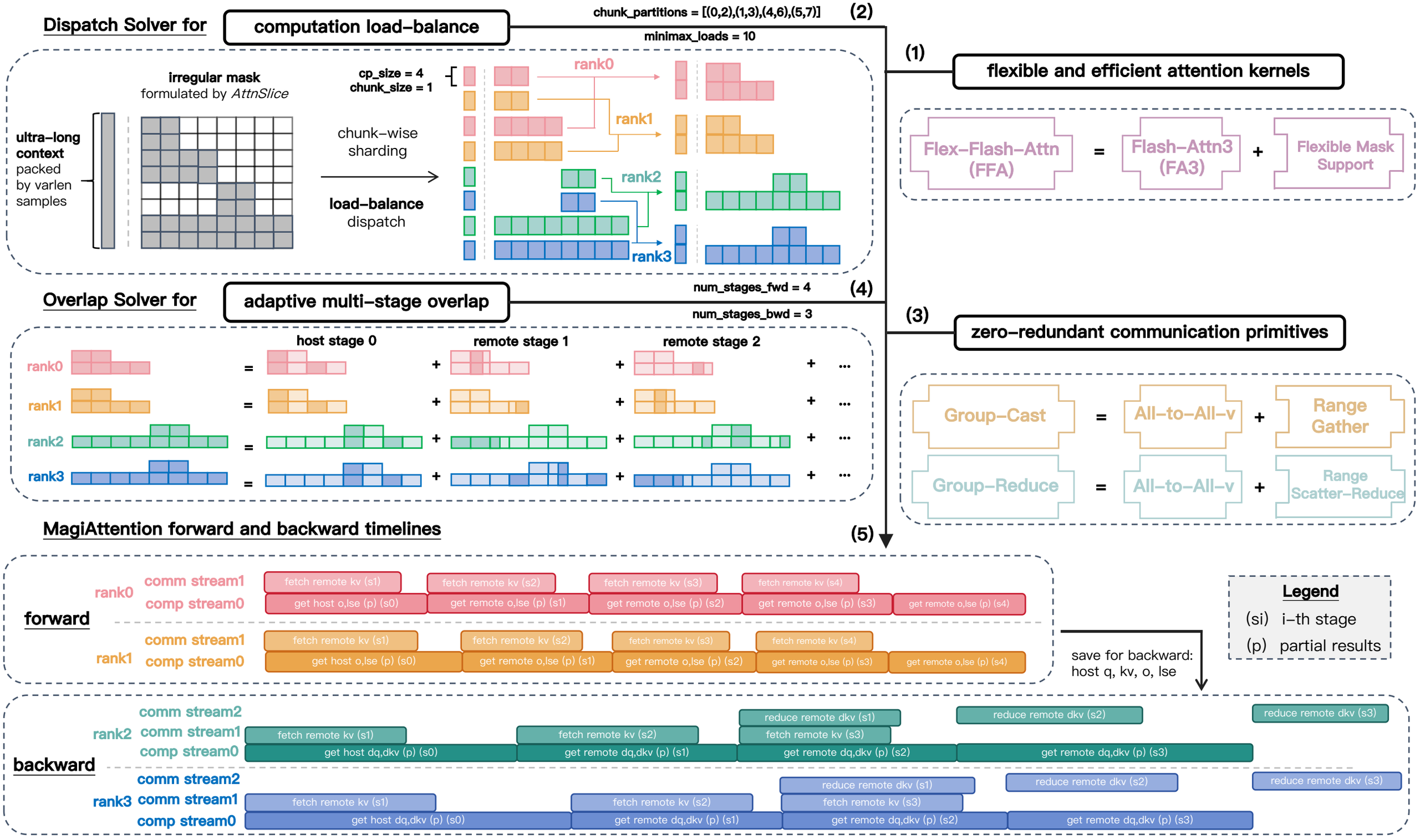
Dispatch Solver for
computation load-balance
chunk_partitions = [(0,2),(1,3),(4,6),(5,7)]
minimax_loads = 10
(2)
(1)
flexible and efficient attention kernels
irregular mask
formulated by AttnSlice
cp_size = 4
chunk_size = 1
ultra-long context packed by varlen samples
chunk-wise sharding
load-balance dispatch
rank0
rank1
rank2
rank3
Flex-Flash-Attn (FFA)
=
Flash-Attn3 (FA3)
+
Flexible Mask Support
Overlap Solver for
adaptive multi-stage overlap
num_stages_fwd = 4
num_stages_bwd = 3
(4)
(3)
zero-redundant communication primitives
host stage 0
remote stage 1
remote stage 2
Group-Cast
=
All-to-All-v
+
Range Gather
Group-Reduce
=
All-to-All-v
+
Range Scatter-Reduce
MagiAttention forward and backward timelines
(5)
forward
comm stream1
comp stream0
fetch remote kv (s1)
fetch remote kv (s2)
fetch remote kv (s3)
fetch remote kv (s4)
get host o,lse (p) (s0)
get remote o,lse (p) (s1)
get remote o,lse (p) (s2)
get remote o,lse (p) (s3)
get remote o,lse (p) (s4)
save for backward:
host q, kv, o, lse
Legend
(si) i-th stage
(p) partial results
backward
comm stream2
comm stream1
comp stream0
reduce remote dkv (s1)
reduce remote dkv (s2)
reduce remote dkv (s3)
fetch remote dq,dkv (p) (s0)
fetch remote dq,dkv (p) (s1)
fetch remote dq,dkv (p) (s2)
fetch remote dq,dkv (p) (s3)
get host dq,dkv (p) (s0)
get remote dq,dkv (p) (s1)
get remote dq,dkv (p) (s2)
get remote dq,dkv (p) (s3)