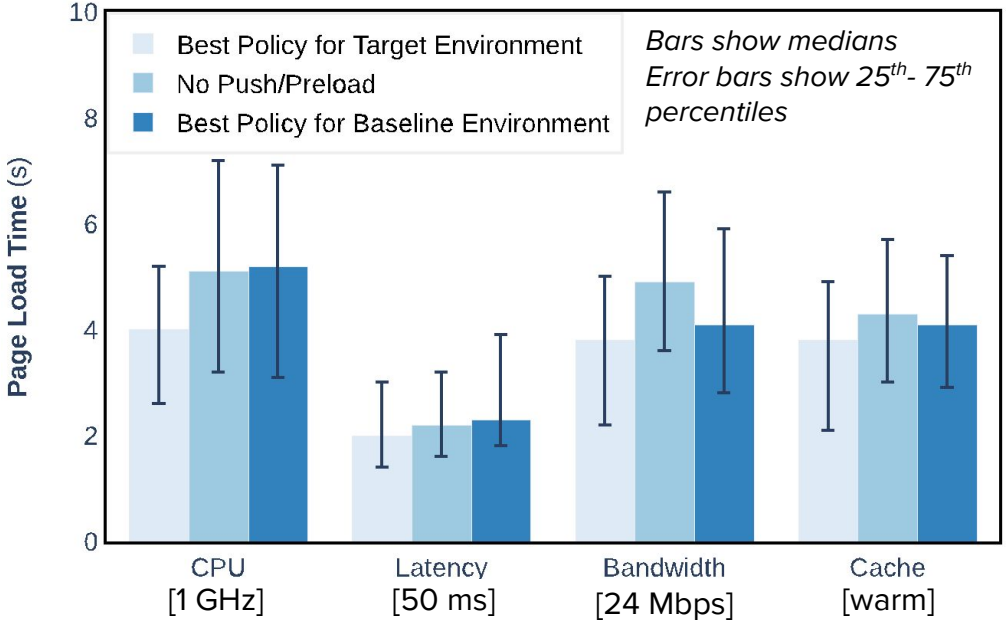
Baseline environment: [2.2 GHz CPU, 100 ms, 12 Mbps, cold cache]
Best Policy: brute force for that condition
No Push/Preload: default browser

Static policies don't generalize

Found best policy for each environment and applied it to a different environment

Policies from the wrong environment **forego 18-30%** of potential benefits

Performance can **degrade over 20%** compared to default load



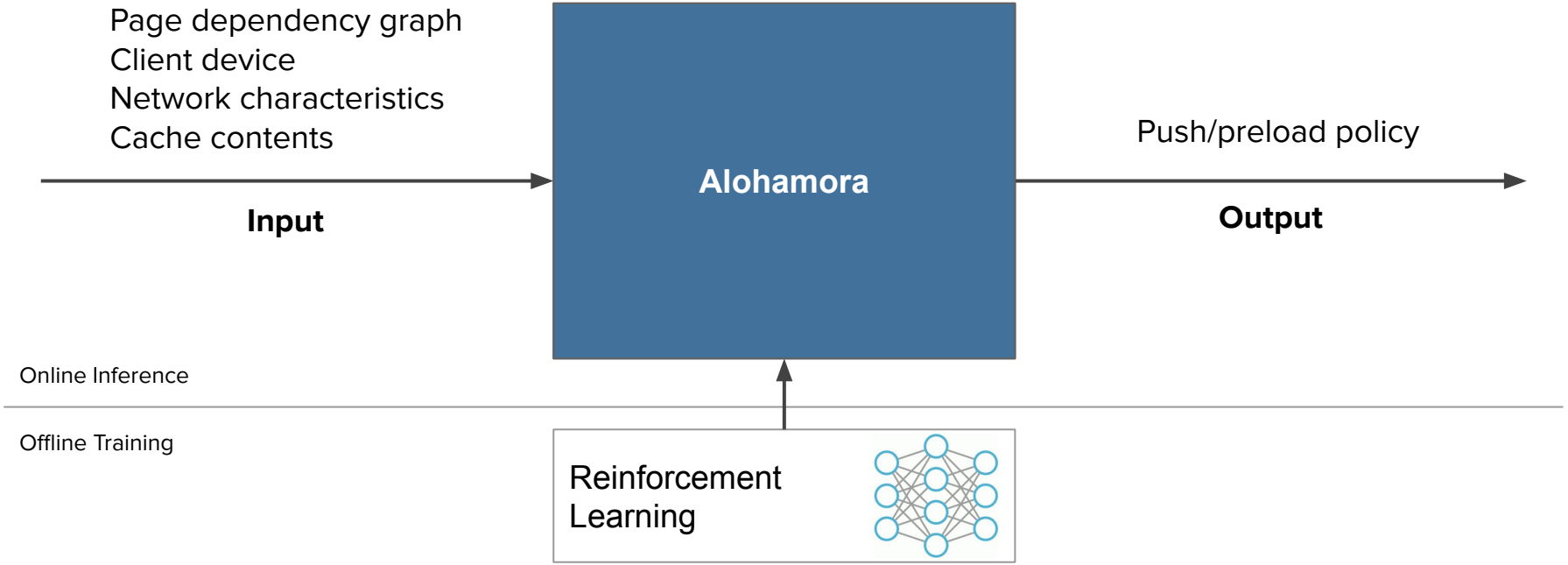
Baseline environment: [2.2 GHz CPU, 100 ms, 12 Mbps, cold cache]
Best Policy: brute force for that condition
No Push/Preload: default browser

Alohamora

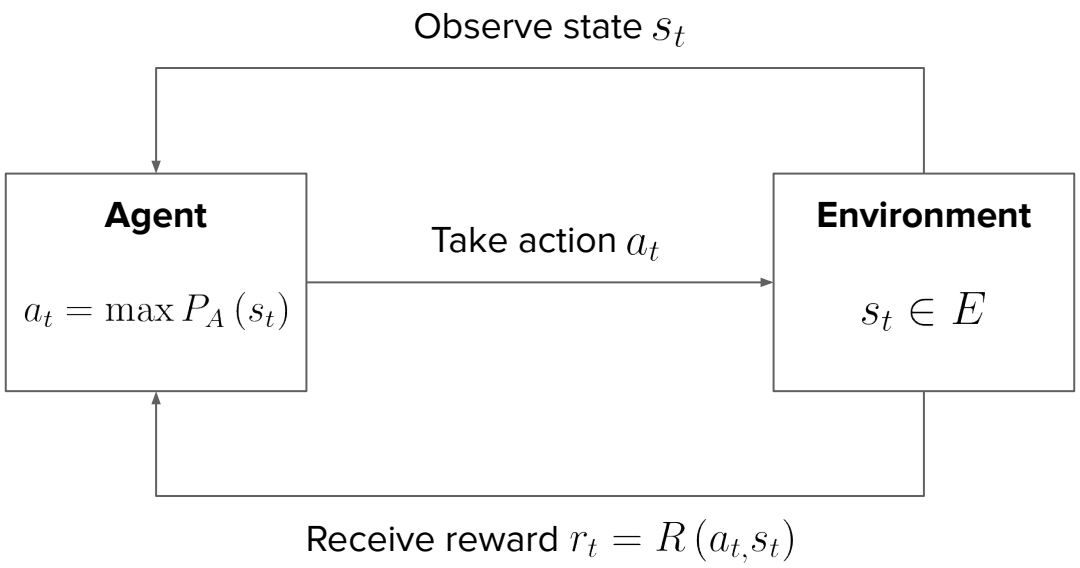
Generates dynamic push + preload policies on-the-fly

Alohamora

Generates dynamic push + preload policies on-the-fly

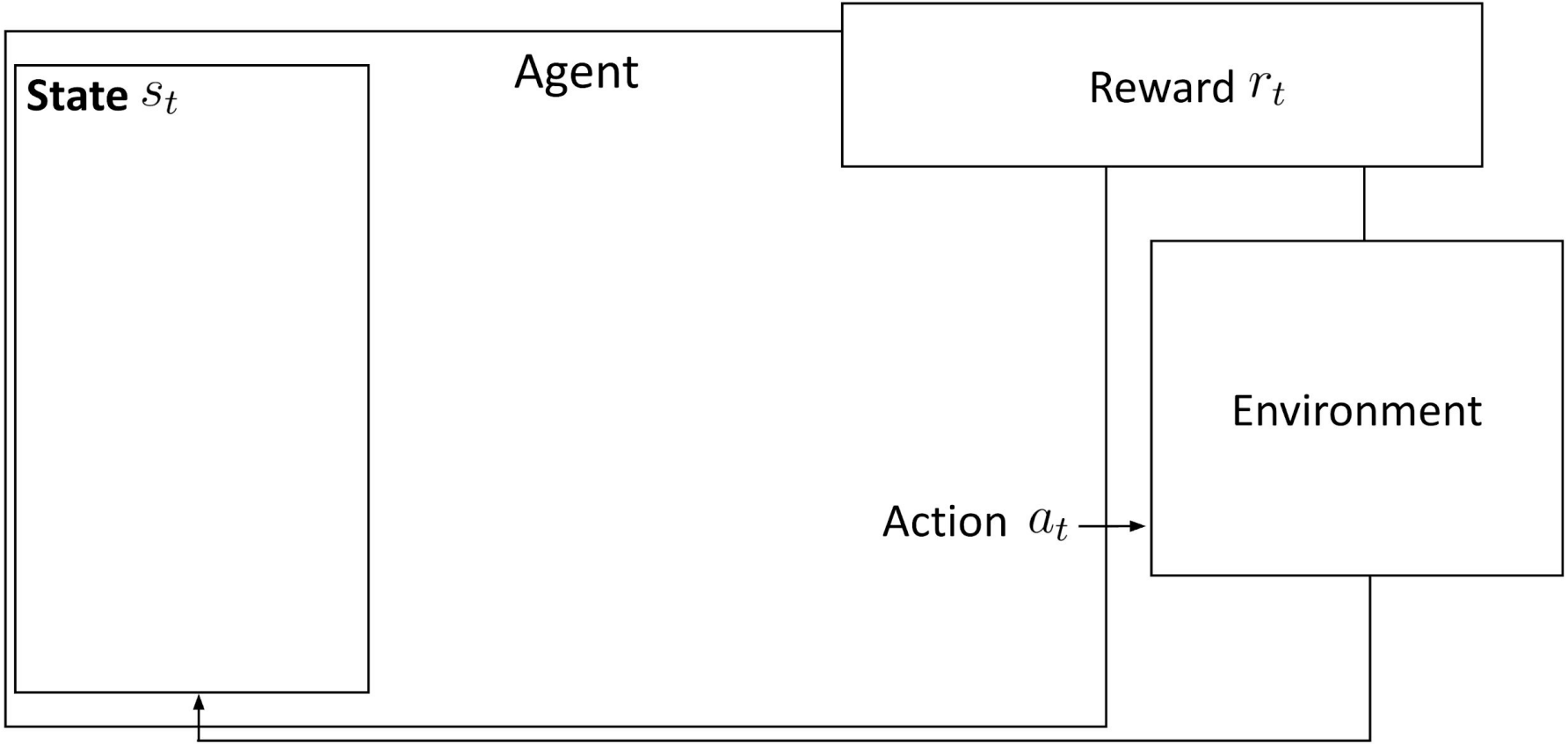


Reinforcement Learning (RL)

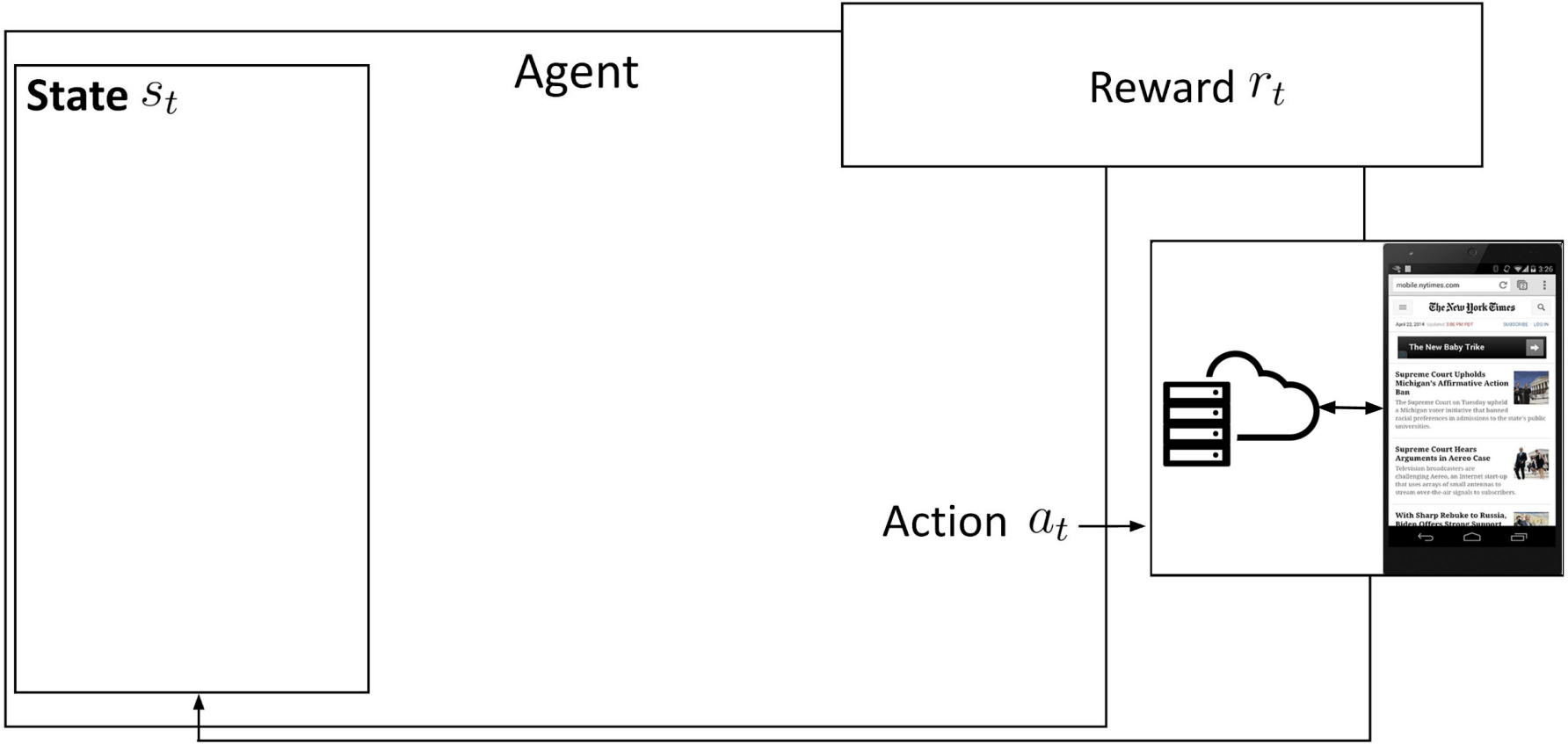


Goal: Find P_A that produces $\max \sum_t r_t$

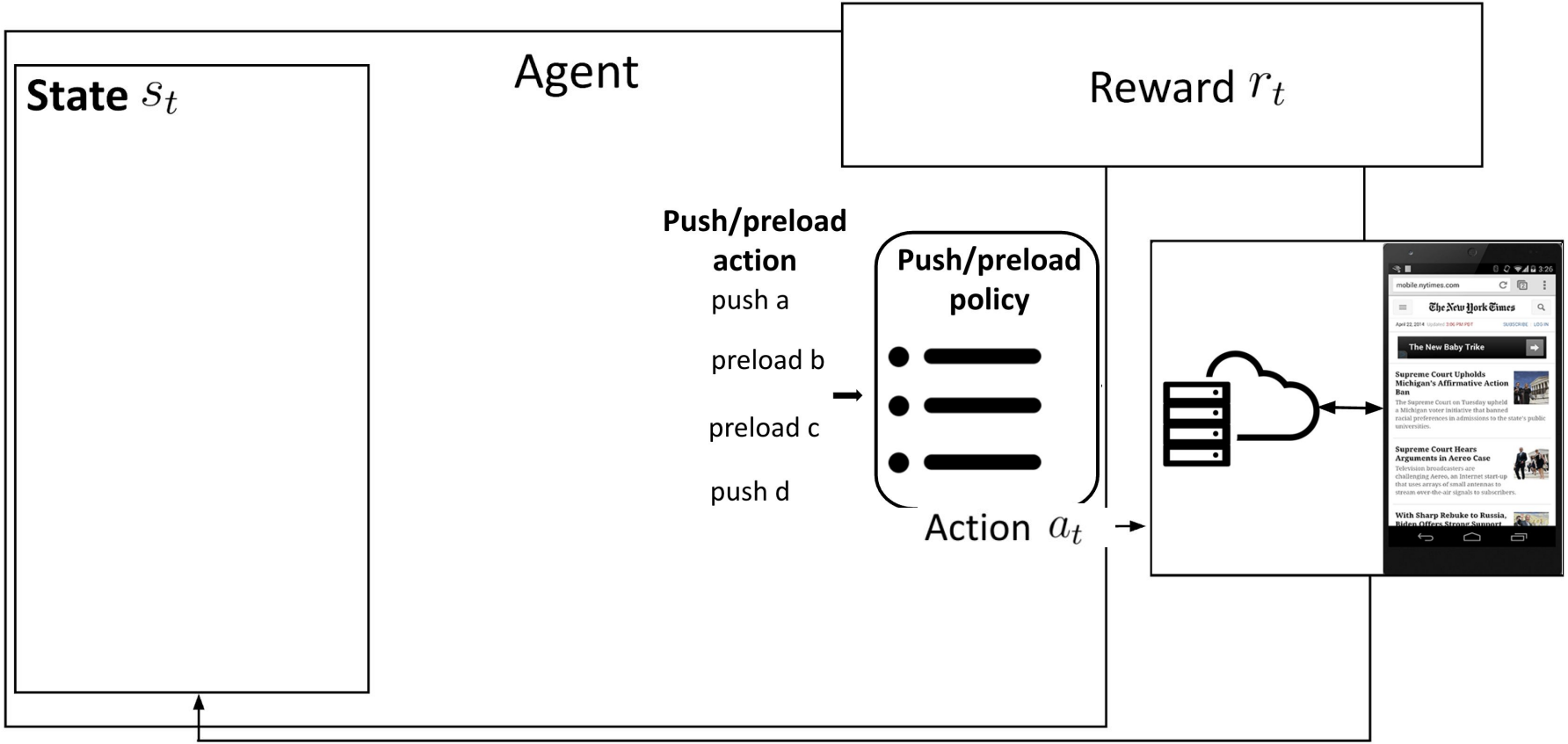
RL in Alohamora



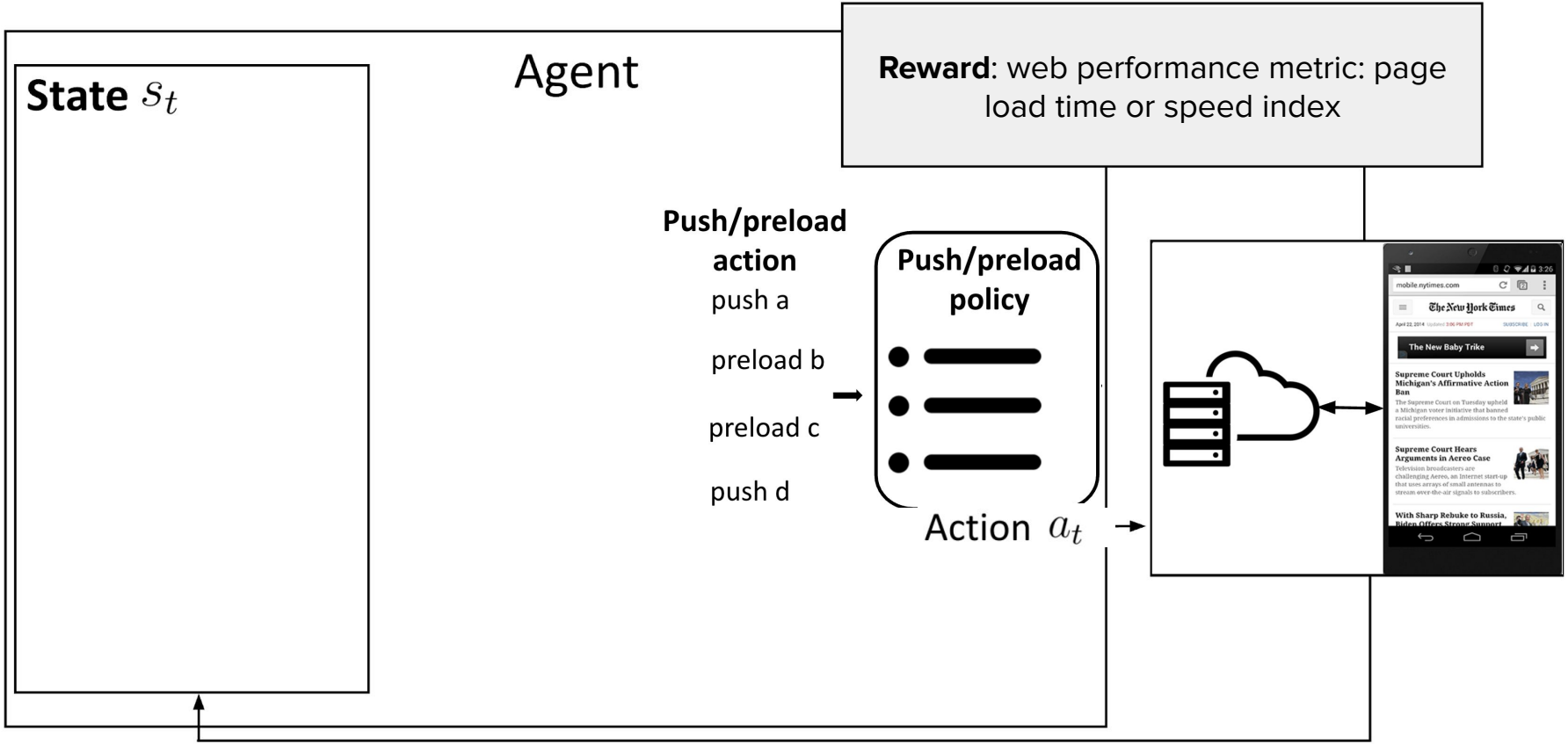
RL in Alohamora



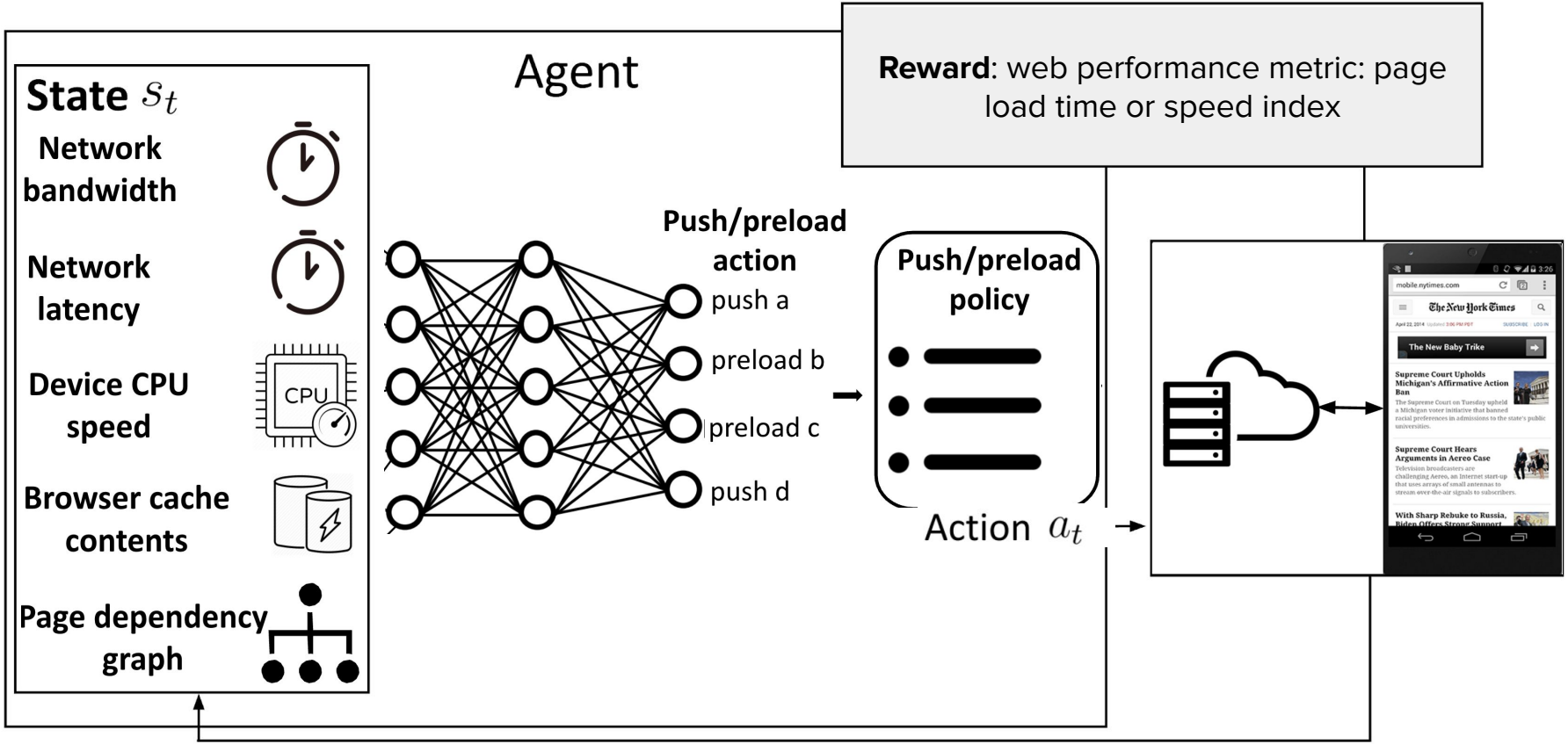
RL in Alohamora



RL in Alohamora



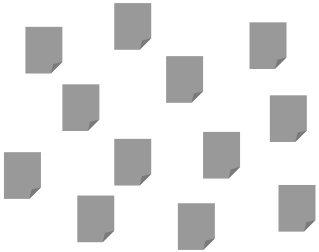
RL in Alohamora



Training challenges

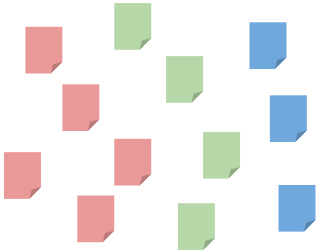
Training challenges

Challenge 1: Too many pages to consider



Training challenges

Challenge 1: Too many pages to consider



Structurally similar pages ➜
similar push/preload policies

Training challenges

Challenge 1: Too many pages to consider

Structurally similar pages ➜
similar push/preload policies

Training challenges

Challenge 1: Too many pages to consider

Structurally similar pages ➜
similar push/preload policies

Solution: train a single model per page cluster

Training challenges

Challenge 1: Too many pages to consider

Structurally similar pages ➜ similar push/preload policies

Solution: train a single model per page cluster

Challenge 2: Too many policies per page

Training challenges

Challenge 1: Too many pages to consider

Structurally similar pages ➜ similar push/preload policies

Solution: train a single model per page cluster

Challenge 2: Too many policies per page

Number of policies:
exponential in # of page objects

Testing time per policy:
10 seconds on average

New York Times homepage:
10 days to train for a single environment

Training challenges

Challenge 1: Too many pages to consider

Structurally similar pages ➜ similar push/preload policies

Solution: train a single model per page cluster

Challenge 2: Too many policies per page

Number of policies:
exponential in # of page objects

Testing time per policy:
10 seconds on average

New York Times homepage:
10 days to train for a single environment

Solution: simulate page loads without a real device/browser

Generalization

Goal
Avoid training one model per page

Challenge
Find clusters that optimize policy performance and number of clusters

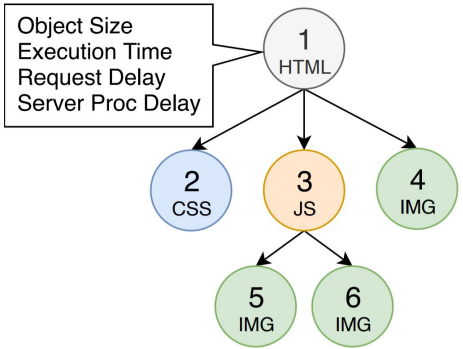
Insight
Policy efficacy is correlated with resource utilization → page structure

Generalization

Goal
Avoid training one model per page

Challenge
Find clusters that optimize policy performance and number of clusters

Insight
Policy efficacy is correlated with resource utilization ➜ page structure



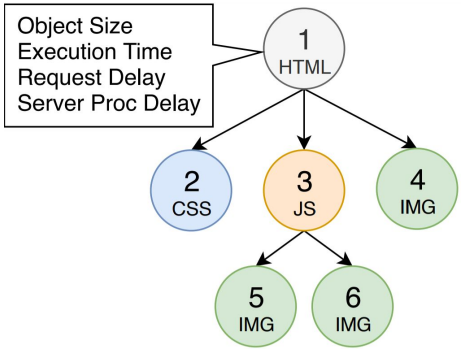
Page dependency tree captures resource utilization

Generalization

Goal
Avoid training one model per page

Challenge
Find clusters that optimize policy performance and number of clusters

Insight
Policy efficacy is correlated with resource utilization → page structure



Page dependency tree captures resource utilization

$$\begin{bmatrix} D(T_1,T_1) & \cdots & D(T_1,T_j) & \cdots & D(T_1,T_n) \\ \vdots & \ddots & & & \\ D(T_i,T_1) & & D(T_i,T_j) & & \\ \vdots & & & \ddots & \\ D(T_n,T_1) & & & & D(T_n,T_n) \end{bmatrix}$$

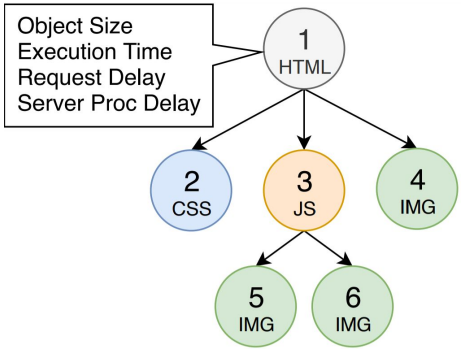
Distance matrix where D is the edit distance between two trees

Generalization

Goal
Avoid training one model per page

Challenge
Find clusters that optimize policy performance and number of clusters

Insight
Policy efficacy is correlated with resource utilization → page structure



Page dependency tree captures resource utilization

$$\begin{bmatrix} D(T_1,T_1) & \cdots & D(T_1,T_j) & \cdots & D(T_1,T_n) \\ \vdots & \ddots & & & \\ D(T_i,T_1) & & D(T_i,T_j) & & \\ \vdots & & & \ddots & \\ D(T_n,T_1) & & & & D(T_n,T_n) \end{bmatrix}$$

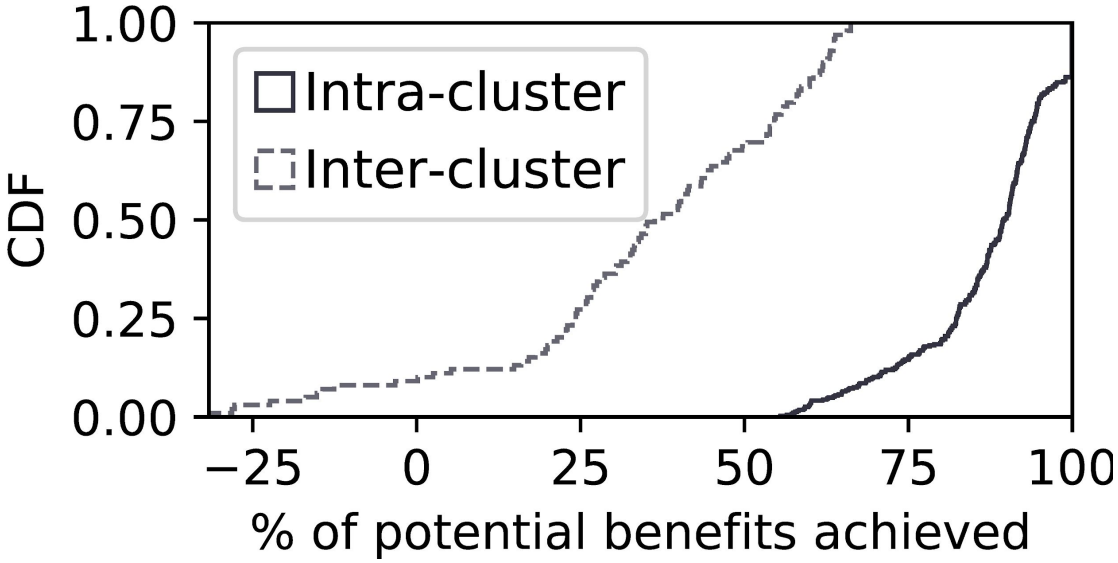
Distance matrix where D is the edit distance between two trees

Generalization

Policies within the same cluster achieve **82-100%** of benefits

Policies applied across clusters forego **31-64%** of benefits

We can train **one model** using the "median page" **per cluster**



Page Load Simulator

Goal
Predict policy performance on a page load in a specific environment

Challenge
Be accurate across arbitrary environments, modeling cross-stack constraints

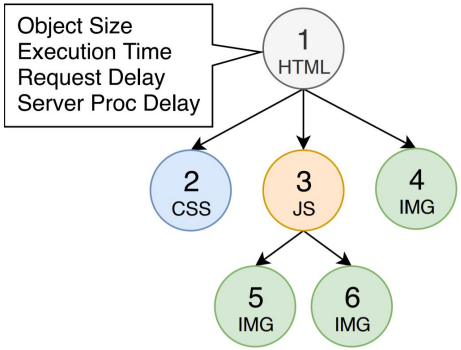
Insight
Model the browser as a multi-queue system and modulate unshaped loads

Page Load Simulator

Goal
Predict policy performance on a page load in a specific environment

Challenge
Be accurate across arbitrary environments, modeling cross-stack constraints

Insight
Model the browser as a multi-queue system and modulate unshaped loads

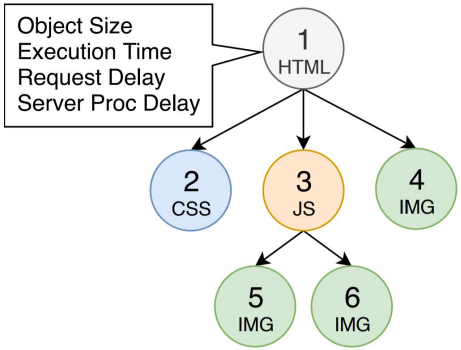


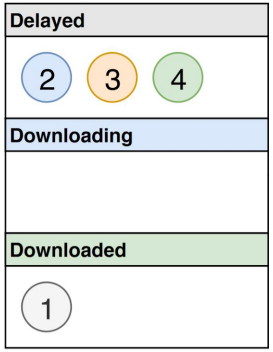
Page Load Simulator

Goal
Predict policy performance on a page load in a specific environment

Challenge
Be accurate across arbitrary environments, modeling cross-stack constraints

Insight
Model the browser as a multi-queue system and modulate unshaped loads



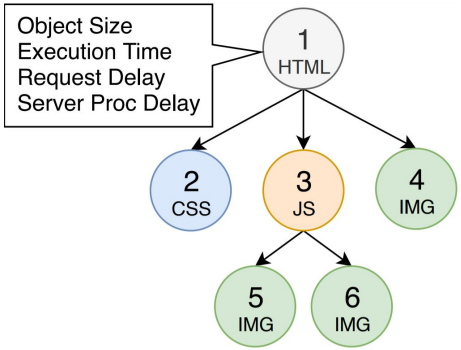


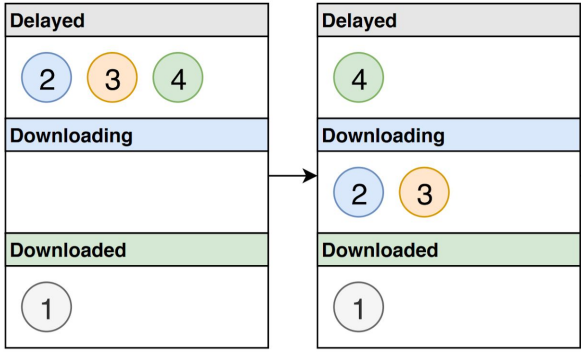
Page Load Simulator

Goal
Predict policy performance on a page load in a specific environment

Challenge
Be accurate across arbitrary environments, modeling cross-stack constraints

Insight
Model the browser as a multi-queue system and modulate unshaped loads



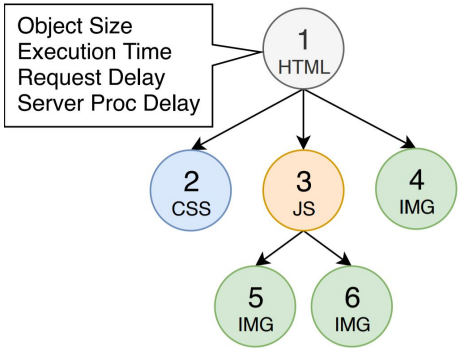


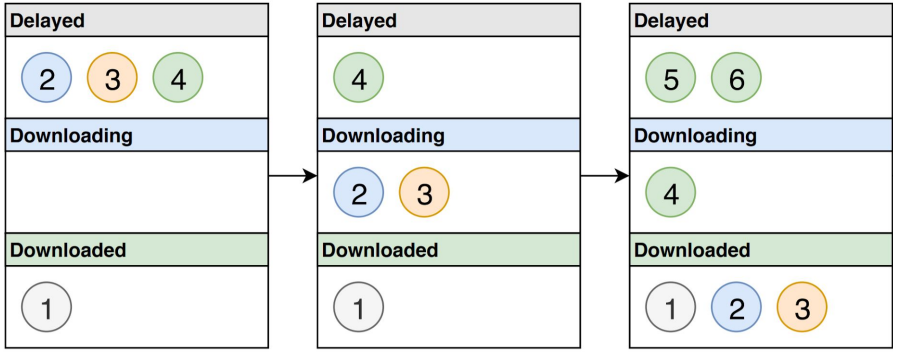
Page Load Simulator

Goal
Predict policy performance on a page load in a specific environment

Challenge
Be accurate across arbitrary environments, modeling cross-stack constraints

Insight
Model the browser as a multi-queue system and modulate unshaped loads



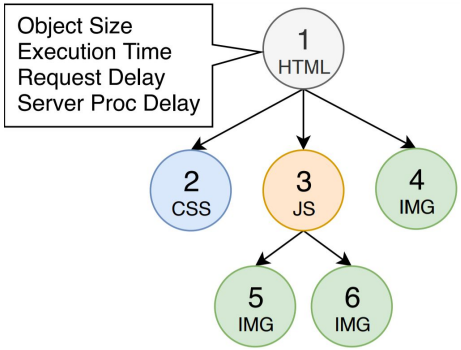


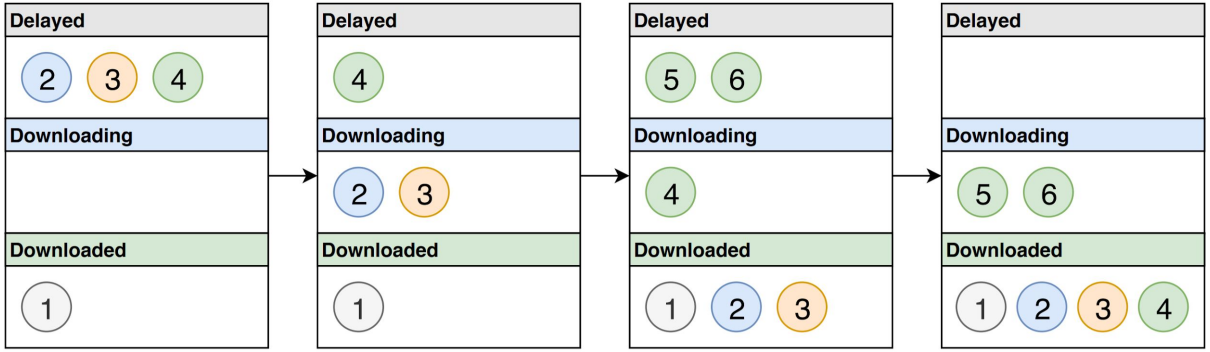
Page Load Simulator

Goal
Predict policy performance on a page load in a specific environment

Challenge
Be accurate across arbitrary environments, modeling cross-stack constraints

Insight
Model the browser as a multi-queue system and modulate unshaped loads



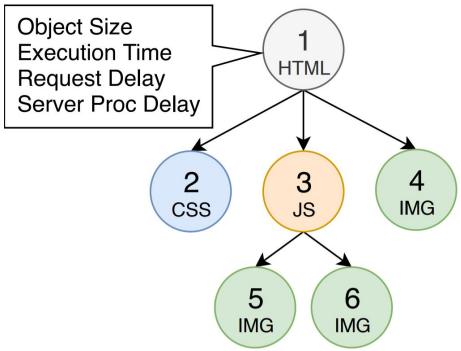


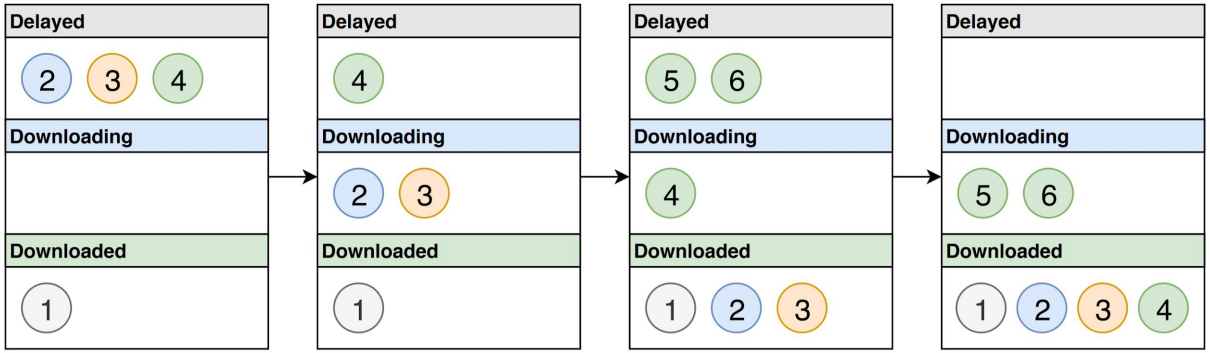
Page Load Simulator

Goal
Predict policy performance on a page load in a specific environment

Challenge
Be accurate across arbitrary environments, modeling cross-stack constraints

Insight
Model the browser as a multi-queue system and modulate unshaped loads





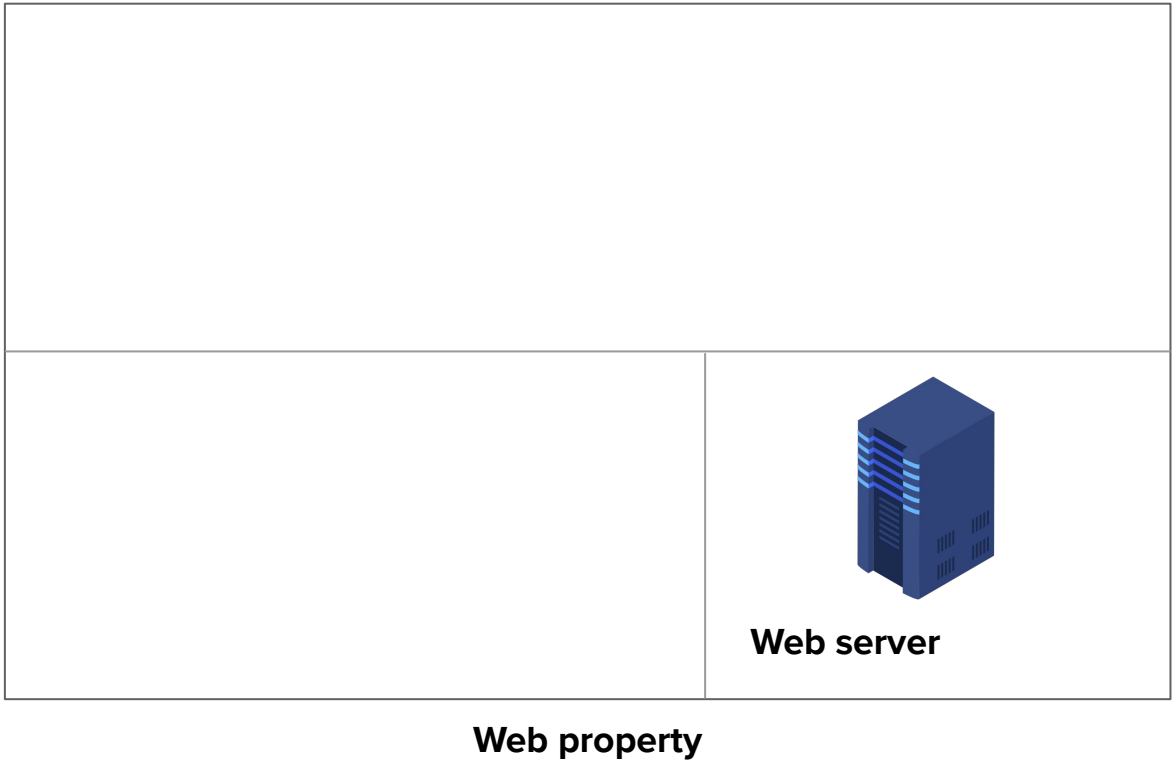
Speed
Time to load page:
10s → 5ms.

Faithfulness
Median error:
0.4%.

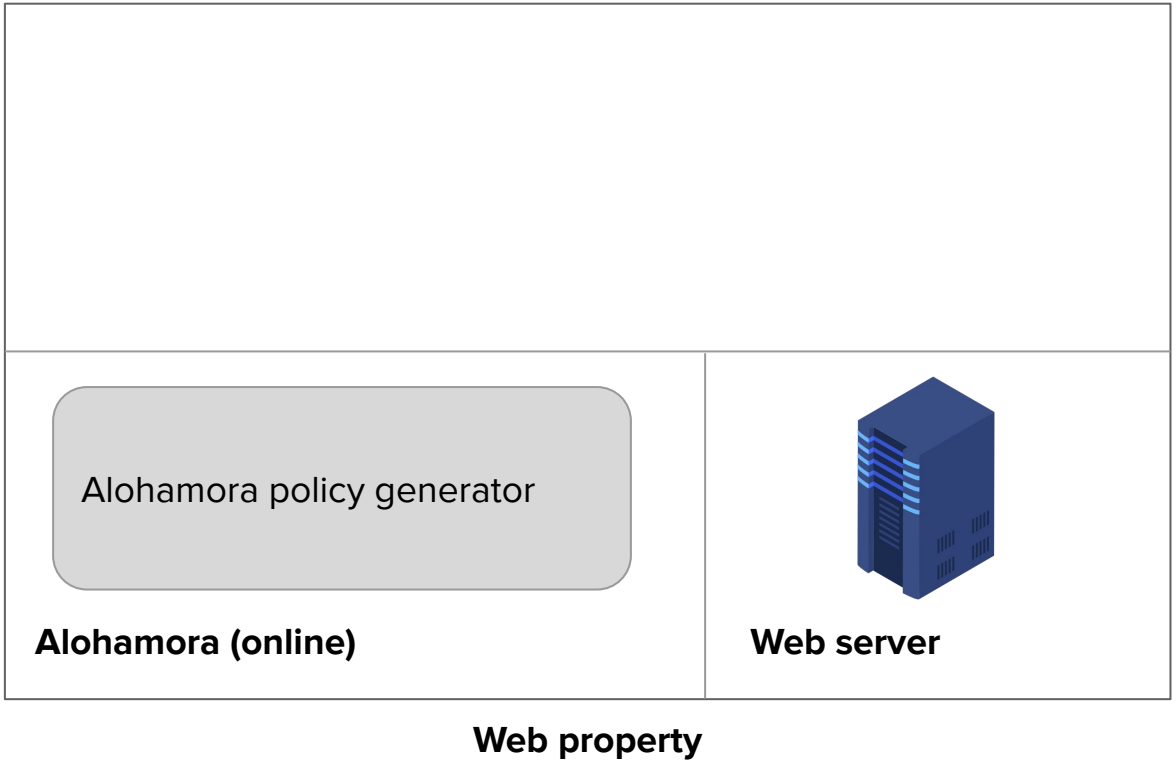
Time to train:
10 days → 20 mins

System design (online inference)

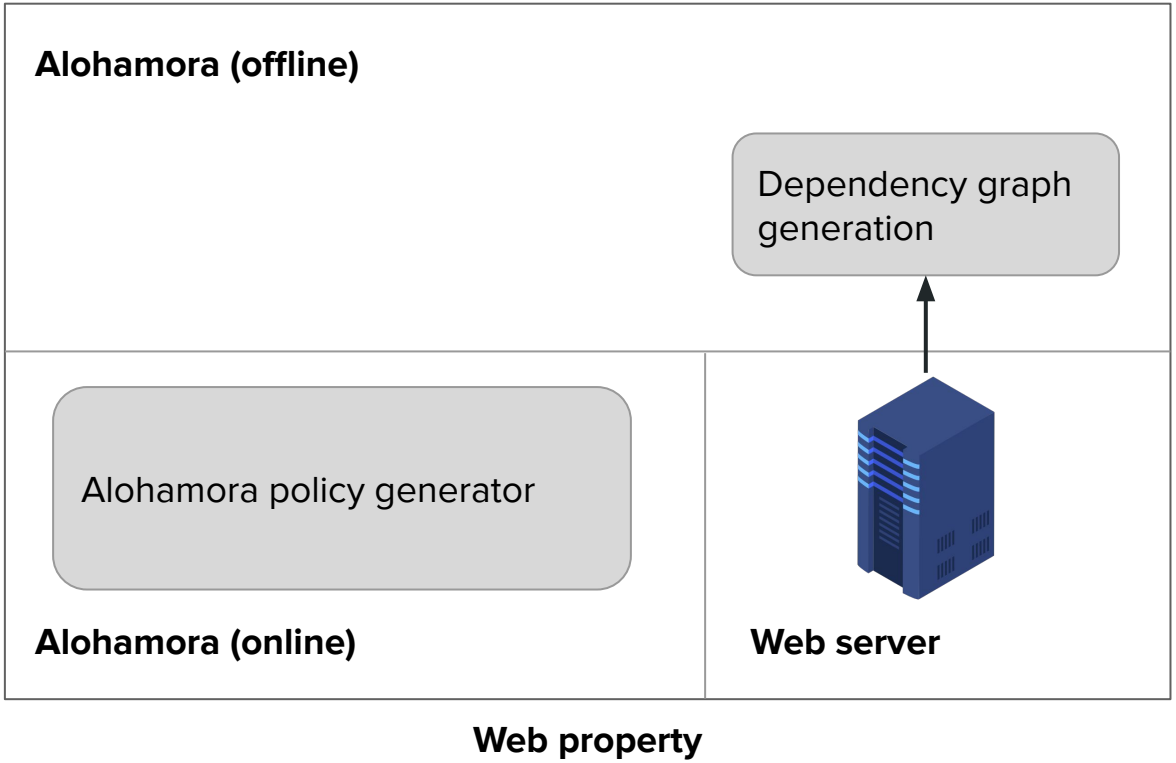
System design (online inference)



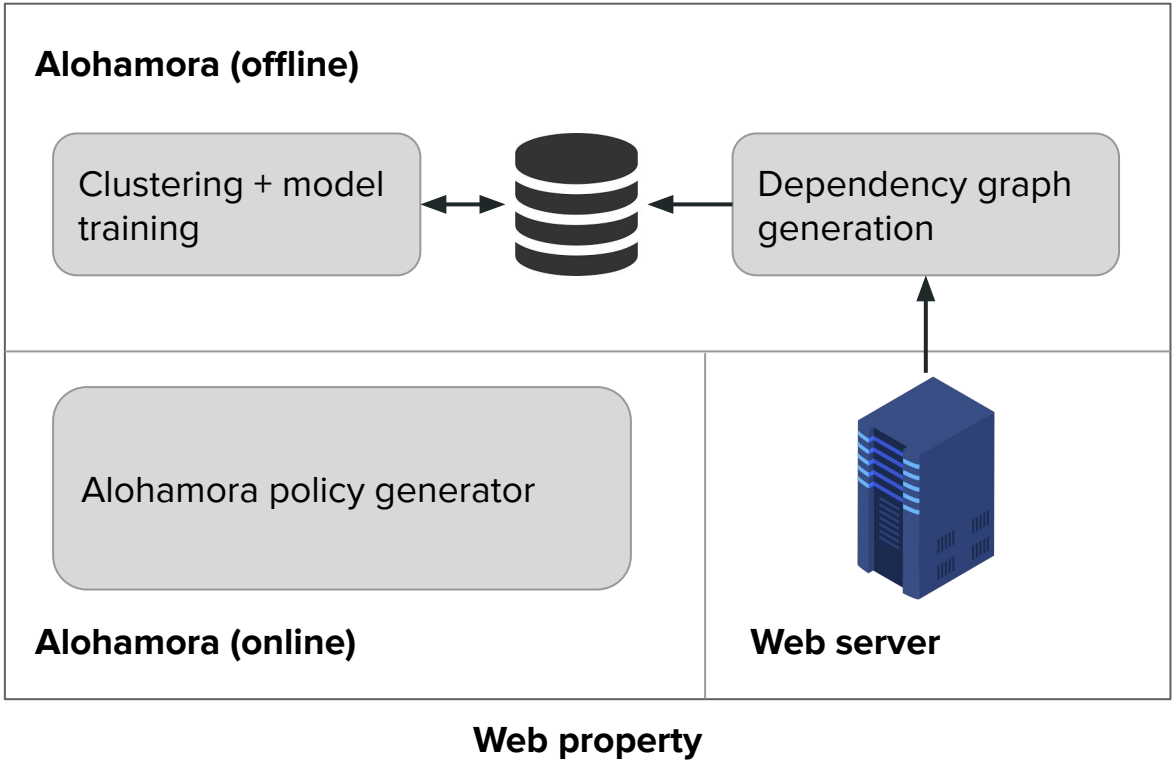
System design (online inference)



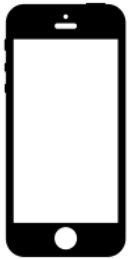
System design (online inference)



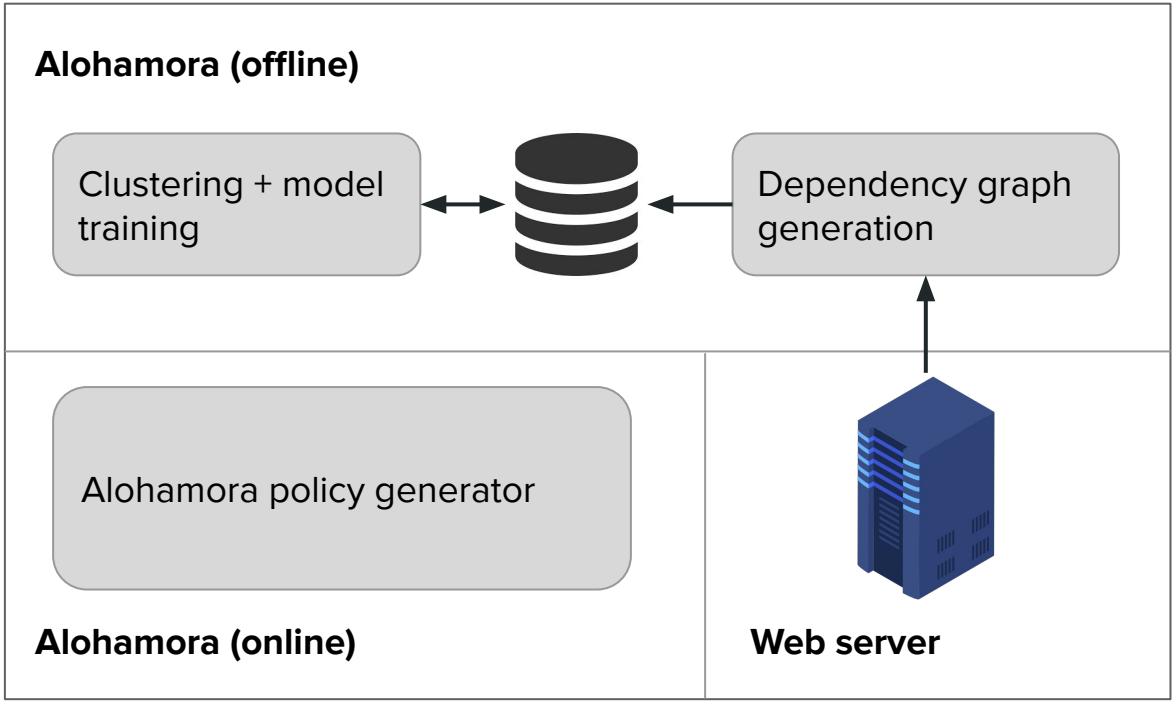
System design (online inference)



System design (online inference)

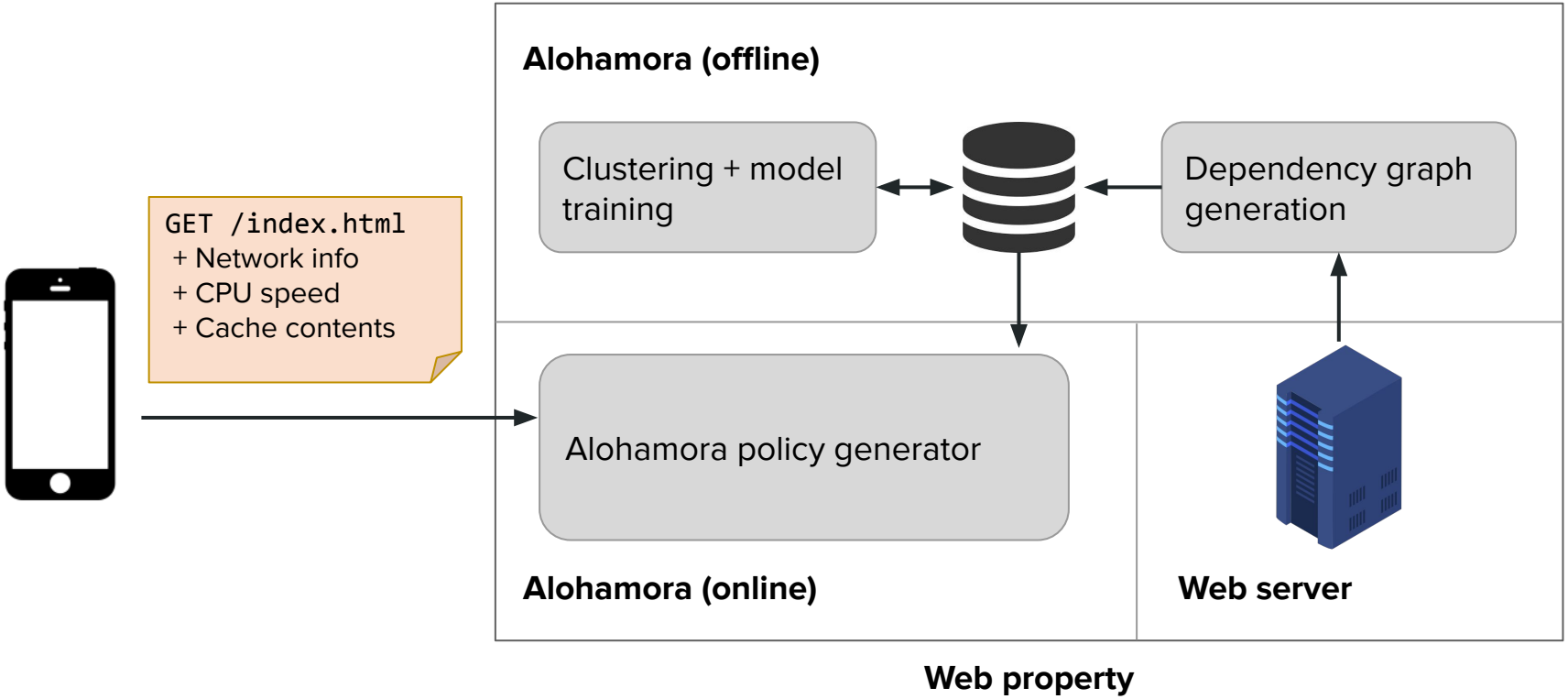


```
GET /index.html
 + Network info
 + CPU speed
 + Cache contents
```

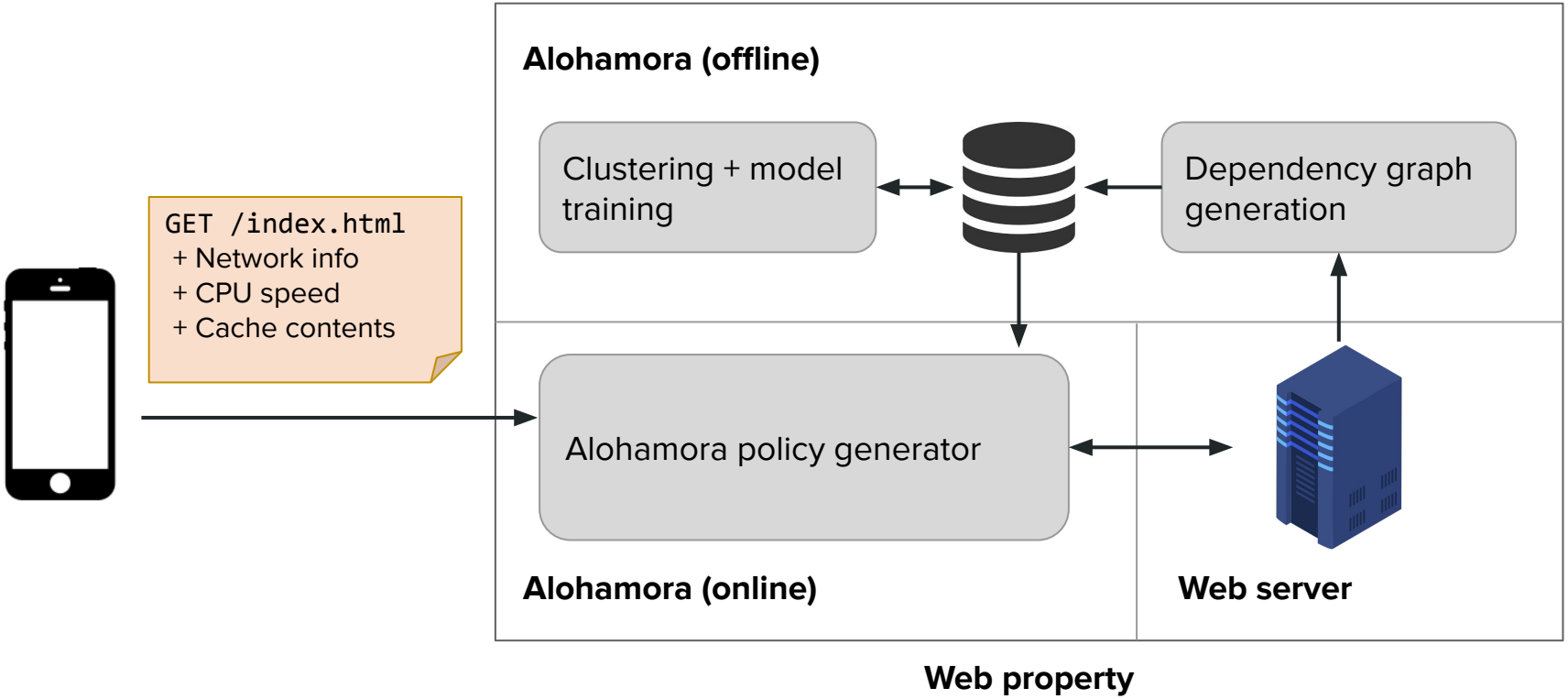


Web property

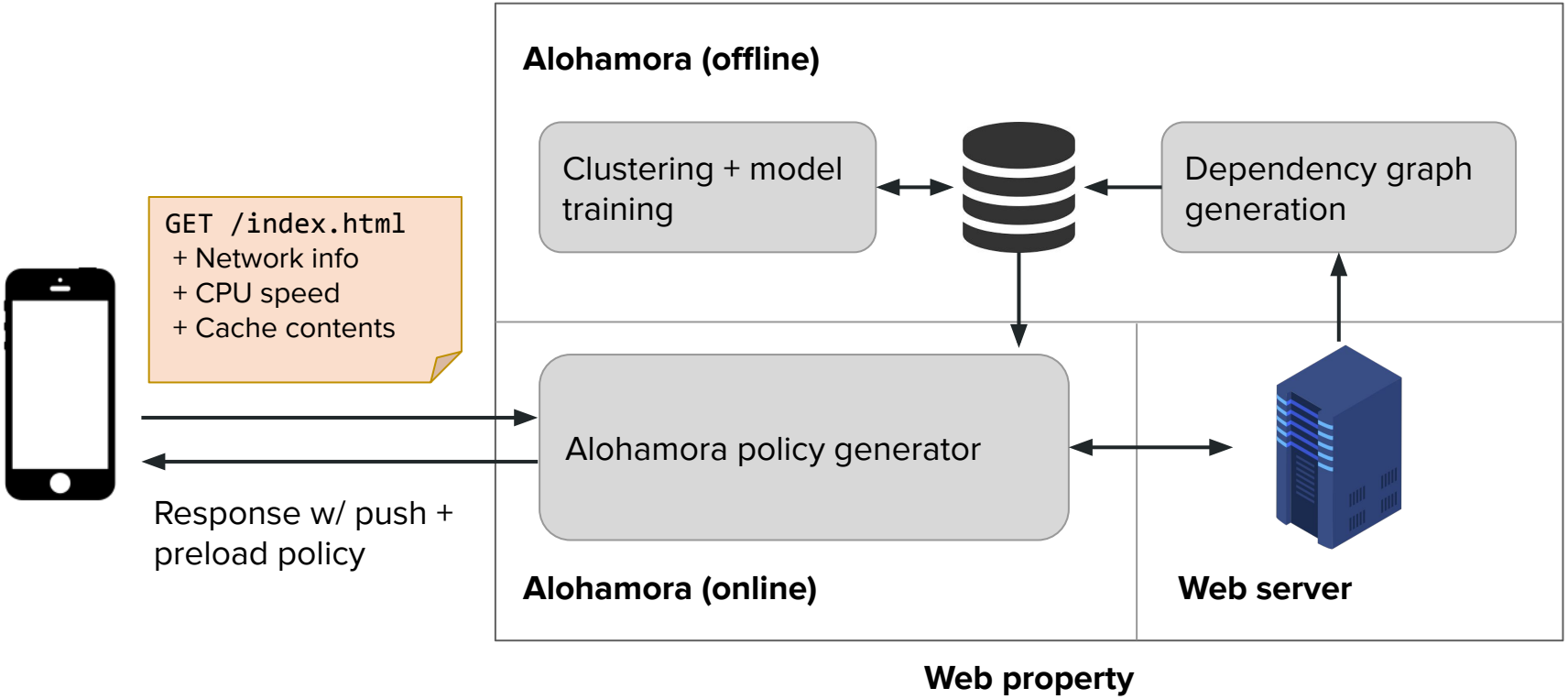
System design (online inference)



System design (online inference)

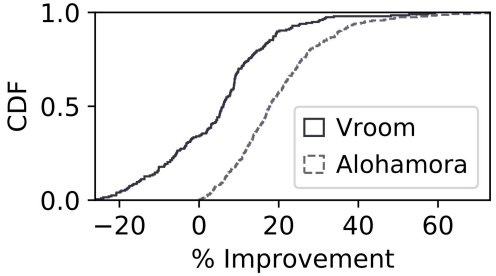


System design (online inference)

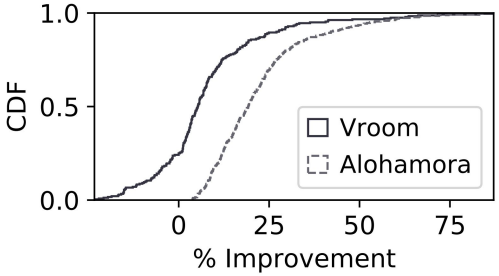


Results

24-61% improvement in PLT,
19-48% improvement in SI,
1.4 to 3.6x faster than Vroom



12Mbps, 100ms, cold, 2x, 0% (PLT)

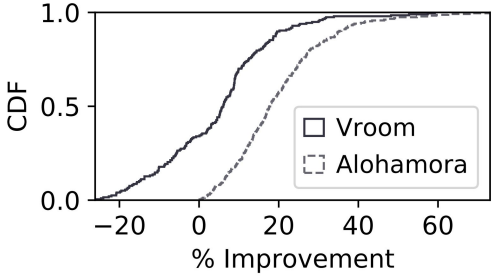


24Mbps, 20ms, cold, 1x, 0% (SI)

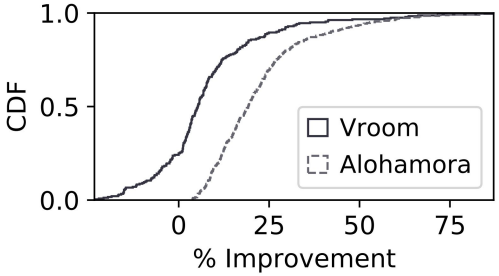
Results

24-61% improvement in PLT,
19-48% improvement in SI,
1.4 to 3.6x faster than Vroom

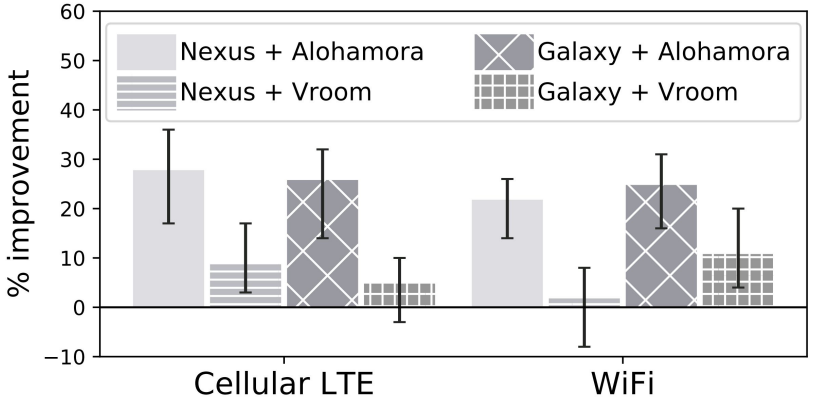
Validated performance in real-world settings



12Mbps, 100ms, cold, 2x, 0% (PLT)



24Mbps, 20ms, cold, 1x, 0% (SI)

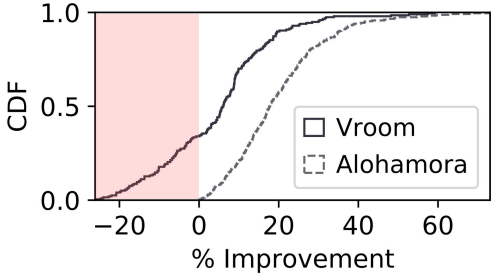


Results

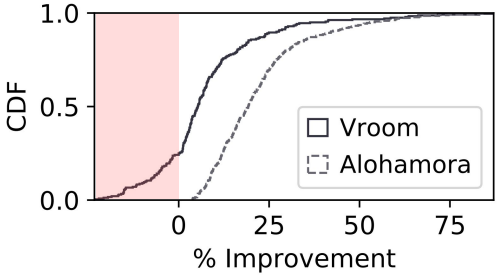
24-61% improvement in PLT,
19-48% improvement in SI,
1.4 to 3.6x faster than Vroom

Validated performance in real-world settings

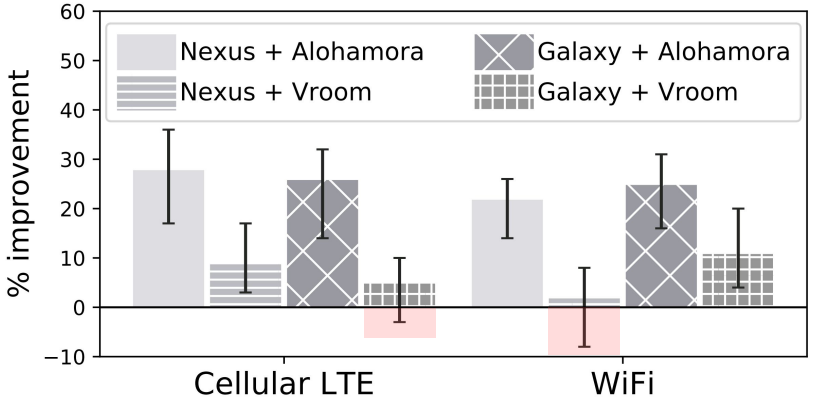
Alohamora **never** degrades page load performance



12Mbps, 100ms, cold, 2x, 0% (PLT)



24Mbps, 20ms, cold, 1x, 0% (SI)



Conclusion

- Alohamora adapts push/preload policies based on the target page and execution environment
- Ensures practicality via:
  - Cross-page clustering
  - Cross-stack page load simulation
- Alohamora can speed up page loads up to 61%, and never slows them down.

https://github.com/nkansal96/alohamora