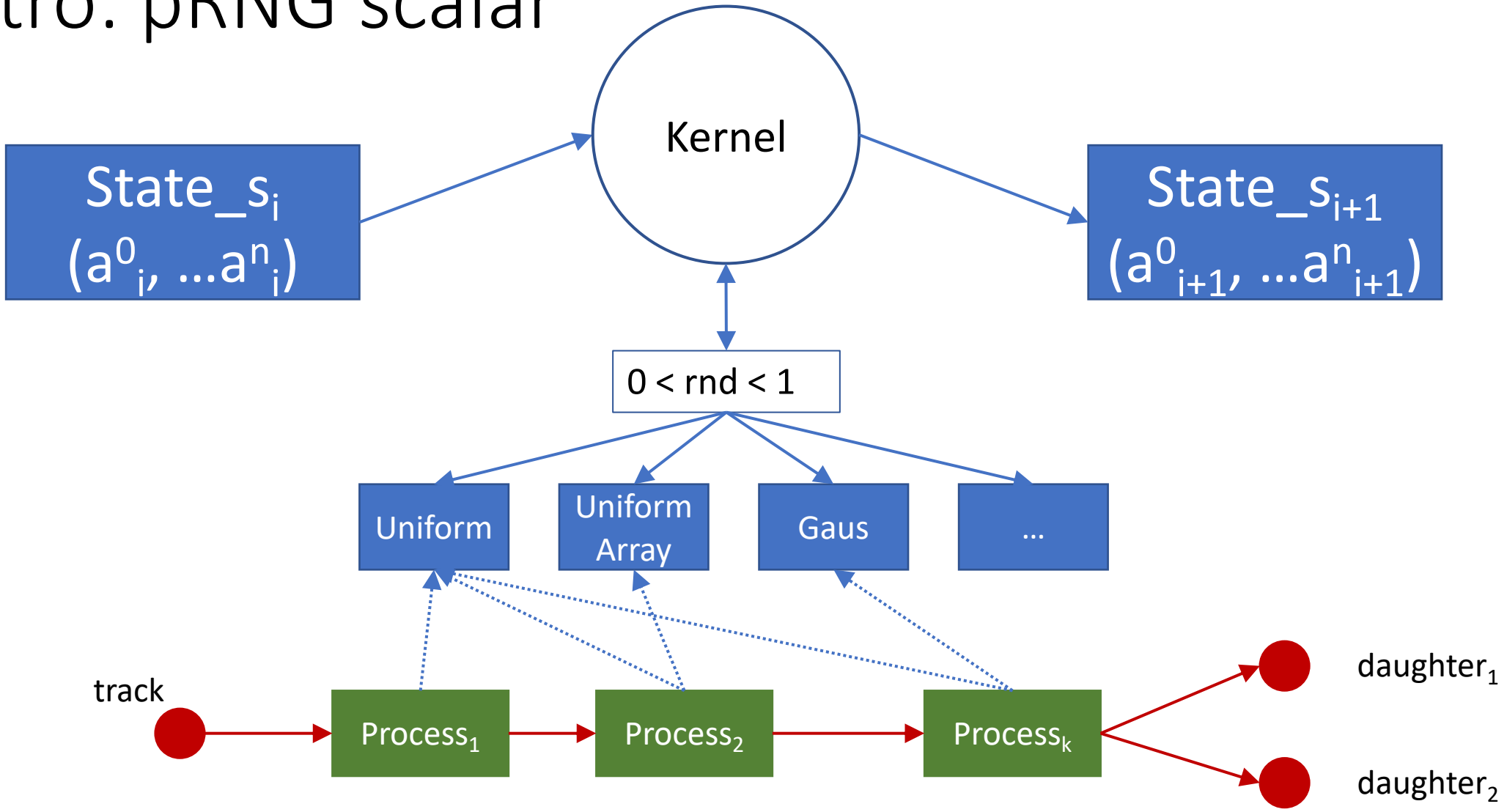
PRNG reproducibility

in multi-thread & scalar & multi-track (basket) case

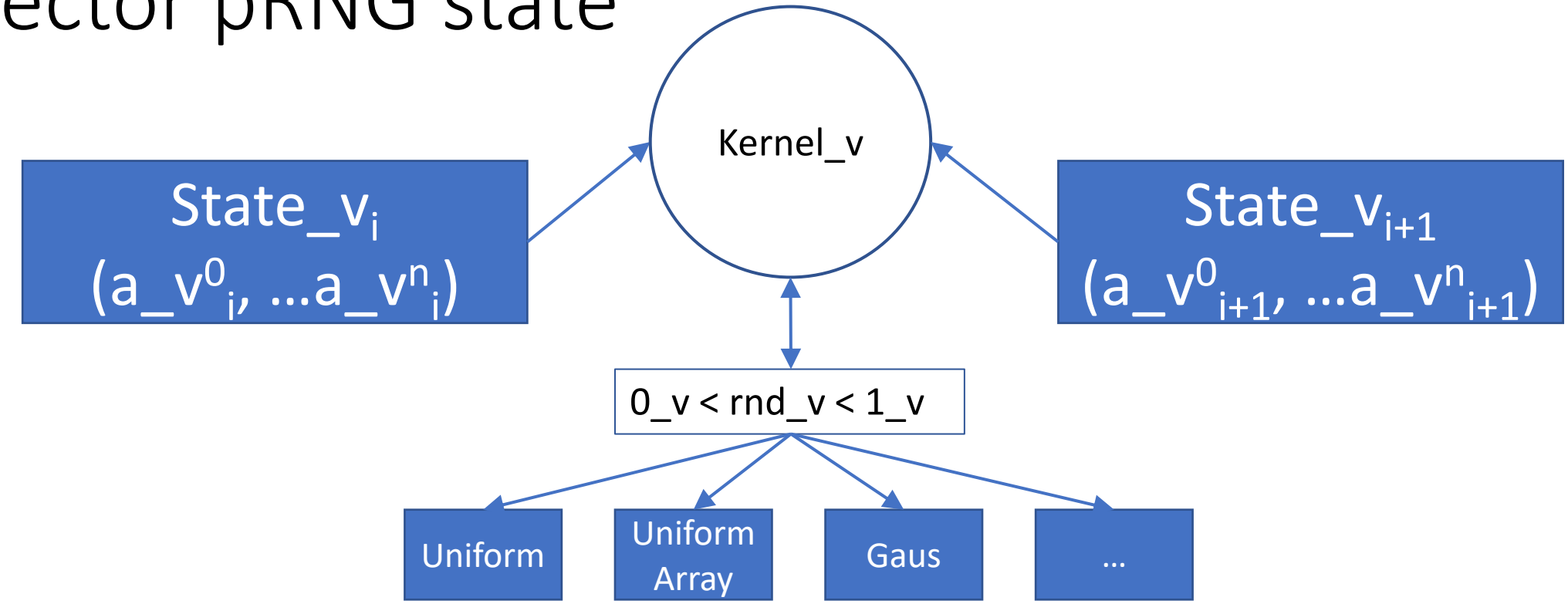
Intro: pRNG scalar



Reproducibility

- Conditions
  - Starting the random sequence from the same seed (state)
  - Preserving the original track sequencing (e.g. basketized vs. non-basketized)
- A global pRNG state is enough if conditions above fulfilled
- Multi-threading and/or basketization change track sequencing for calls to pRNG
  - Need to make the pRNG state intrinsically coupled with the track state, not exposing its evolution to sequencing

Vector pRNG state



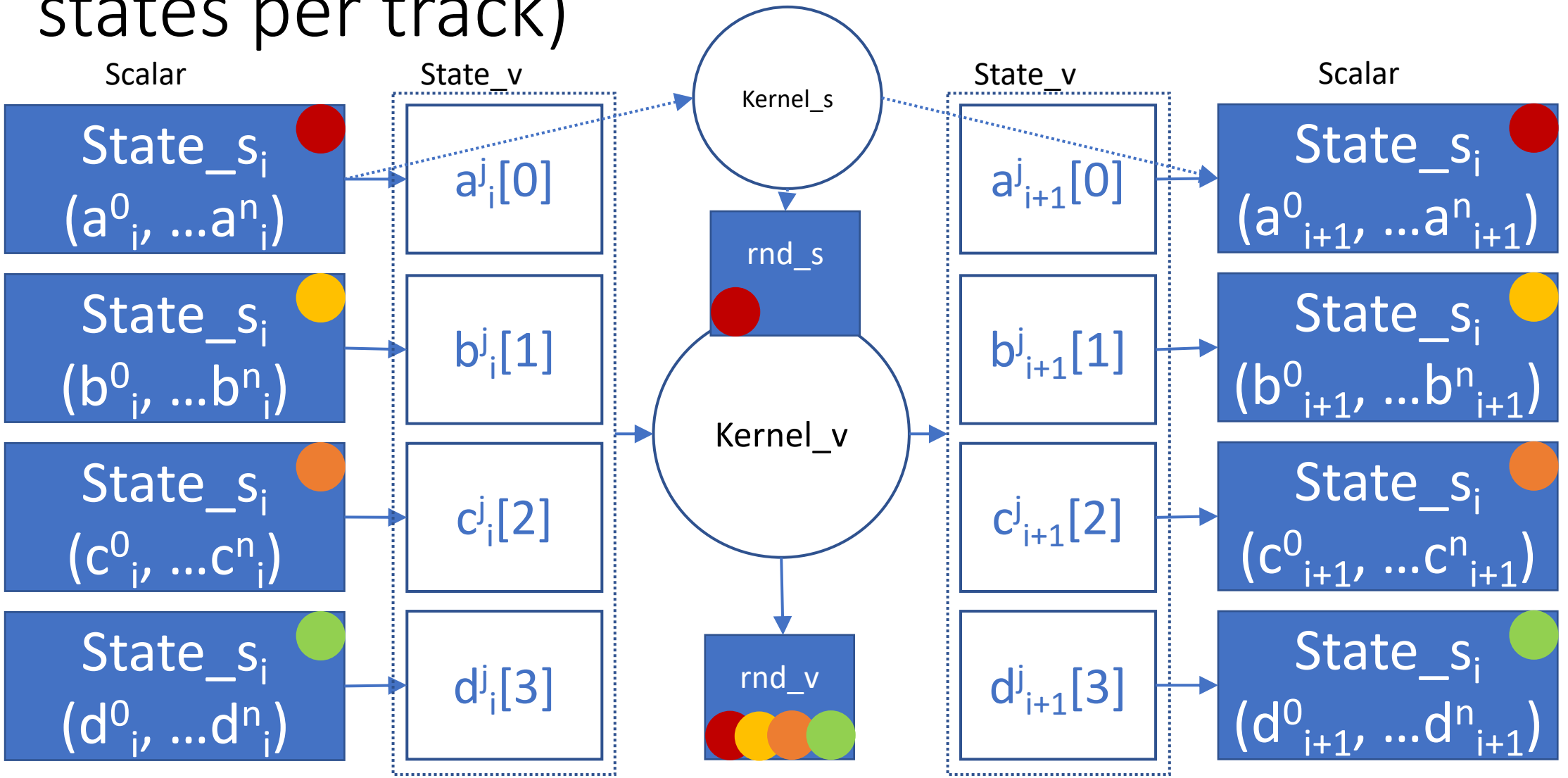
where: $a\_v^0_i$

AVX double (n_{lanes}= 4)

$a^0_i[0]$	$a^0_i[1]$	$a^0_i[2]$	$a^0_i[3]$

A vector State_v contains n_{lanes} scalar states

Vectorized multi-track approach (scalar pRNG states per track)



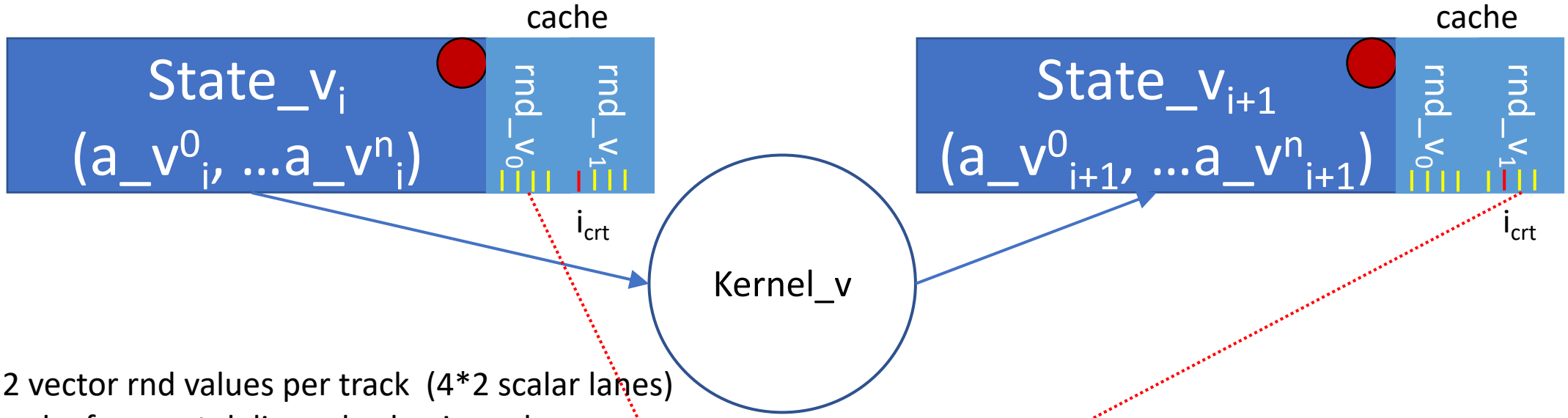
Gather scalar states to individual State_v lanes

Scatter lanes into original track scalar states

Remarks

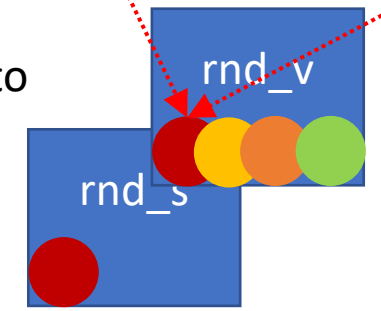
- Easy way to implement the proxy
- Memory footprint: N_{tracks}*sizeof(State_s)
- 4*sizeof(State) bytes copied per track
  - track state -> proxy memory -> vector register -> proxy memory -> track state
- Rejection has to be called in single track mode
  - Values computed in vector mode may be dropped for some lanes otherwise, compromising reproducibility
- Single track goes scalar

Vectorized caching approach (vector pRNG states per track)



Cache 2 vector rnd values per track (4*2 scalar lanes)
Keep track of current delivered value in cache

Basketized



Non-basketized

Gather scalar value from cached lanes into *rnd_s/rnd_v* in case: $(i_{crt} \% 4) > 0$

Call vector kernel to advance the state and refill the cache in case: $(i_{crt} \% 4) = 0$

Remarks

- More complicated implementation of the proxy
- Memory footprint: N_{tracks}*sizeof(State_v) + 2*sizeof(Double_v)
- 2*sizeof(State) + sizeof(double) bytes copied per track
  - track state -> vector register -> track state + 1 rnd cached
- All calls to the kernel done in vector mode
  - Run-time benefit in the scalar case
- More complex initialization per track
  - Need N_{lanes} sequences per track instead of one

Discussion

- The 2 approaches cannot be inter-changed during the same run
- There are benefits/overheads in the 2 approaches
- In my opinion we need an implementation for both types of proxies
- Where to implement the proxies?