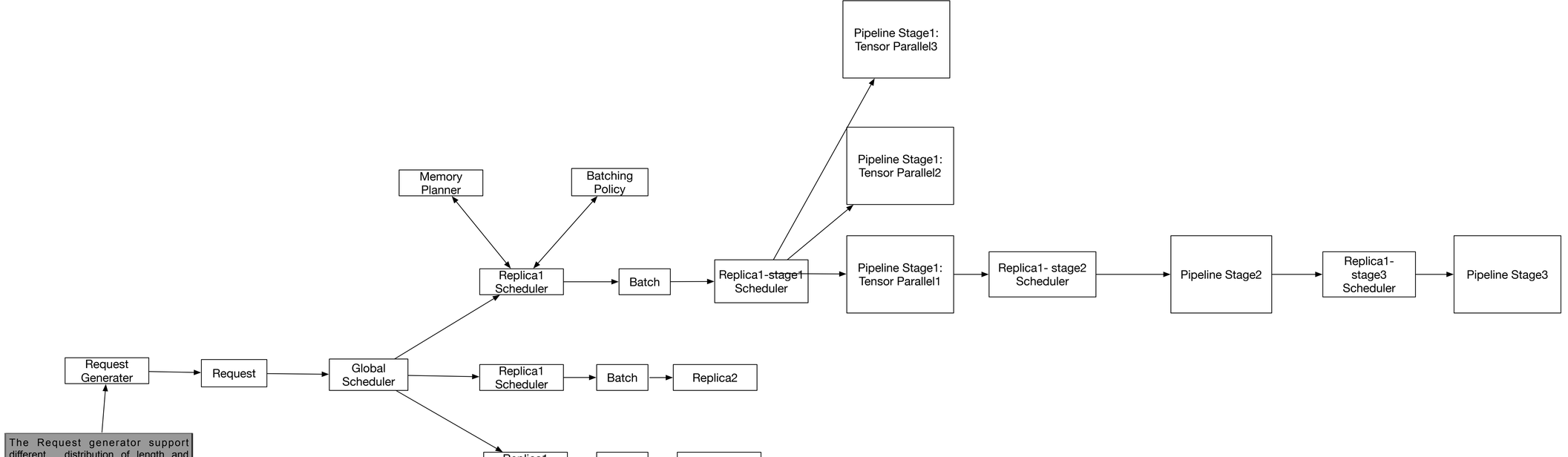
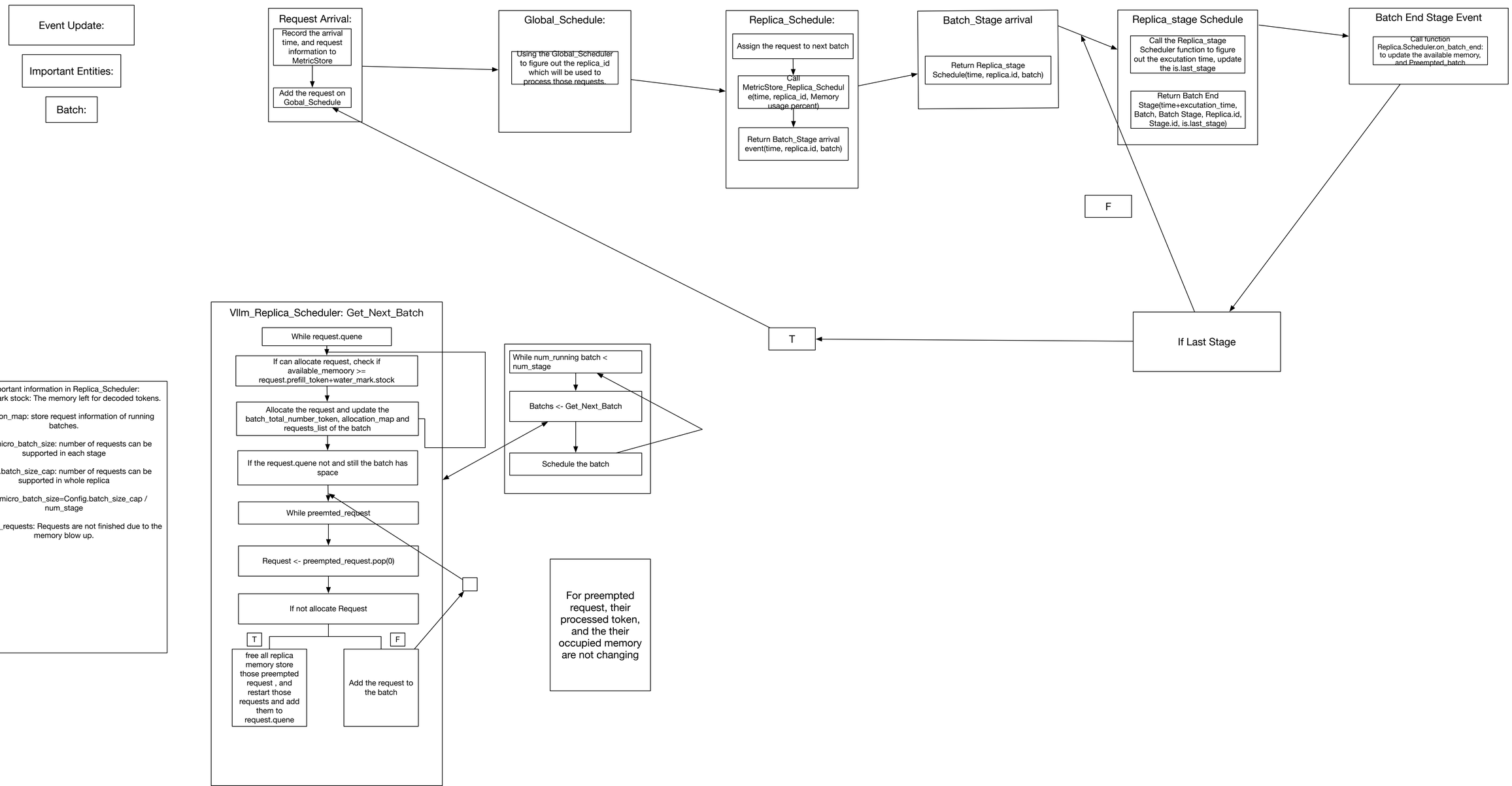

Pipeline Stage1: Tensor Parallel3
Pipeline Stage1: Tensor Parallel2
Memory Planner
Batching Policy
Pipeline Stage1: Tensor Parallel1
Replica1 Scheduler
Batch
Replica1-stage1 Scheduler
Replica1- stage2 Scheduler
Pipeline Stage2
Replica1- stage3 Scheduler
Pipeline Stage3
Request Generater
Request
Global Scheduler
Replica1 Scheduler
Batch
Replica2
Replica1 Scheduler
Batch
Replica3
The Request generator support different distribution of length and interval for Requests.


Event Update:
Important Entities:
Batch:
Request Arrival:
Record the arrival time, and request information to MetricStore
Add the request on Gobal_Schedule
Global_Schedule:
Using the Global_Scheduler to figure out the replica_id which will be used to process those requests.
Replica_Schedule:
Assign the request to next batch
Call MetricStore_Replica_Schedule(time, replica_id, Memory usage percent)
Return Batch_Stage arrival event(time, replica.id, batch)
Batch_Stage arrival
Return Replica_stage Schedule(time, replica.id, batch)
Replica_stage Schedule
Call the Replica_stage Scheduler function to figure out the excutation time, update the is.last_stage
Return Batch End Stage(time+excutation_time, Batch, Batch Stage, Replica.id, Stage.id, is.last_stage)
Batch End Stage Event
Call function Replica.Scheduler.on_batch_end: to update the available memory, and Preempted_batch
F
T
If Last Stage
VIlm_Replica_Scheduler: Get_Next_Batch
While request.quene
If can allocate request, check if available_memoory >= request.prefill_token+water_mark.stock
Allocate the request and update the batch_total_number_token, allocation_map and requests_list of the batch
If the request.quene not and still the batch has space
While preemted_request
Request <- preempted_request.pop(0)
If not allocate Request
T
F
free all replica memory store those preempted request , and restart those requests and add them to request.quene
Add the request to the batch
While num_running batch < num_stage
Batchs <- Get_Next_Batch
Schedule the batch
Important information in Replica_Scheduler:
Water_mark stock: The memory left for decoded tokens.
Allocation_map: store request information of running batches.
Max_micro_batch_size: number of requests can be supported in each stage
Config.batch_size_cap: number of requests can be supported in whole replica
Max_micro_batch_size=Config.batch_size_cap / num_stage
Preemted_requests: Requests are not finished due to the memory blow up.
For preempted request, their processed token, and the their occupied memory are not changing