6th International Conference on Cloud and Big Data Computing.
Birmingham City University, Birmingham, United Kingdom August 18-20, 2022.

Cost-Efficient and Latency-Aware Event Consuming in Workload-Skewed Distributed Event Queues

Mazen Ezzeddine[1,2], Gael Migliorini[2], Françoise Baude[1], Fabrice Huet[1]
[1]Université Côte d'Azur, CNRS, I3S
Nice, France
[2]HighTech Payment Systems, HPS
Aix En Provence, France
mazen.ezzeddine@univ-cotedazur.fr

Motivation and Problem statement (1)

- **Distributed Event Queues :** a central component in building modern large-scale and real-time cloud applications.
  - Recording and analyzing web accesses for recommendations and ad placement, fraud detection, and health care monitoring etc.
  - Cloud providers (AWS, GCP, Azure) already offer distributed event queue as a service

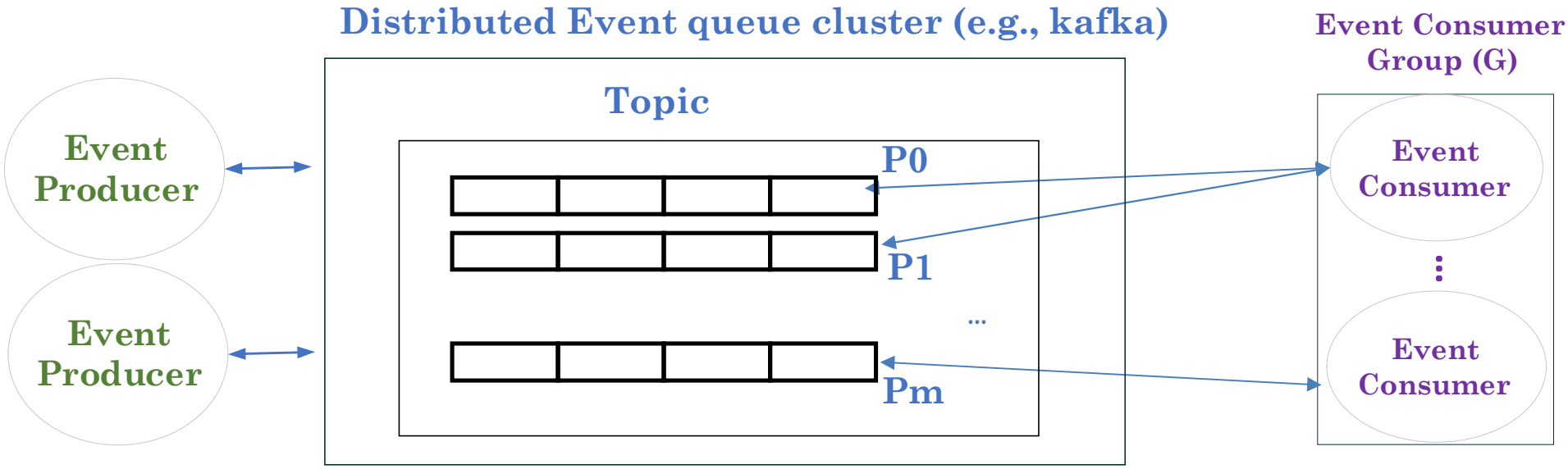
- **Latency-aware and cost-efficient event consuming**
  - **Guarantee *maximum event processing latency* (WSLA) for high percentile of events, under low cost in terms of event consumer replicas**
    - **Autonomous scale in and out of event consumers replicas to cater to the incoming arrival rate of events**

Motivation and Problem statement (2)

- Cloud providers offer solutions for ***autoscaling*** event consumers in distributed Event Queues. **Two limitations :**
  - ***Threshold based autoscalers***
  - ***Not designed with workload skewness in mind***
    - **Load aware and fair assignment** of events to event consumers according to event arrivals per event key.

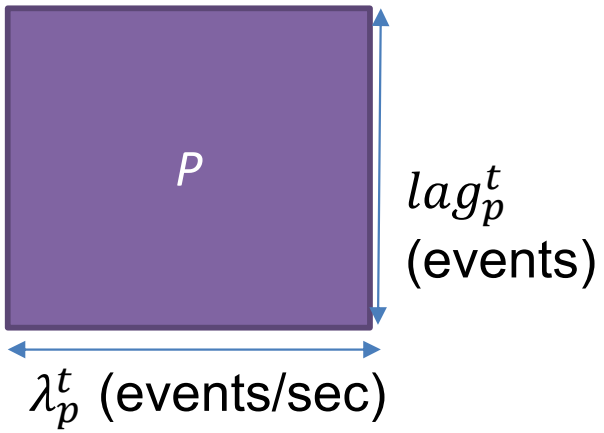
- This work : *A framework and methodology for scaling event consumers of a distributed event queue*
  1. ***Guarantee a maximum event processing latency W_{SLA} for high percentile of events***
  2. ***At low cost (in terms of number of event consumer replicas used)***
  3. ***Workload skewness support***

Context and Background (1)



Partition p

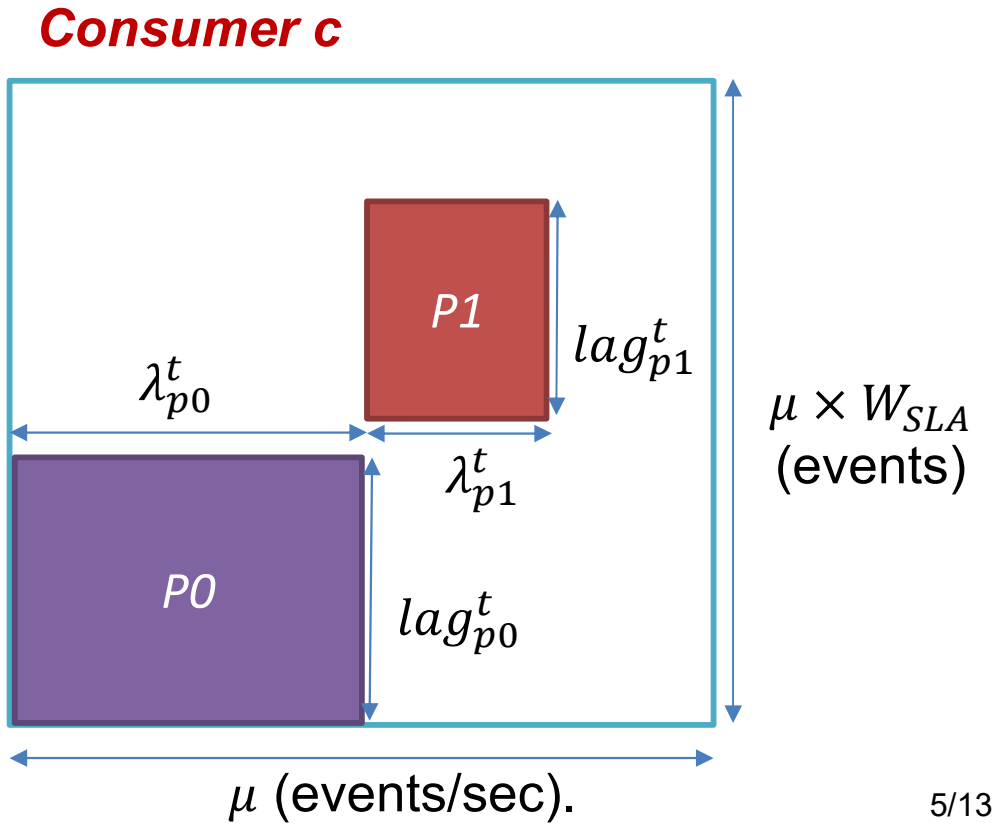
- λ_p^t = arrival rate of events at time t (events/sec)
- lag_p^t = number of events in the partition backlog at time t



Event Consumer C

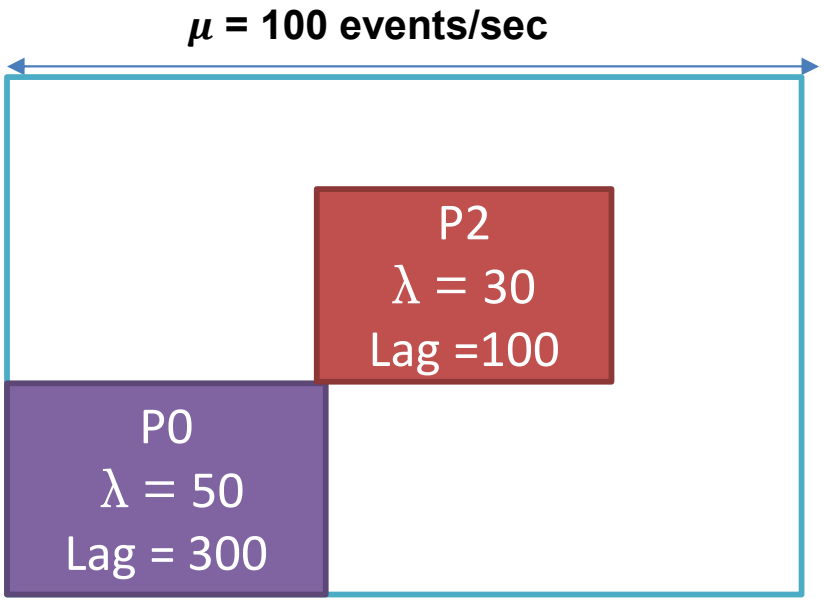
- μ = service rate (events/sec).
  - homogeneous consumers
- $\mu \times W_{SLA}$ = Maximum allowable events backlog of the consumer without violating the W_{SLA} latency.

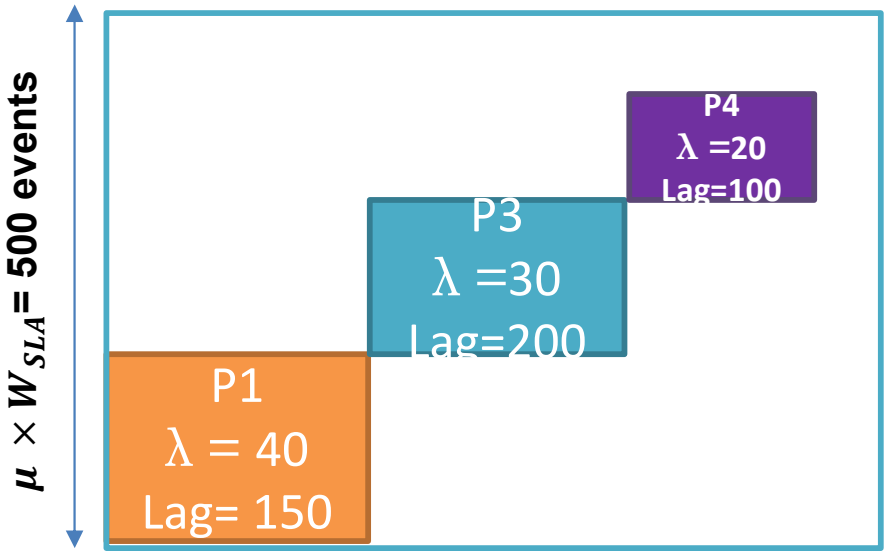
- λ_c^t = arrival rate into the consumer
- lag_c^t= number of events in the backlog of the consumer



Problem formulation

- What is the minimum number of consumers needed at time t, so that all the events will be processed in less than the maximum event processing latency W_{SLA}
  - $min|G^t|\ such\ that\ \ \forall\, c_j \in G^t,\ \ lag_{c_j}^t < \mu \times W_{SLA}\ AND\ \lambda_{c_j}^t < \mu$
    - $|G^t|$ minimal number of event consumers needed at time t so that W_{SLA} (maximum event processing latency) is respected
    - Integer Linear Programming (ILP) formulation in the paper.

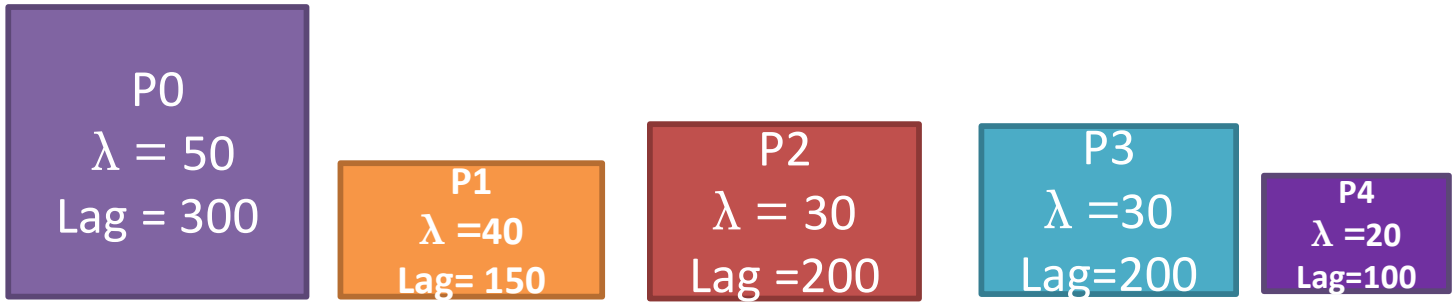


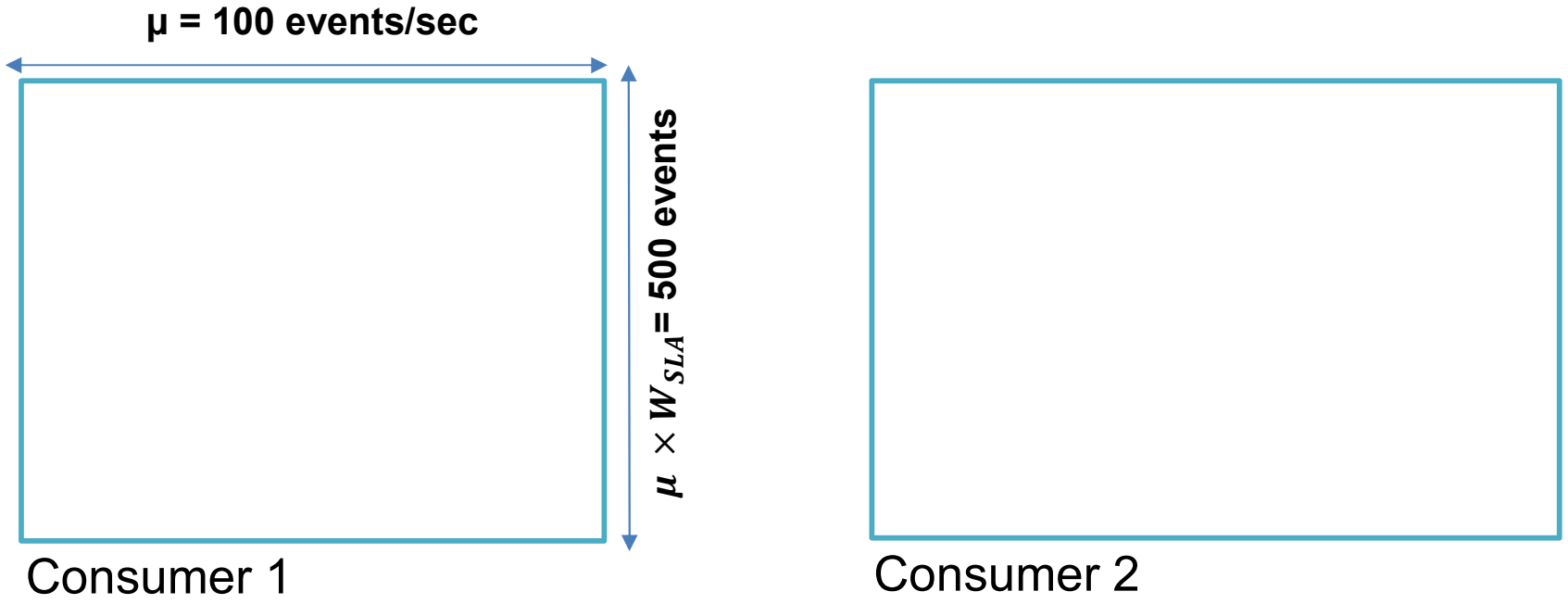


Bin Pack heuristic : Least Loaded First Fit Decreasing LLFFD

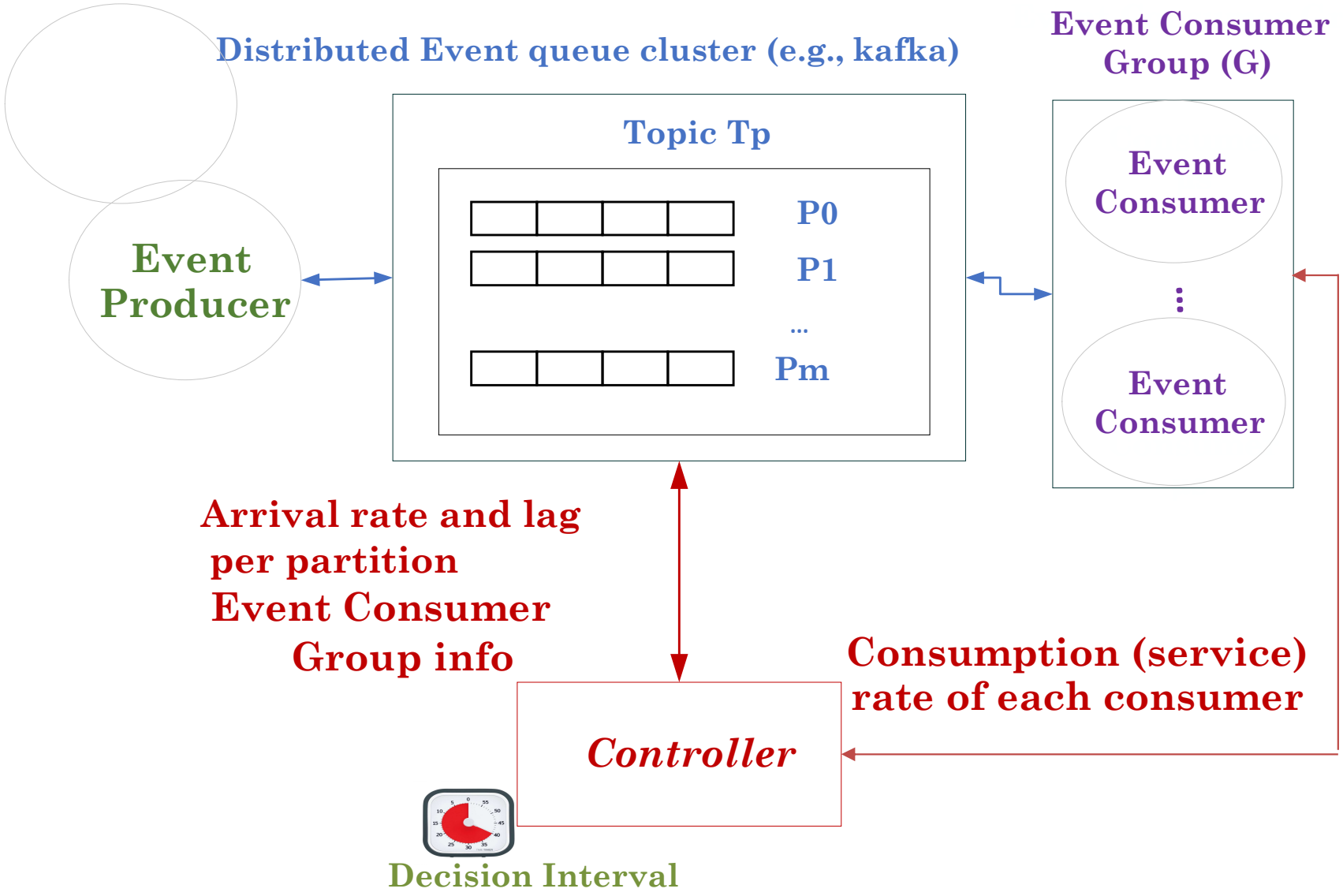
- Used the bin pack **heuristic Least Loaded First Fit Decreasing LLFFD**
- **LLFFD** was proposed to **pack VMs to pyhiscal servers** in the datacenter
  - Guarantees a load balance across the bins (event consumers)
  - Detailled example next slide.

Least Load First Fit Decreasing





System Architecture



Controller Autoscale Logic

AutoScaleCG (λ_P^t, lag_P^t ,W_{SLA})

Set $|G^{t-1}|$ to the existing set of event consumers

Set $|G^t|$ = *Least-Load FirstFitDecreasing* (λ_P^t, lag_P^t , W_{SLA})

IF $|G^t| > |G^{t-1}|$

Scale *up* by $|G^t| \setminus |G^{t-1}|$ // *implicitly will trigger rebalance/reassignment*

ELSE IF $|G^t| < |G^{t-1}|$

Scale down by $|G^{t-1}| \setminus |G^t|$ //*implicitly will trigger rebalance/reassignment*

ELSE // $G^{t-1} = G^t$

IF *currentAssignmentDoesNotViolateTheSLA*()

return

ELSE

Trigger a rebalance/reassignment

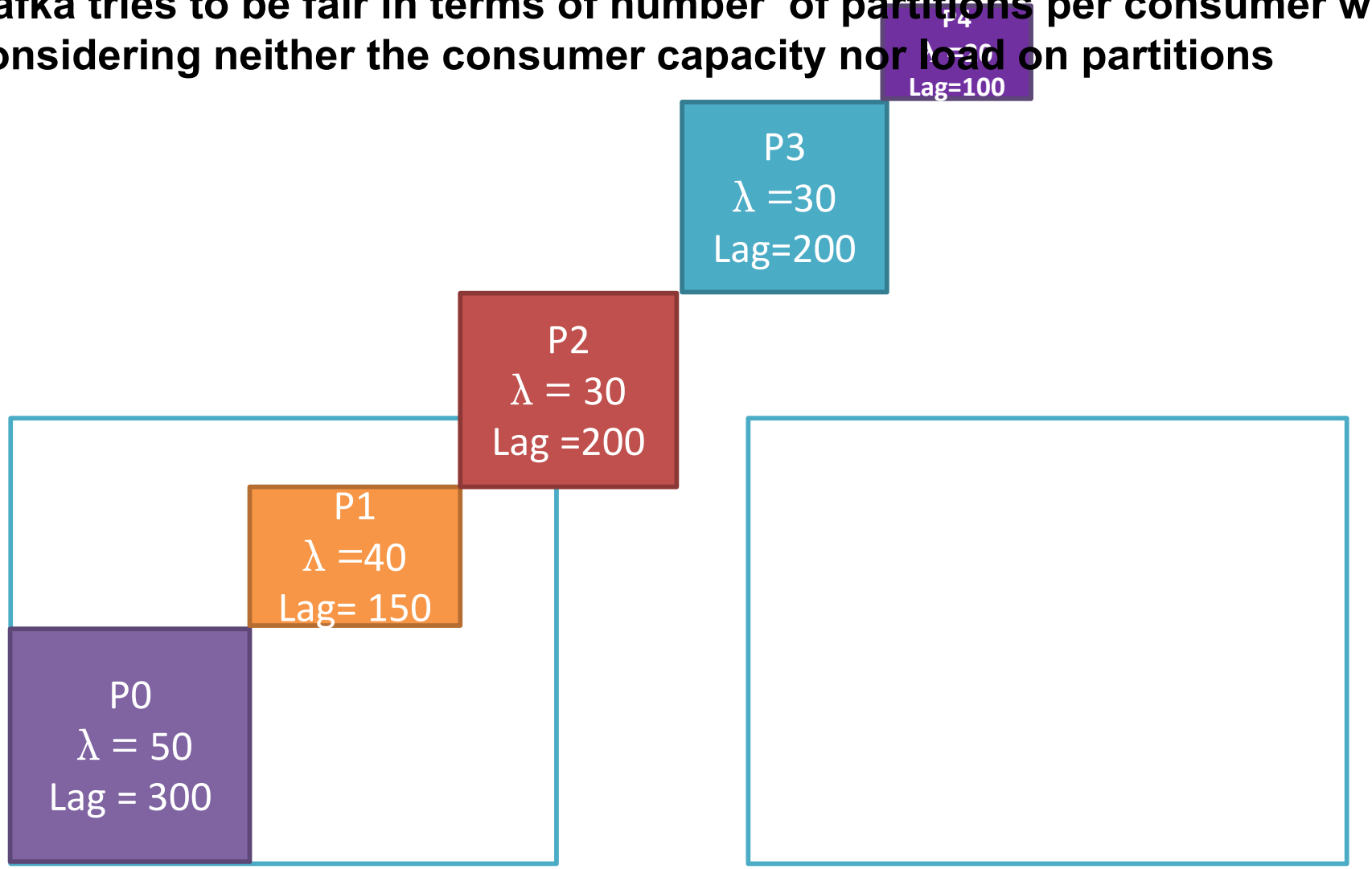
END IF

END IF

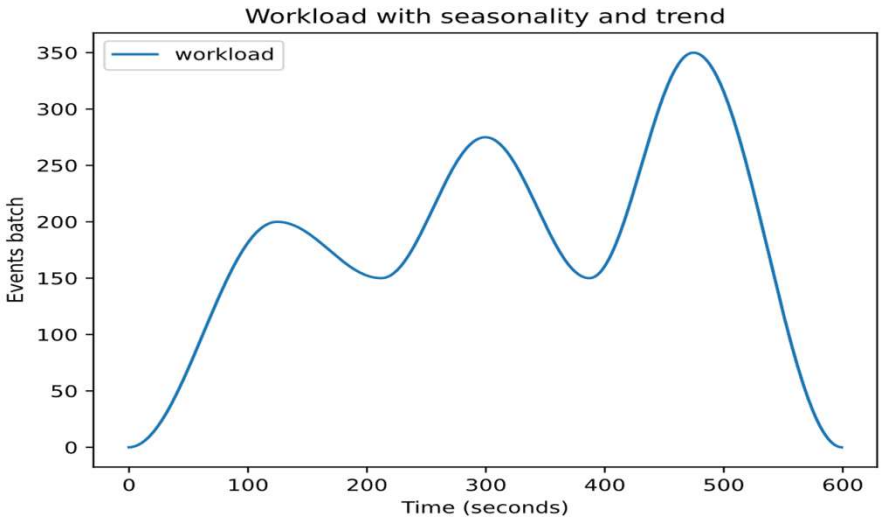
G^t : minimal number of consumers needed at time t, and the assignement of these consumers to partitions

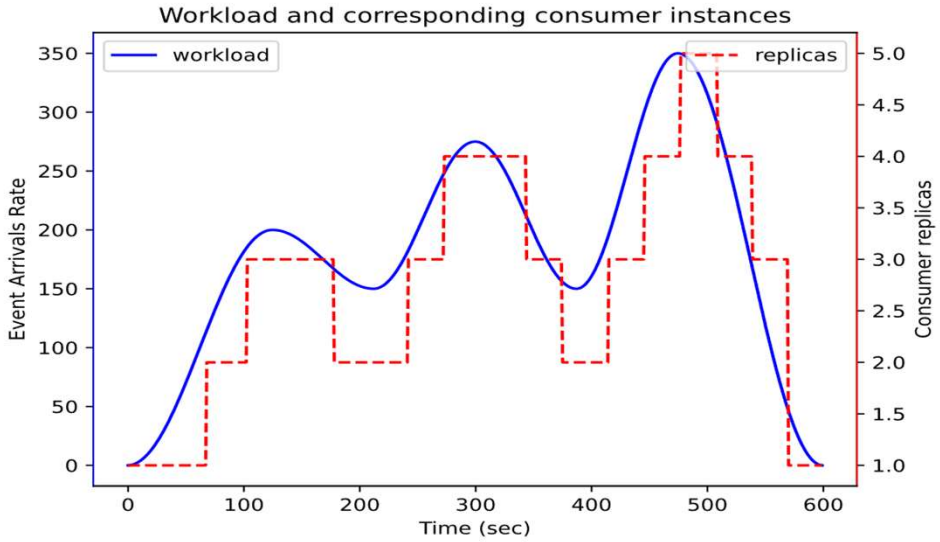
Incremental Cooperative Rebalancing (Kafka recommended/default)

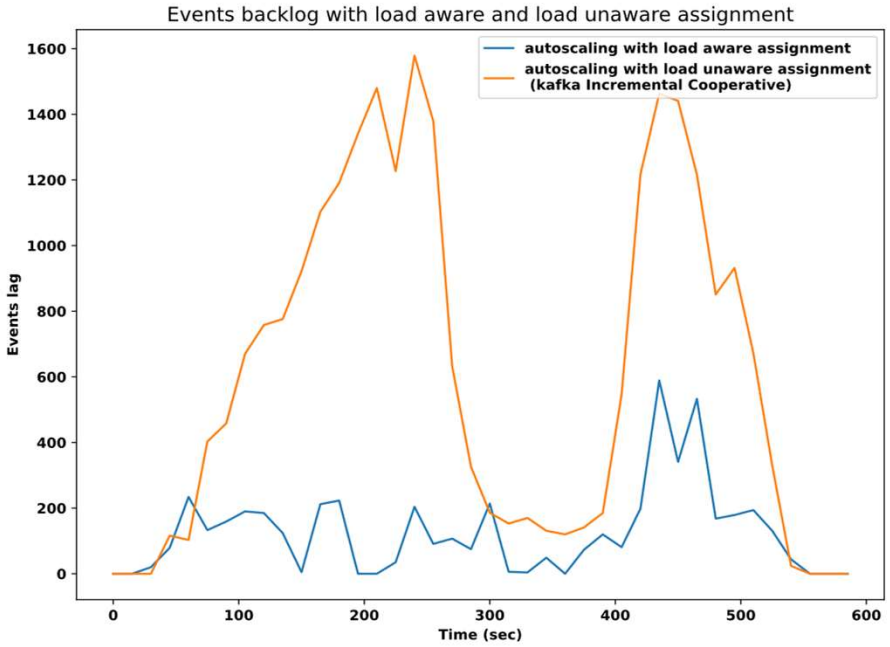
Kafka tries to be fair in terms of number of partitions per consumer without considering neither the consumer capacity nor load on partitions



Experimental Work







W_{SLA} = 5 seconds
Decision interval = 1 second

Autoscaling with load-aware assignement
100% latency guarantee.
26.1 replica.minutes.

Autoscaling with load-unaware assignement
(kafka incremental cooperative assignement)
77% latency guarantee
26.1 replica.minutes.

Conclusion and Perspectives

- **A framework for Latency-aware and cost-efficient event consuming from distributed event queue**
  - Support for workload skewness
  - Complemented with load-aware partitions-consumers assignment

- **In distributed Event queues, non-load aware (Incremental Cooperative) autoscaling strategy is not optimal for latency guarantee**
  - Much less latency guarantee under same scaling actions and number of replicas as that of a load-aware autoscaling

- More design space exploration under larger scale deployment
  - The heterogenous event consumers case.