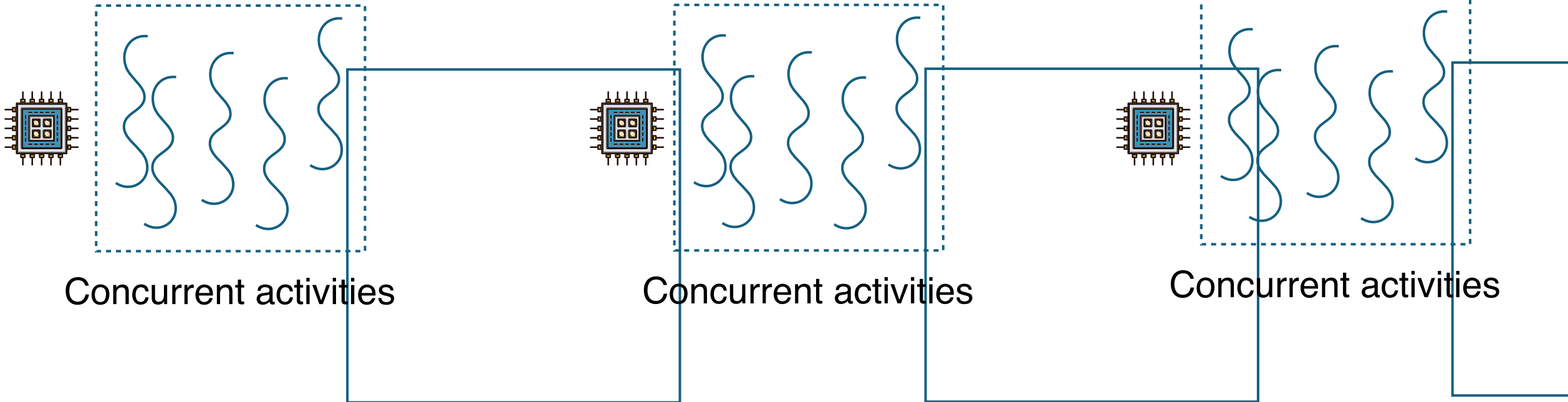


Mitigating Application Resource Overload with Targeted Task Cancellation

Yigong Hu, Zeyin Zhang, **Yicheng Liu**, Yile Gu, Shuangyu Lei, Baris Kasikci, Peng Huang



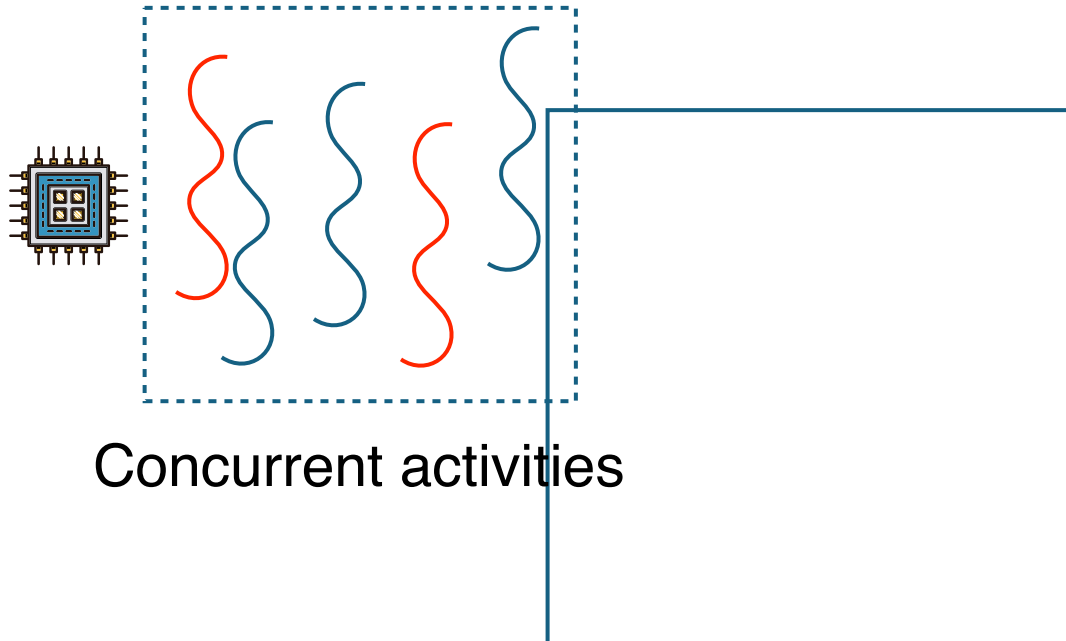
Datacenter Apps Have Tight SLO Requirements



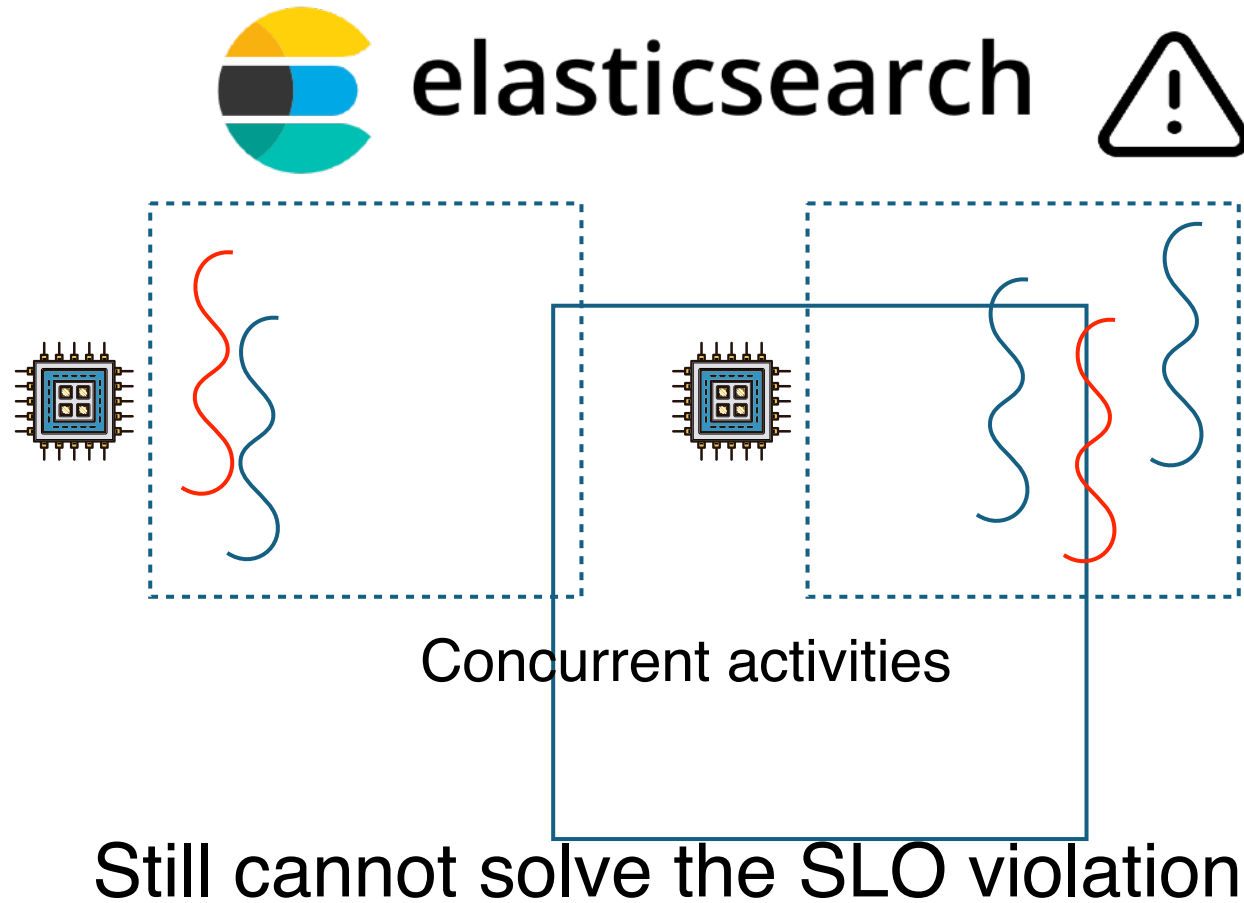
Requirements: Tight SLO

Problem: Often violated because of limited resources

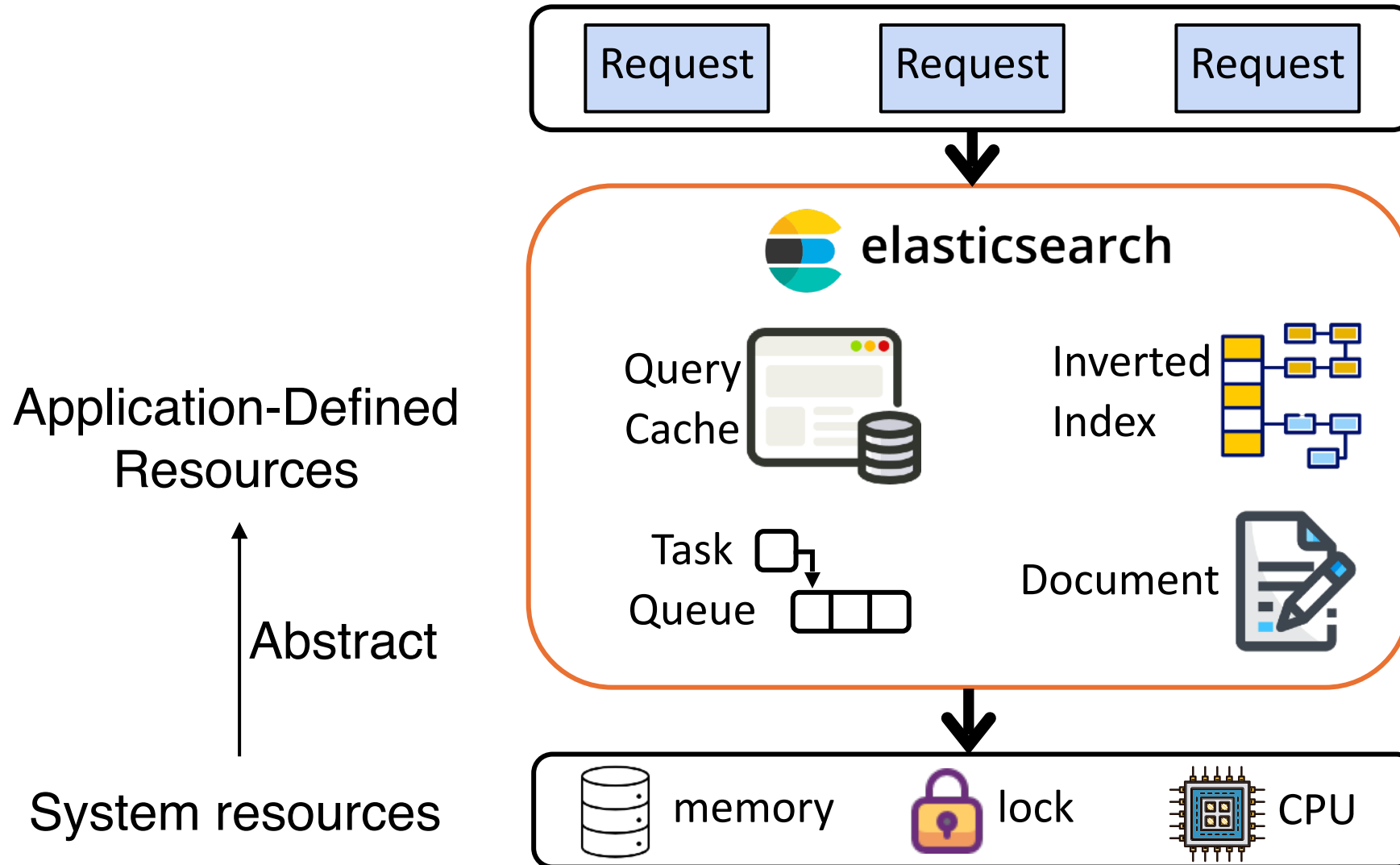
Assign More System Resources?



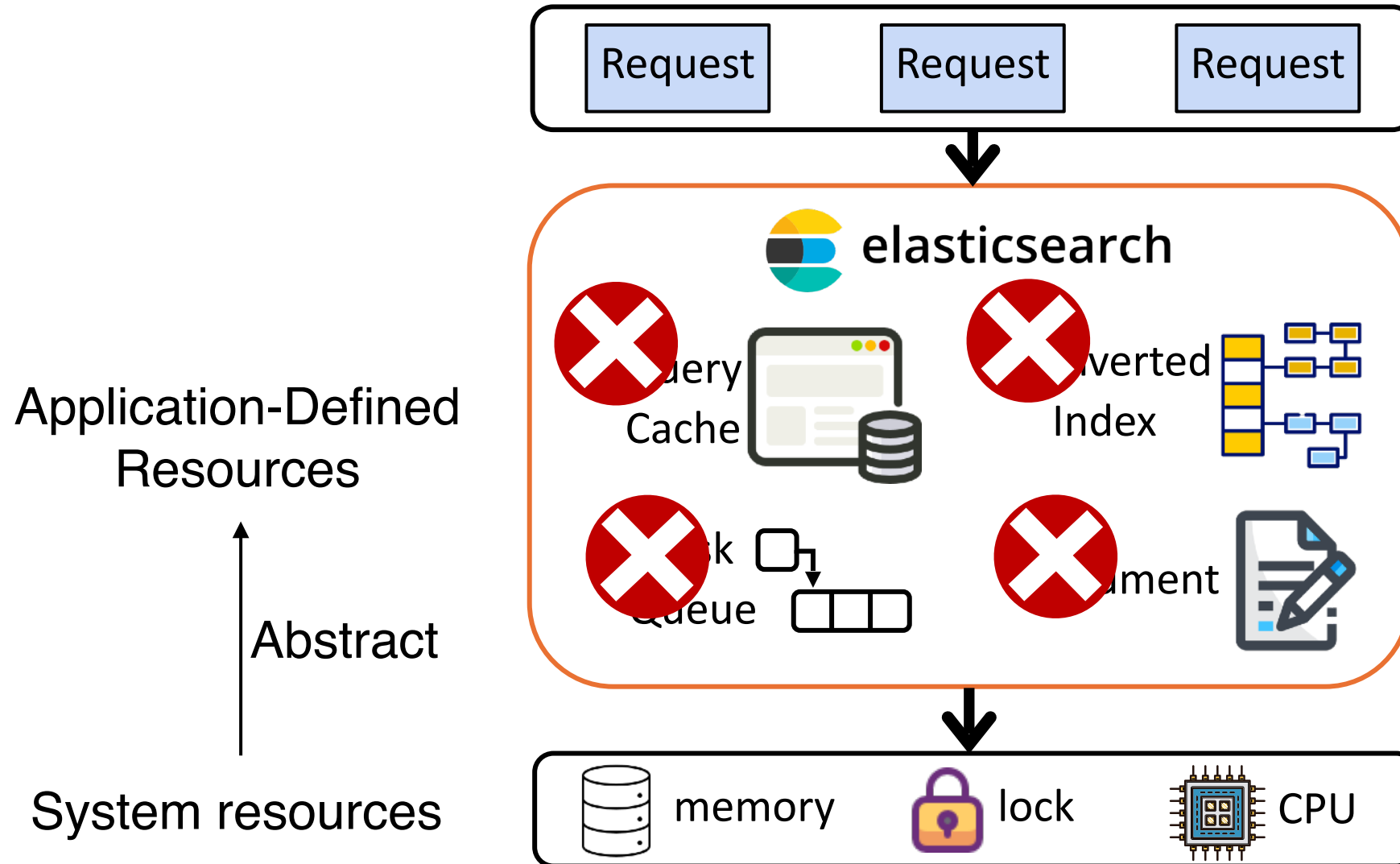
Assign More System Resources?



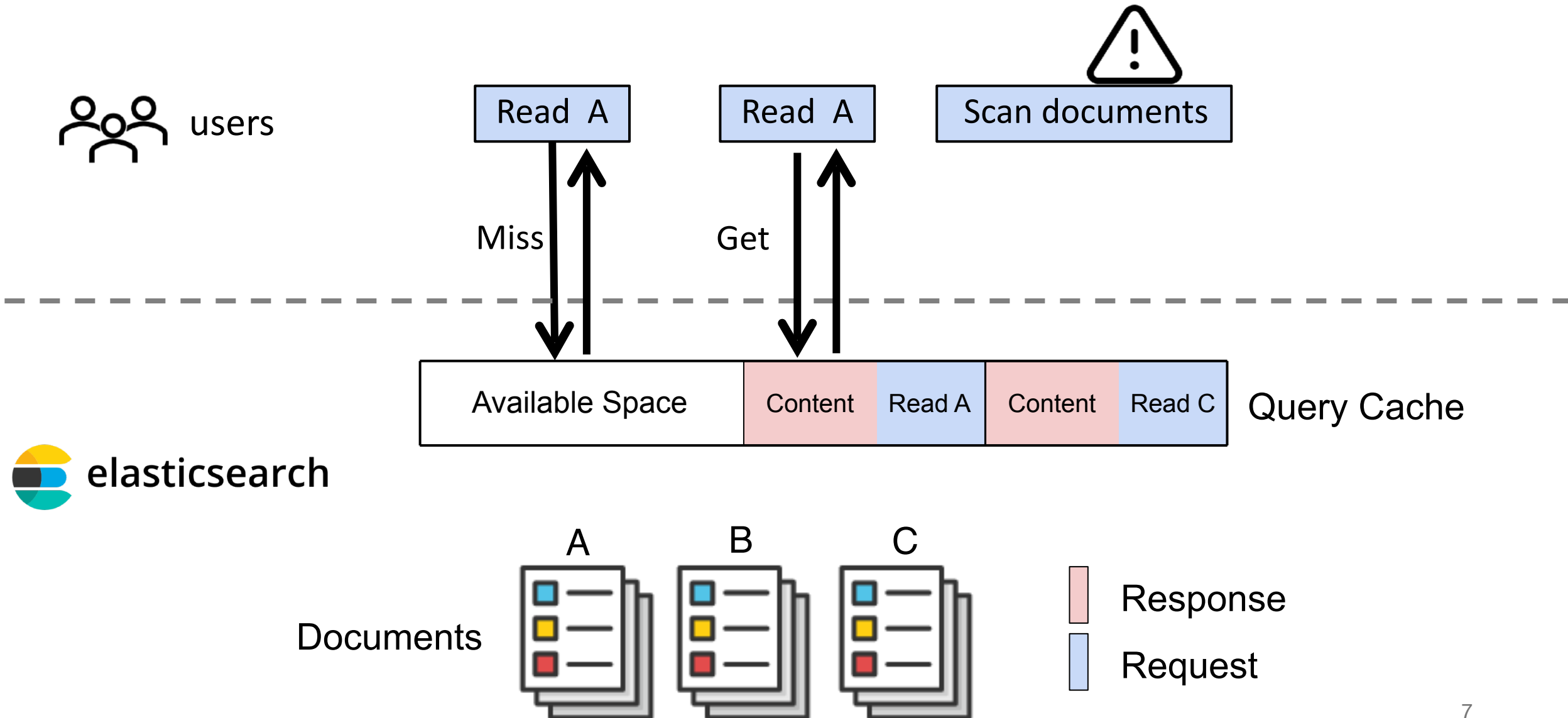
App-Defined Resources Are Critical to App Perf.



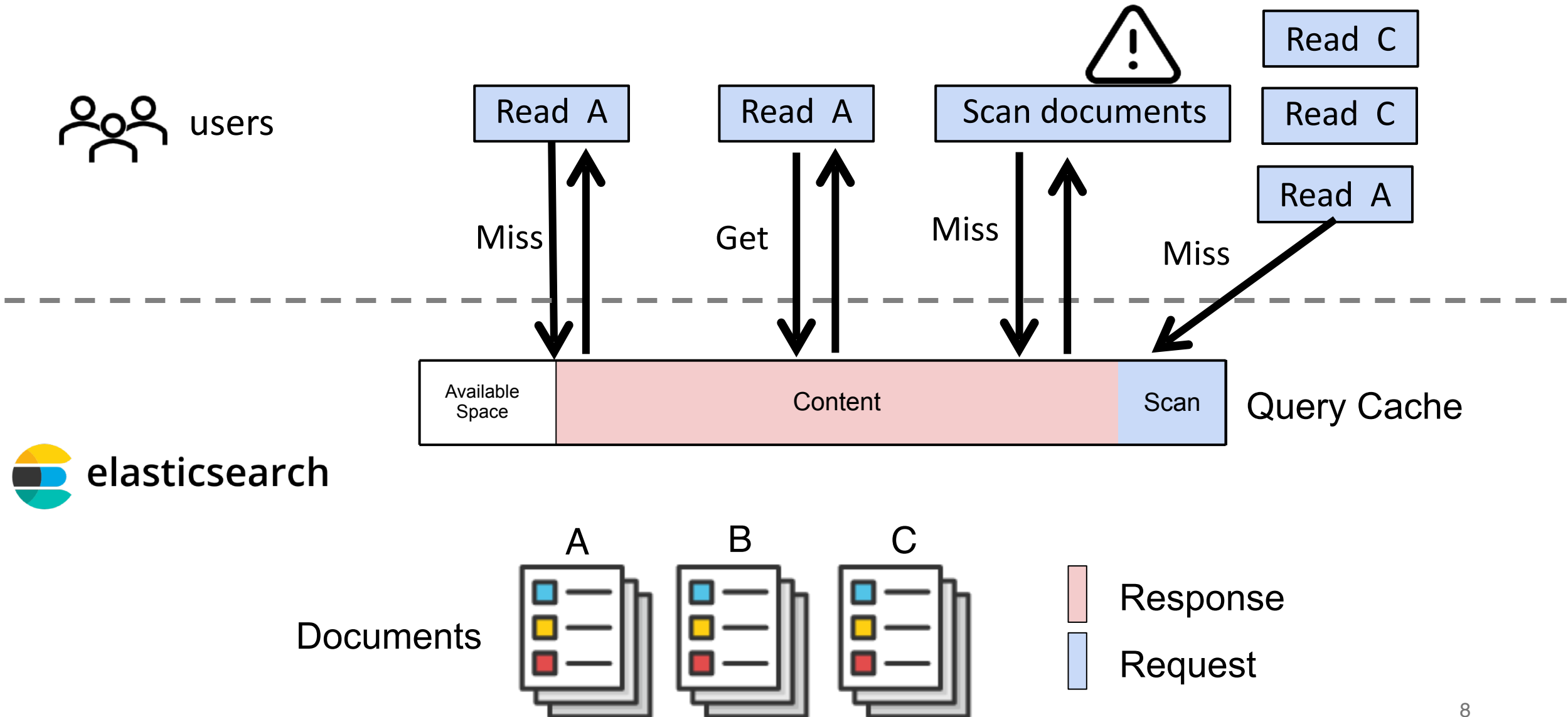
Overload Happens in App-Defined Resources



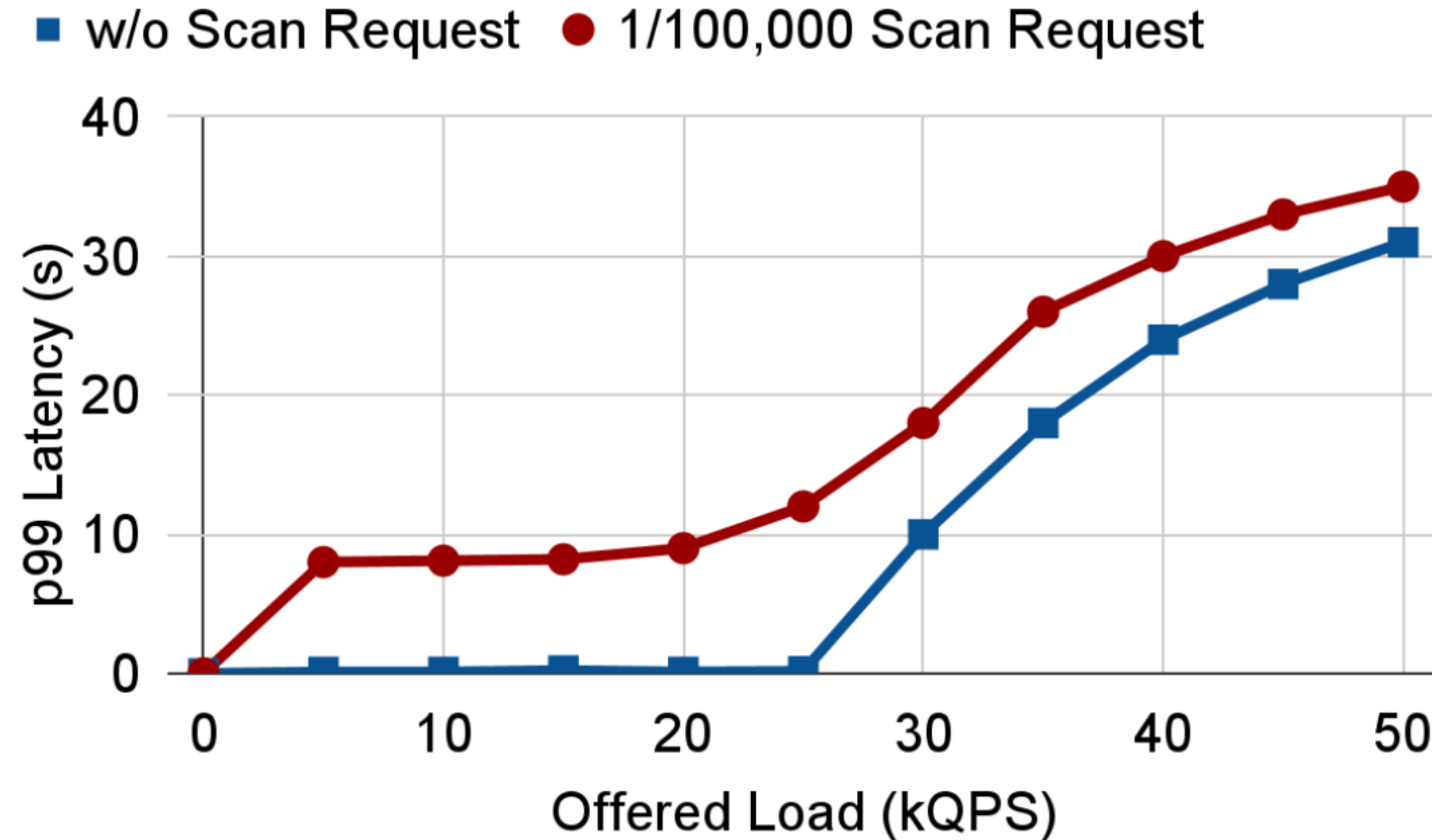
Overload Example: Es. Query Cache



Overload Example: Es. Query Cache

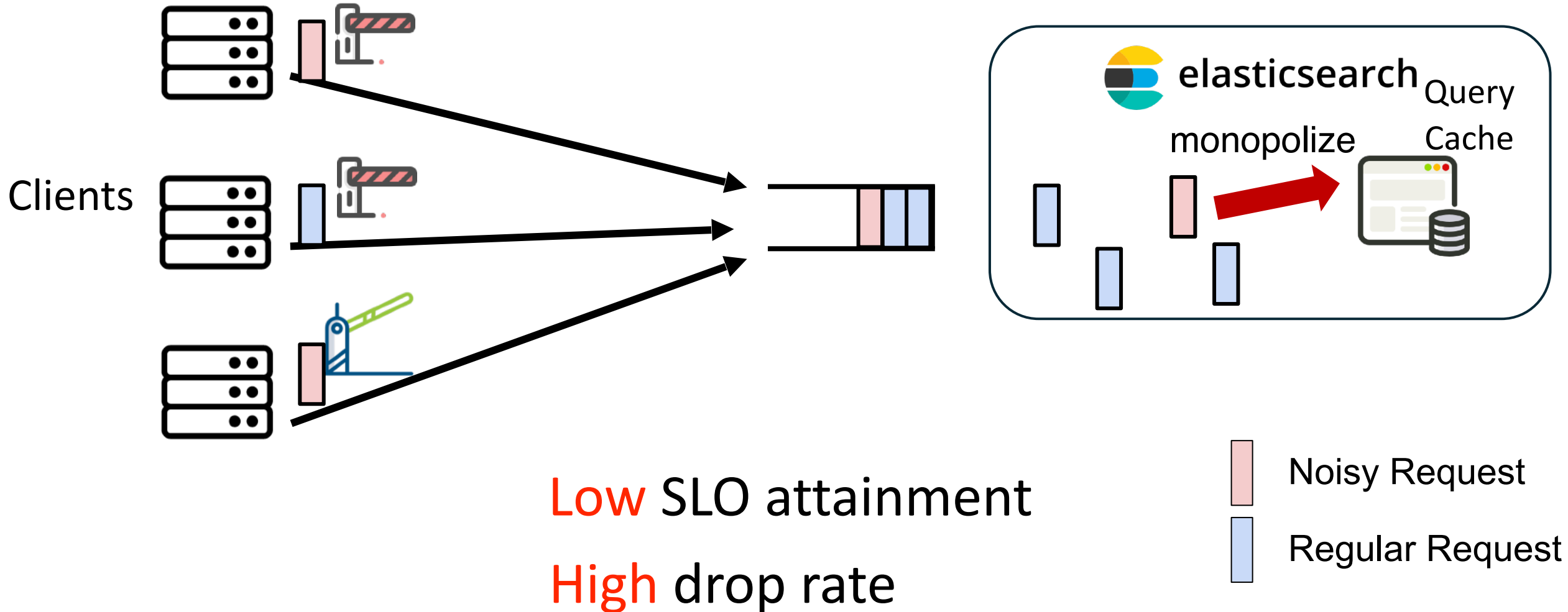


One Query Can Overload App-Defined Resource

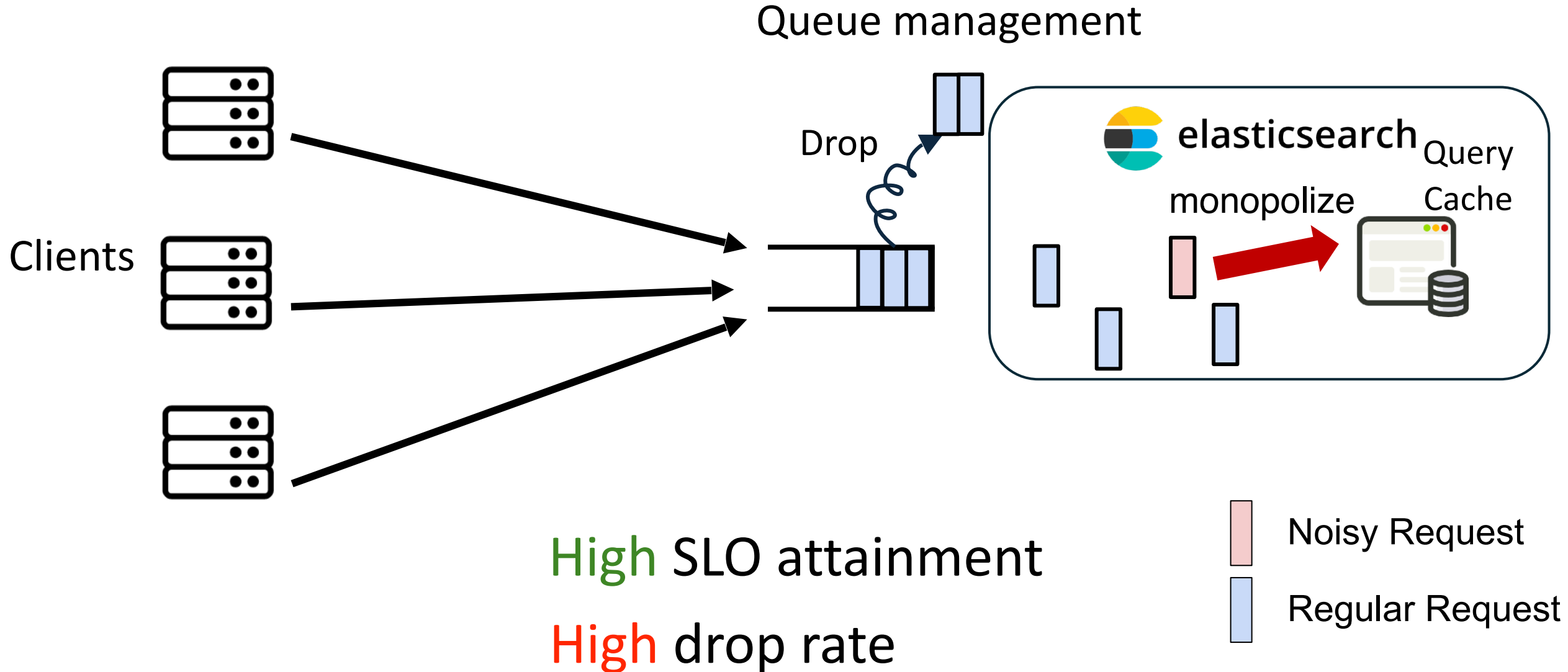


Existing Overload Control Is Insufficient

Admission control

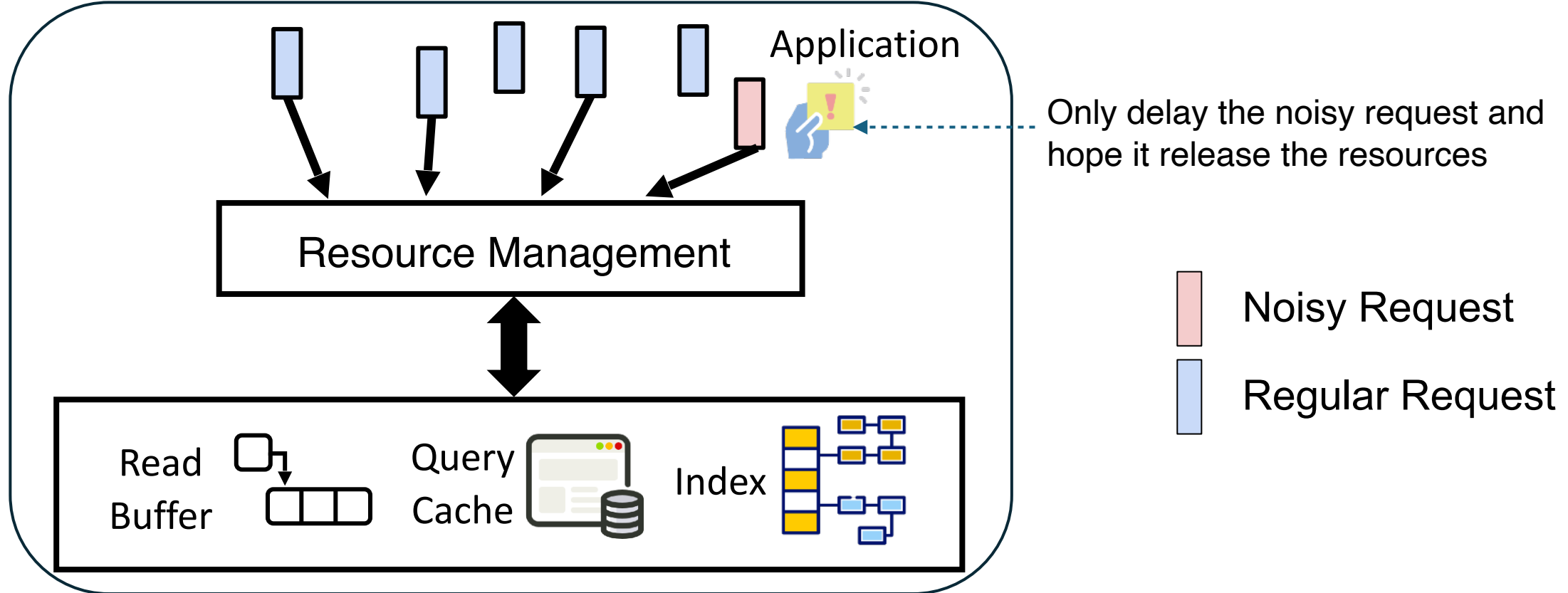


Existing Overload Control Is Insufficient



**Key Takeaway: Overload Control Solutions
Should be Aware of App-Defined Resources**

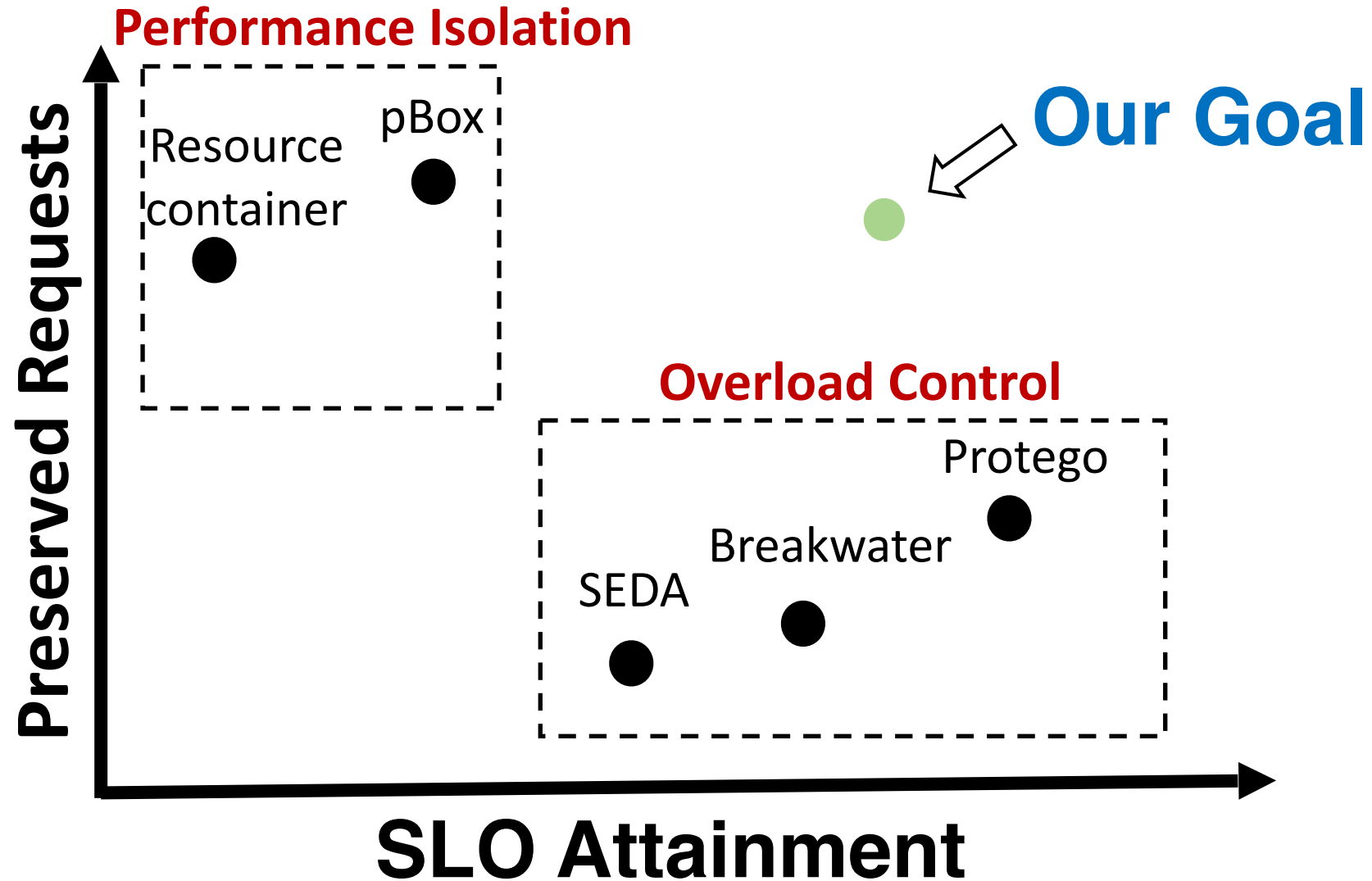
Performance Isolation Is Best-effort



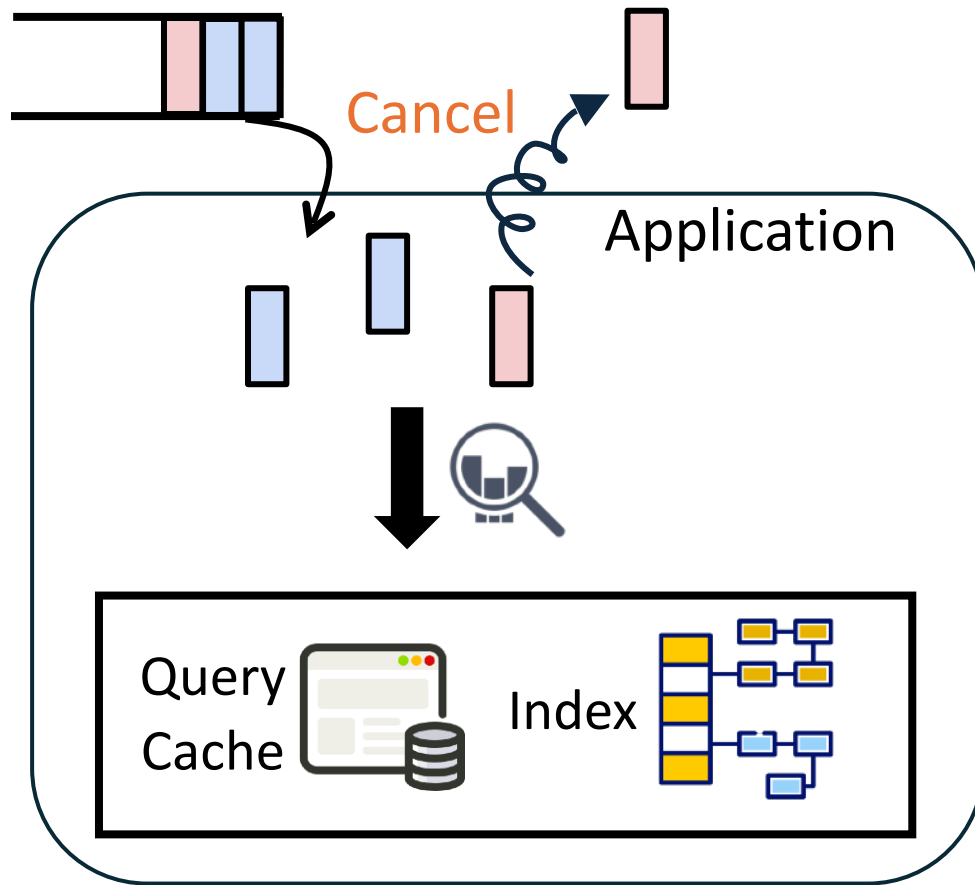
Aware of App-defined resources

Low SLO attainment

Mitigating the Application-Defined Resource Overload



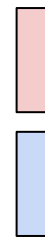
Atropos: A System to Mitigate App Resource Overload



Allow any requests to execute first

Monitor end-to-end performance and per-task app-defined resource usage

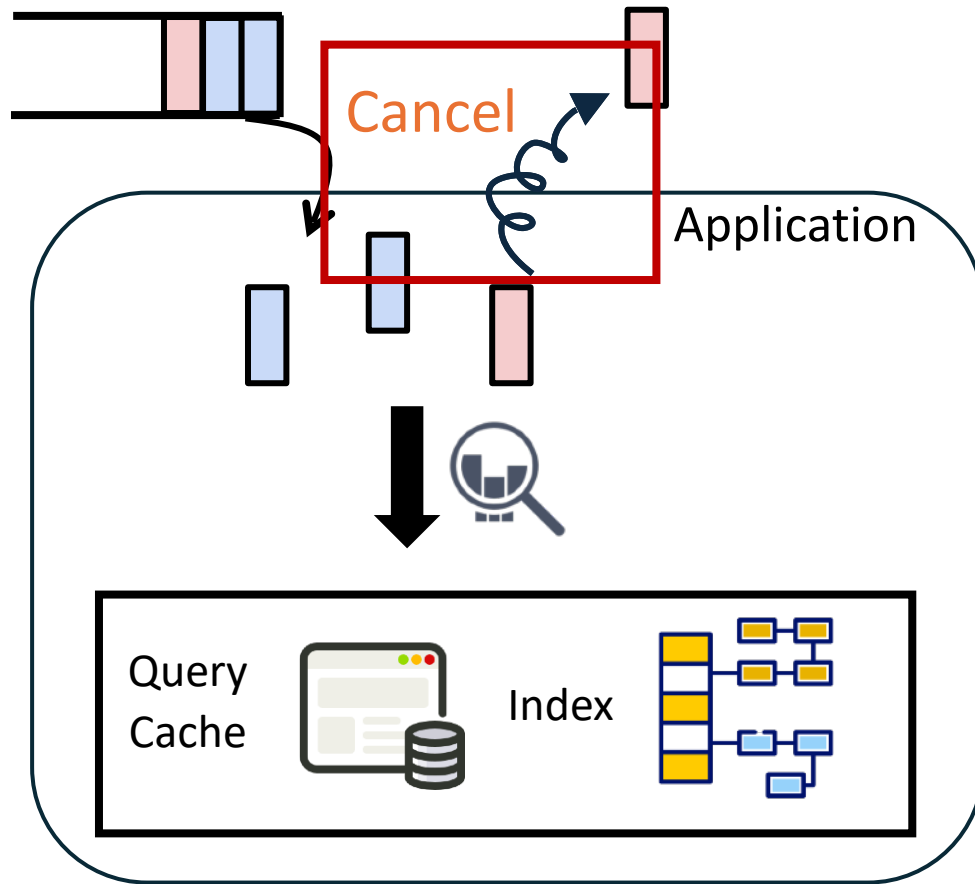
If resource overload happens, **cancel** the request that releases the most load



Noisy Request

Regular Request

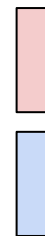
Atropos: A System to Mitigate App Resource Overload



Allow any requests to execute first

Monitor end-to-end performance and per-task app-defined resource usage

If resource overload happens, **cancel** the request that releases the most load



Noisy Request

Regular Request

Cancel Request via Safe Cancellation

```
1    public void run() {  
2        try { ...  
3        } catch (InterruptedException e) {  
4            // receiver handles the cancel request  
5        }  
6        ...  
7        if (Thread.currentThread().isInterrupted()) {  
8            // receiver handles the cancel request  
9        }  
10    }
```

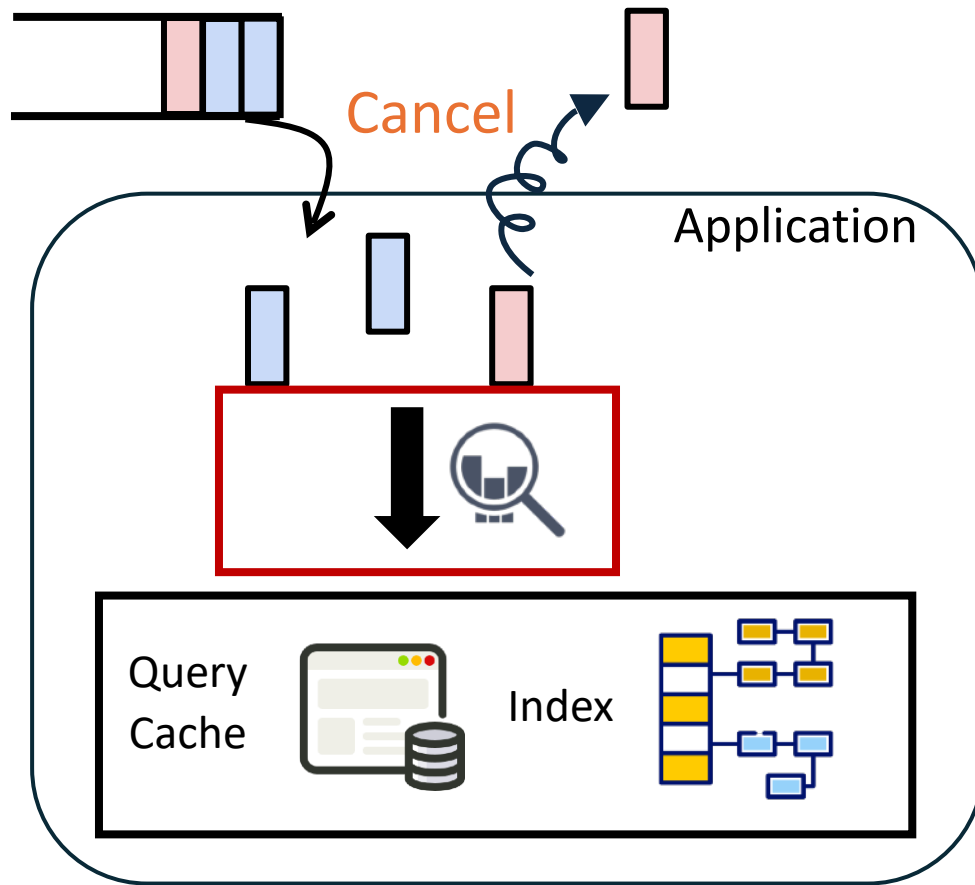
Example: Implement safe cancellation with Interruption in Java
Safe cancellation requires properly resource releasing, states clearing,
etc.

Safe Cancellation Has Been Widely Supported

Language	Applications	Supporting Cancel	With Initiator
C/C++	60	49	46
Java	34	25	25
Go	44	32	29
Python	13	9	9
Total	151	115 (76%)	109 (95% of 115)

Throughout **4** different languages in **115** out of **151** applications

Atropos: A System to Mitigate App Resource Overload



Allow any requests to execute first

Monitor end-to-end performance and per-task app-defined resource usage

If resource overload happens, **cancel** the request that releases the most load



Noisy Request

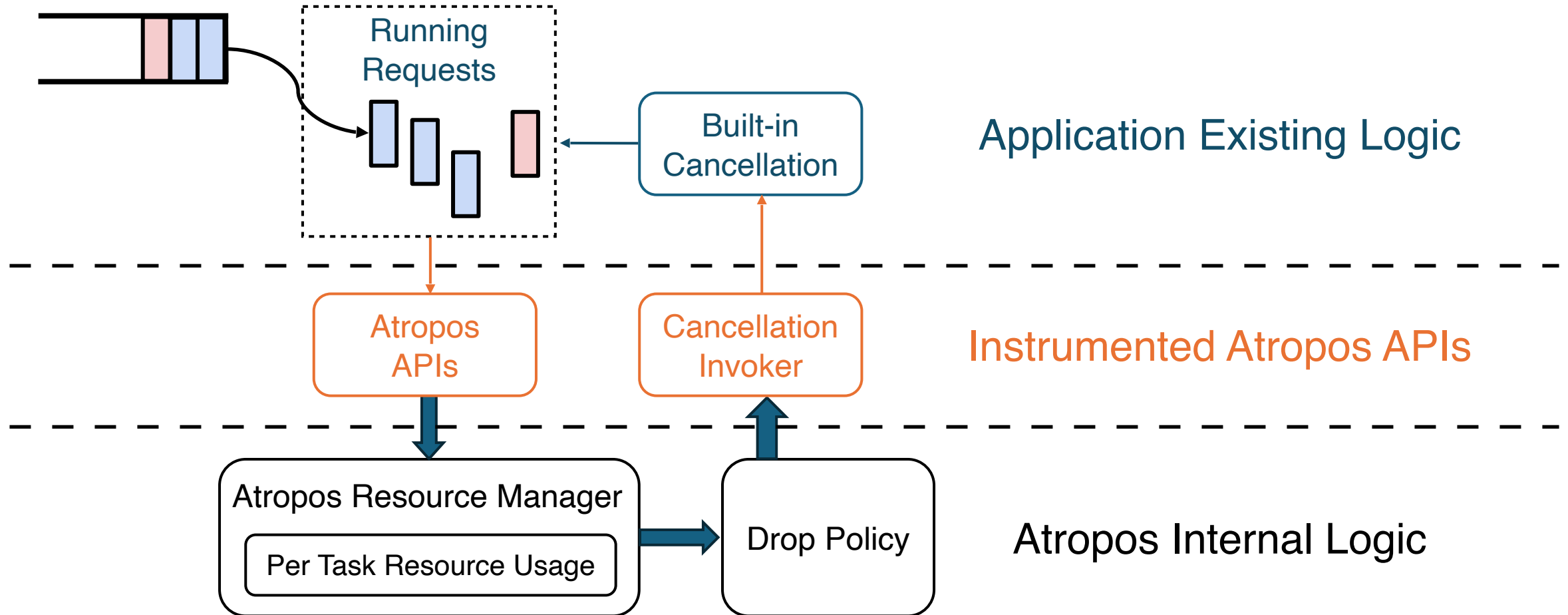
Regular Request

Monitor Requests

- **How to cover diverse application-defined resources?**
 - **Support 3 kinds of abstract resources**
 - **Provide trace APIs to instrument each kind of resources**
- **How to identify **the resource** that is overloaded and **the request** that could release the most load on the resource?**
 - **Propose 2 metrics to quantify overload level and resource gain**

For details, please refer to our paper

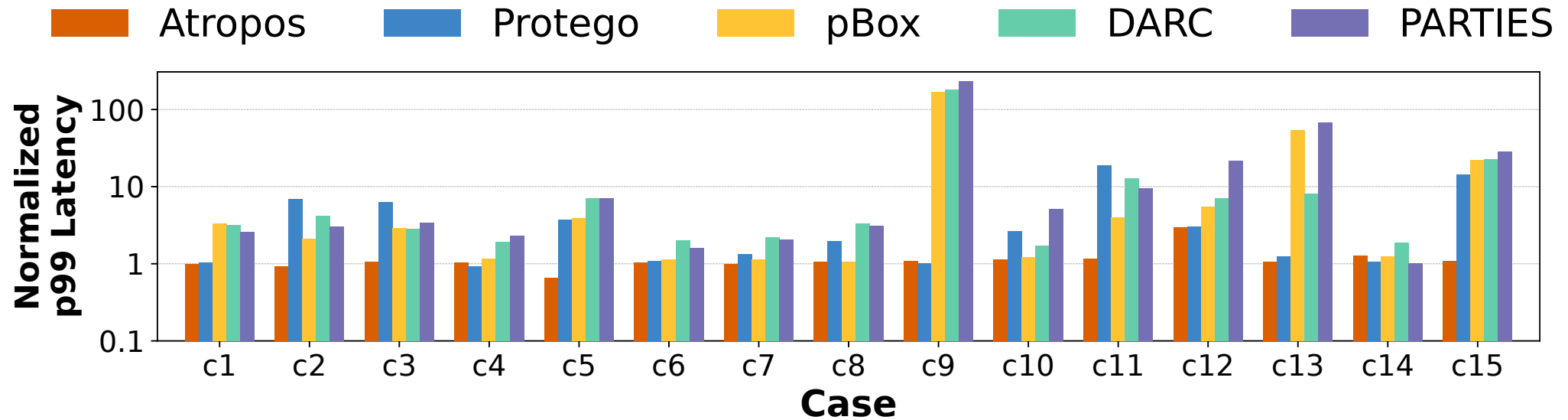
Design of Atropos



Evaluation

- **We adapted Atropos to 6 applications across 3 languages**
 - **MySQL, Postgresql, Apache, elasticsearch, solr, etcd**
 - **Adaptation effort: 49.5 LOC per application**
- **We evaluated Atropos with 15 real world cases**
 - **Noisy requests monopolizing resources degrade performance**
- **We compared Atropos with 4 SOTA baselines**
 - **Overload control: Protego**
 - **Performance isolation: pBox**

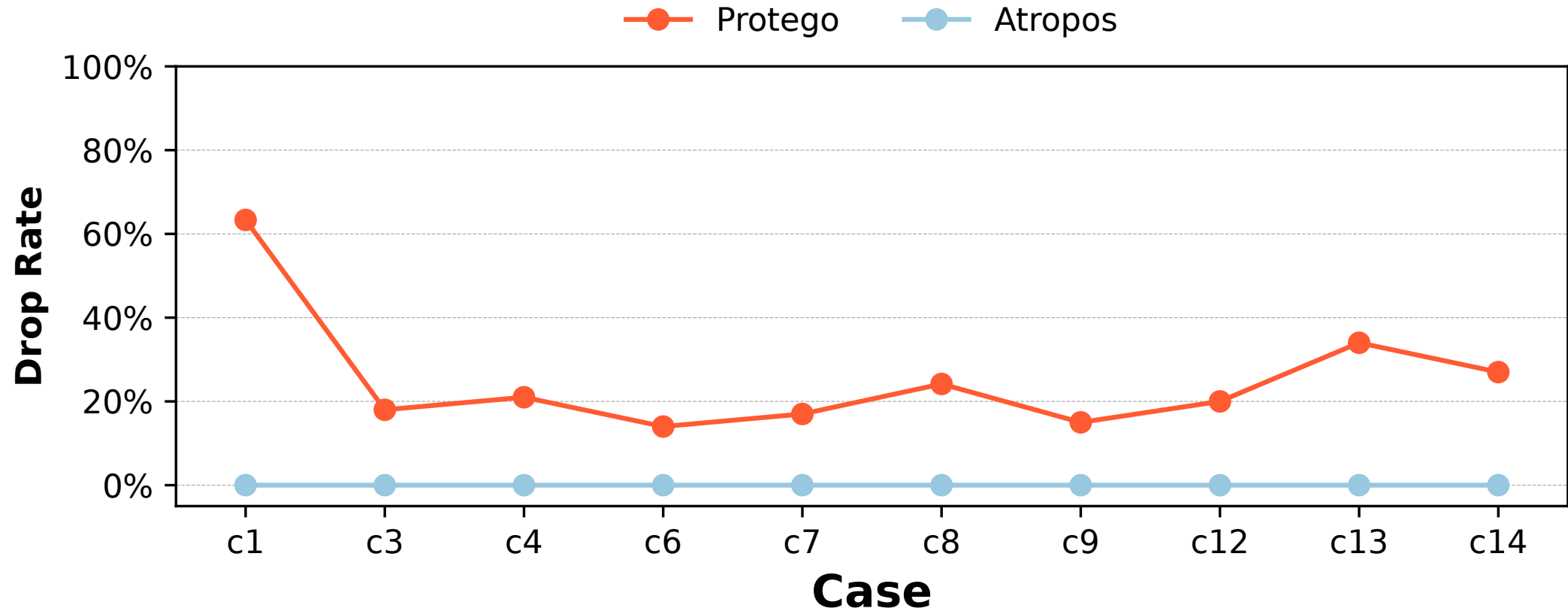
Reduction of Overload



Surpass all baselines on all cases in p99 latency and throughput*
Bounded p99 latency within 16% compared to none-overload case
The best baseline Protego is 88%

* Throughput result is in the paper

Request Drop Rate



Much lower drop rate compared to Protego

Conclusion

- **Problem:** Overload in application-defined resources is harmful to application performance but hard to prevent
- **Key insight:** Let requests run instead of dropping them before execution to precisely cancel the request that monopolizes resource
- **Key results:** Achieves lower tail latency and lower request drop rate

<https://github.com/OrderLab/Atropos>