Charades

An Adaptive Parallel Discrete Event Simulation Framework on Charm++

Eric Mikida

PPL UIUC

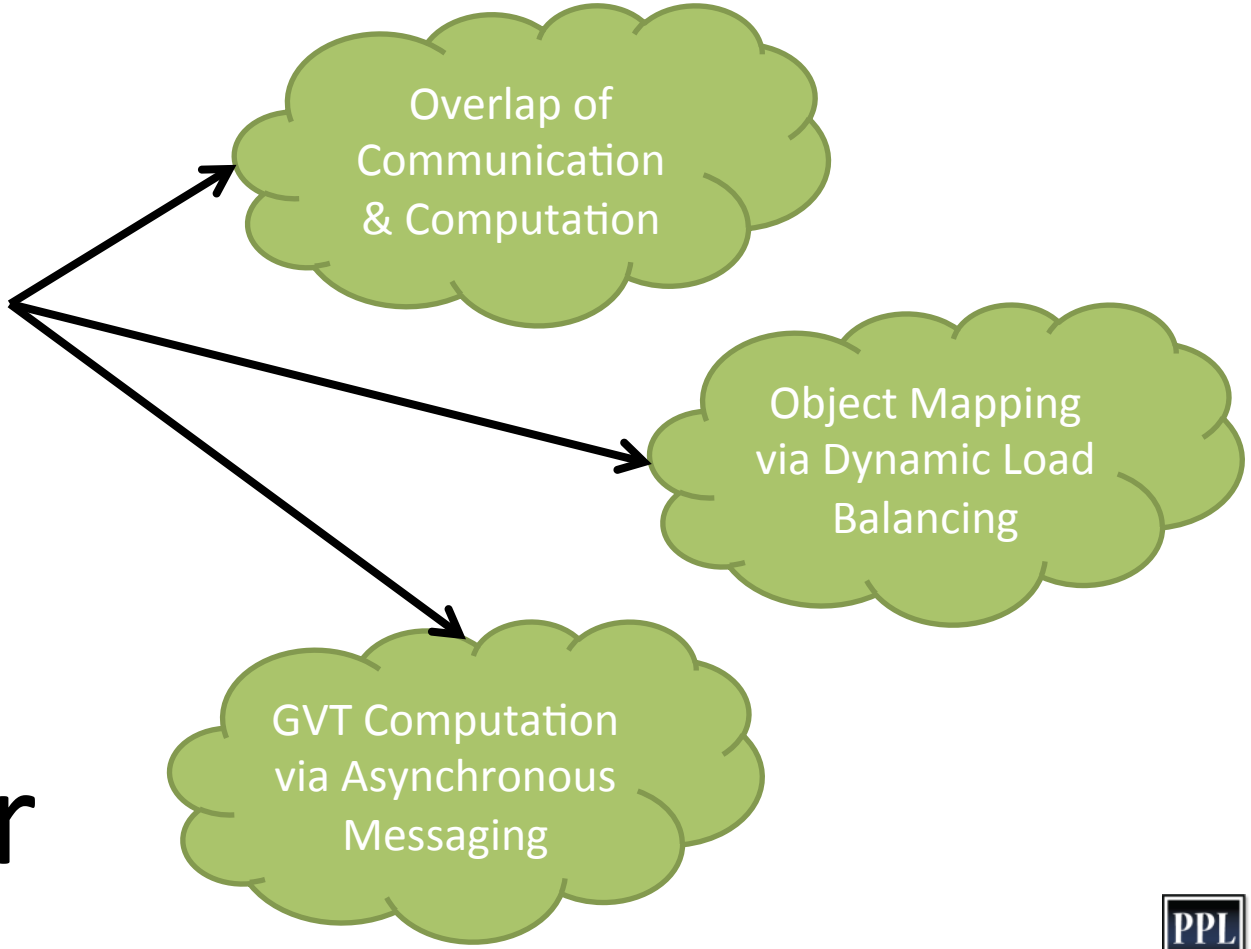
Charm++

Adaptive

Discrete

Event

Simulator



PPL UIUC

Brief PDES Description

- Simulation made up of Logical Processes (LPs)
- LPs process events in timestamp order
- Synchronization is conservative or optimistic
- Periodically compute global virtual time (GVT)

PPL UIUC

Performance Metrics

$$\text{Event Rate} = E_{committed} / s$$

$$\text{Event Efficiency} = E_{committed} / E_{total}$$

PPL UIUC

Performance Tuning Tradeoffs

- Shown good performance on benchmarks
  - We can execute/send large numbers of fine-grained events effectively
- How do we adapt to improve performance in the less ideal cases?
  - What events are we actually executing/sending

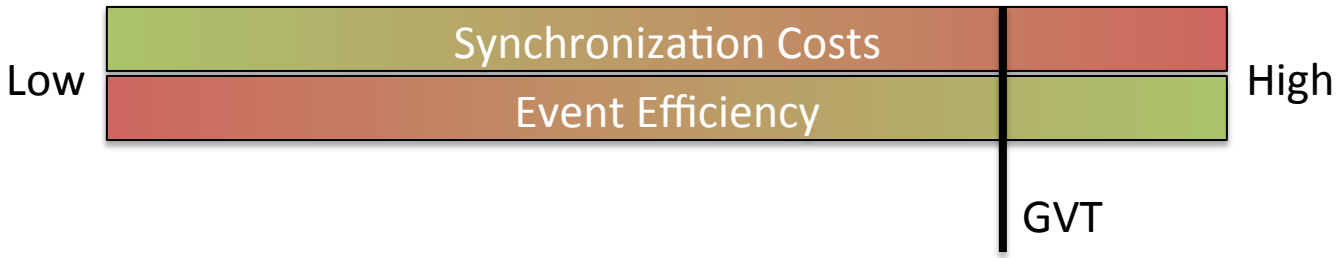
GVT Computation

- Global computation, required frequently
- Common solution blocks entirely during computation
  - Side effect of blocking is bounded-optimism
  - Leads to higher event efficiency

PPL UIUC

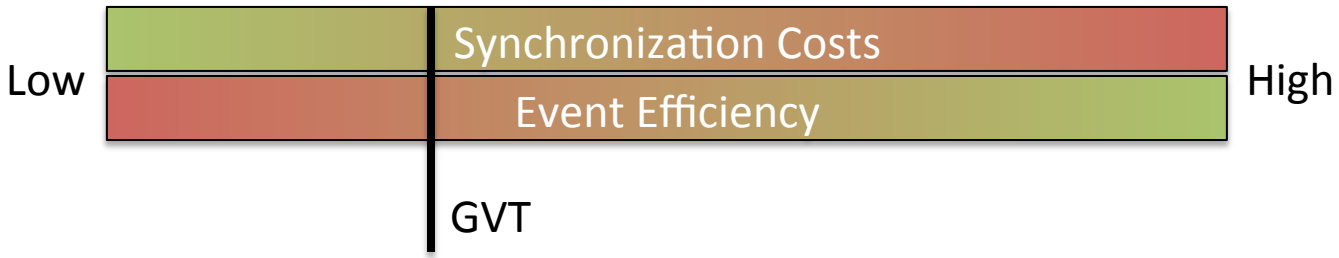
GVT Tradeoff

- One coupled tuning knob
- Common case is high synchronization



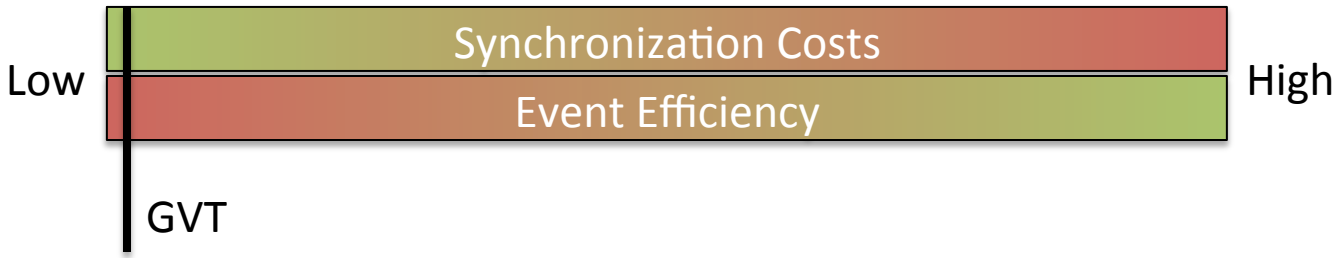
GVT Tradeoff

- One coupled tuning knob
- Common case is high synchronization
- As we lower synchronization, we lose efficiency

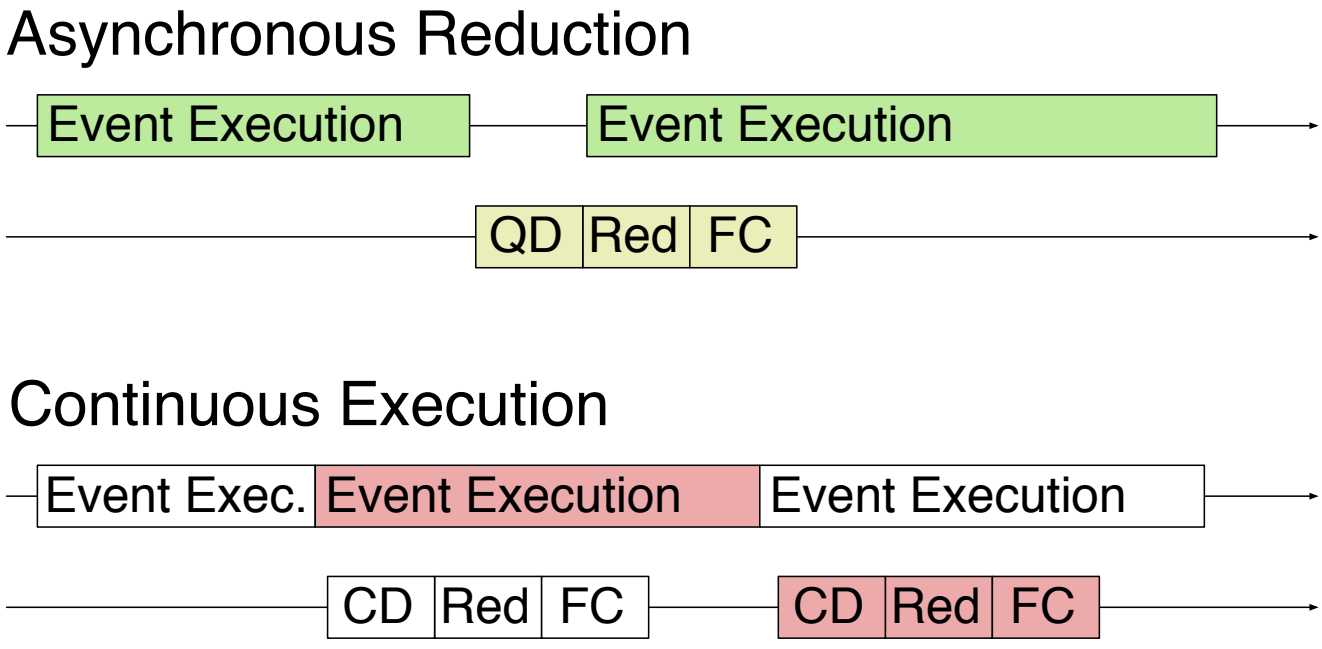


GVT Tradeoff

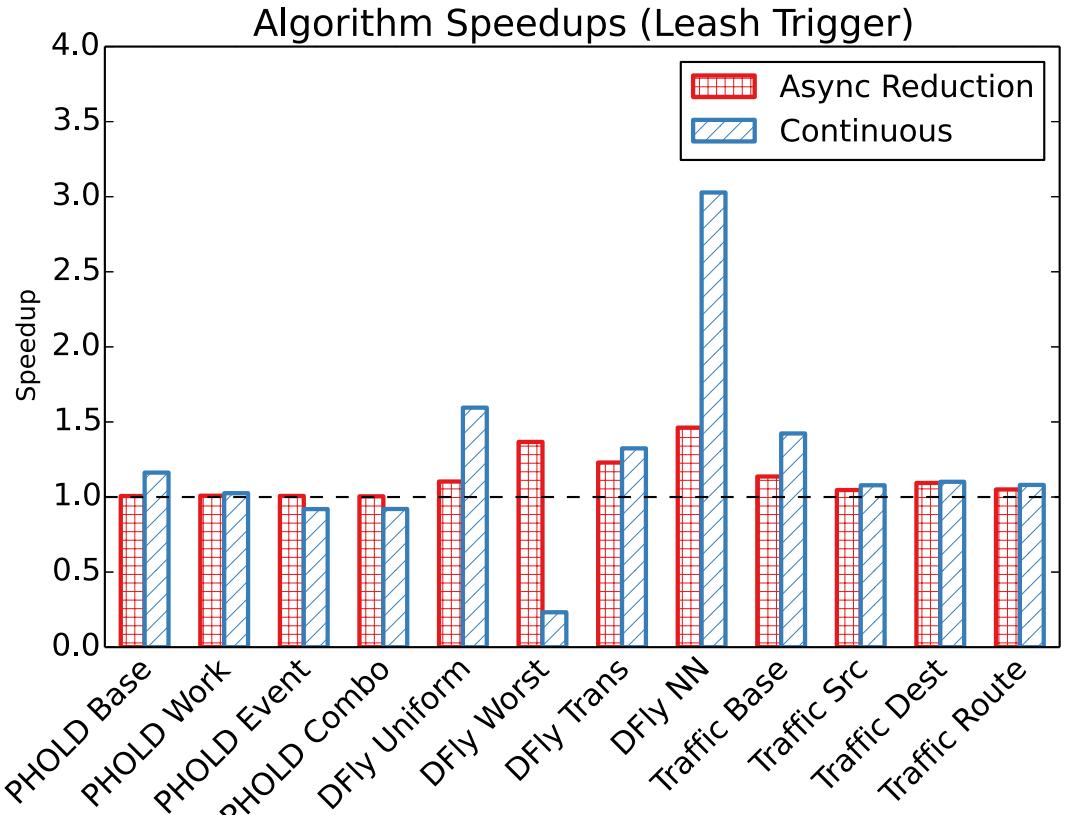
- One coupled tuning knob
- Common case is high synchronization
- As we lower synchronization, we lose efficiency



Reducing Synchronization Costs

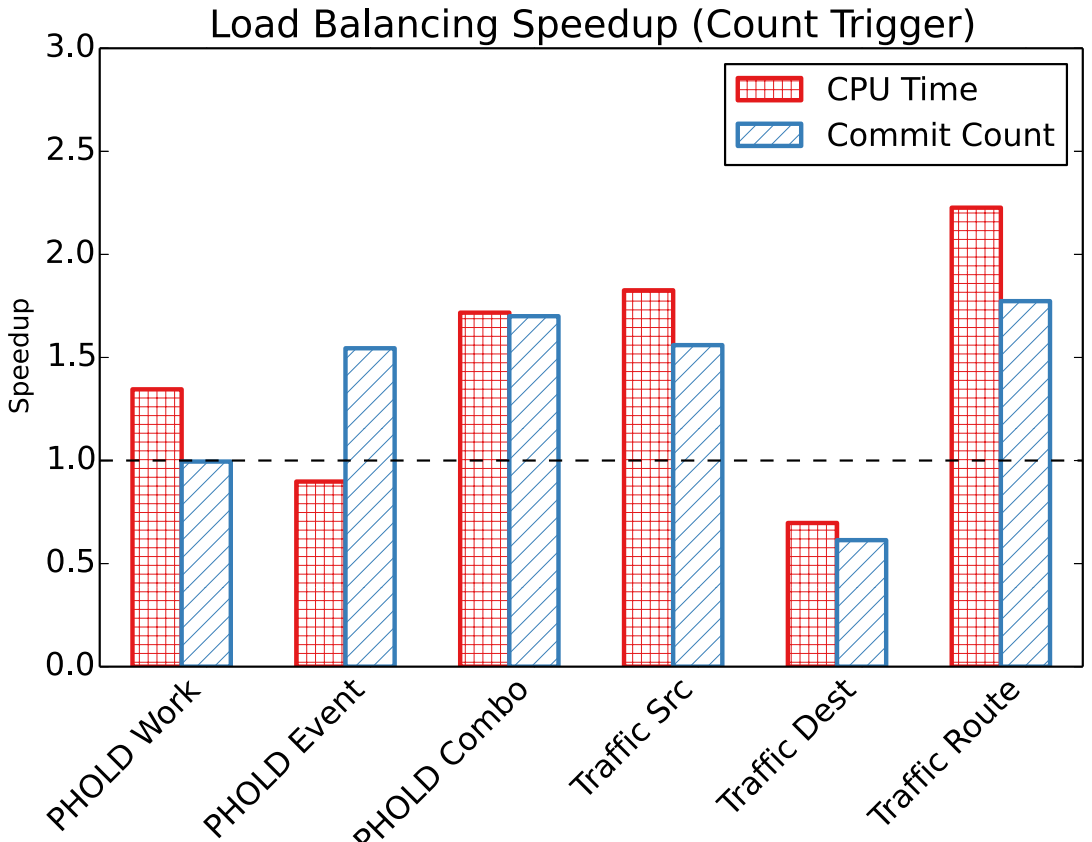


Performance on Blue Waters

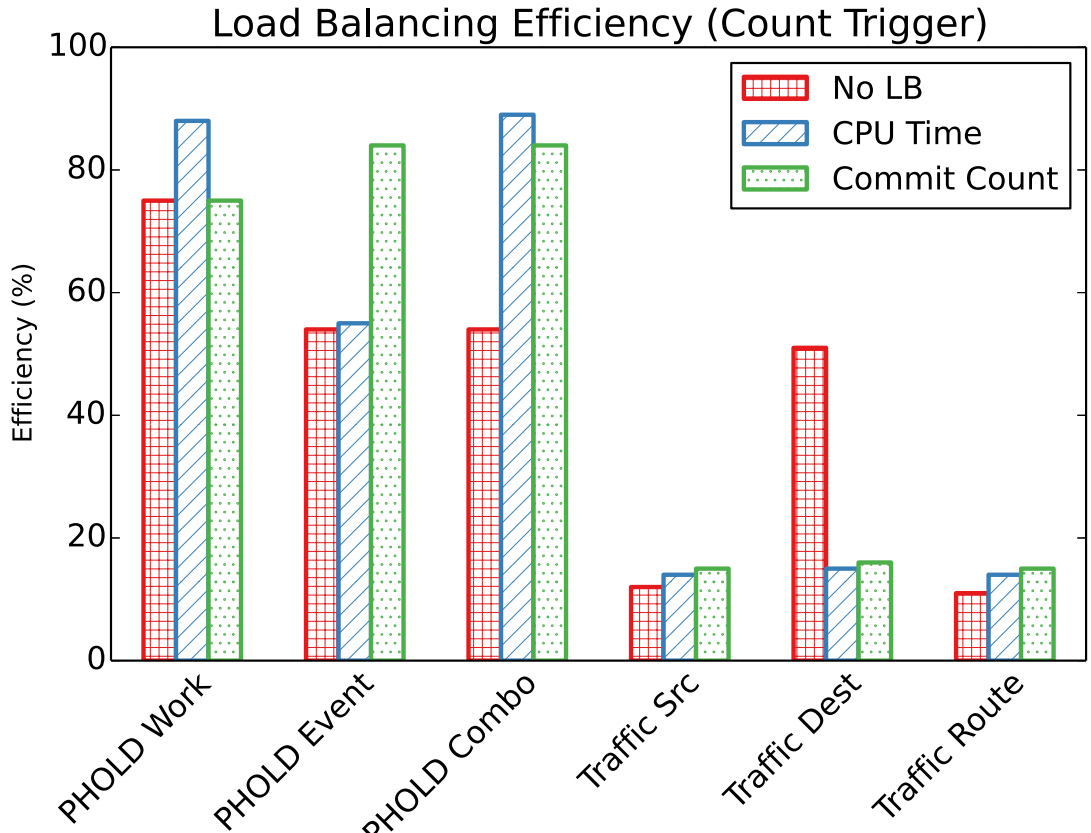


	Sync	Cont
PHOLD Base	98%	95%
PHOLD Work	76%	52%
PHOLD Event	84%	60%
PHOLD Combo	93%	31%
DFly Uniform	62%	36%
DFly Worst	91%	2%
DFly Trans	85%	27%
DFly NN	93%	67%
Traffic Base	96%	55%
Traffic Src	97%	16%
Traffic Dest	96%	52%
Traffic Route	97%	15%

Load Balancing on Blue Waters

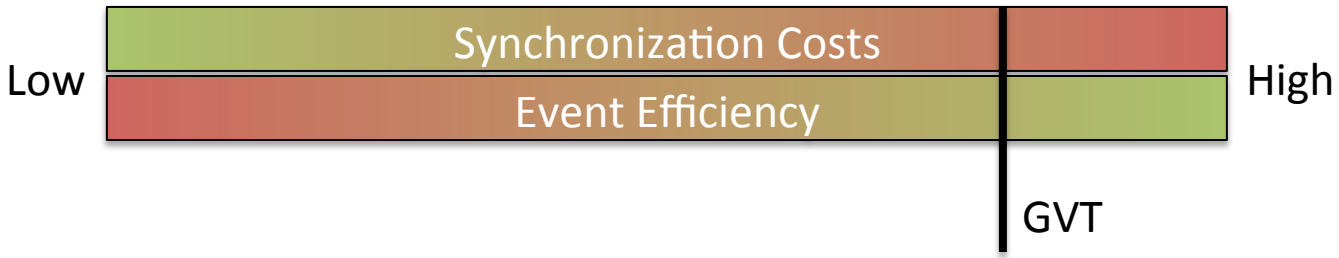


Load Balancing on Blue Waters



Decoupling the Tuning Knob

- GVT can tune for synchronization
- LB can tune for efficiency

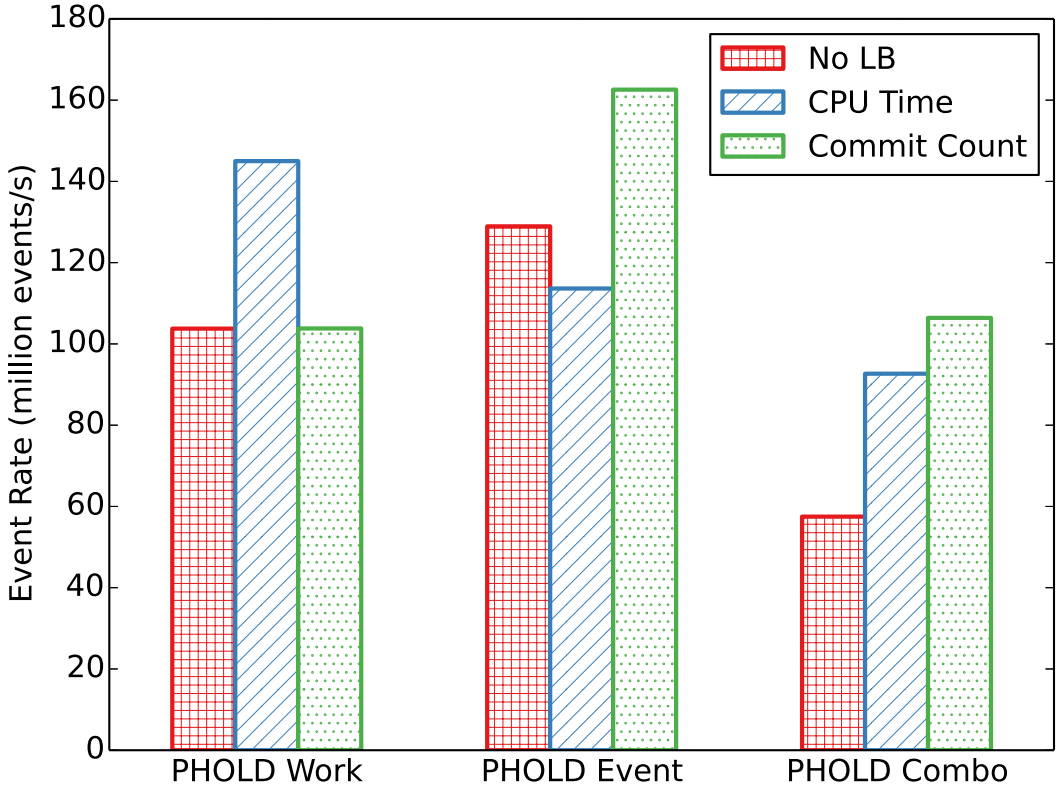


Decoupling the Tuning Knob

- GVT can tune for synchronization
- LB can tune for efficiency

PPL UIUC

Continuous GVT w/ Load Balancing

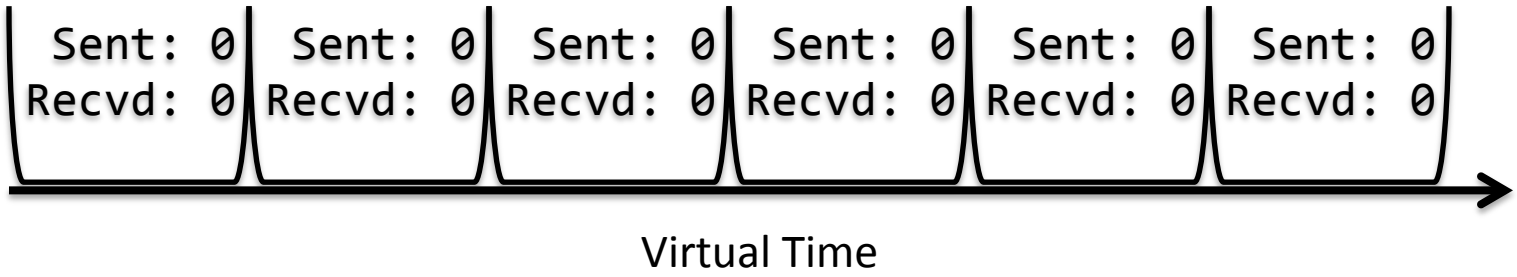


Bucketed GVT Scheme

PPL UIUC

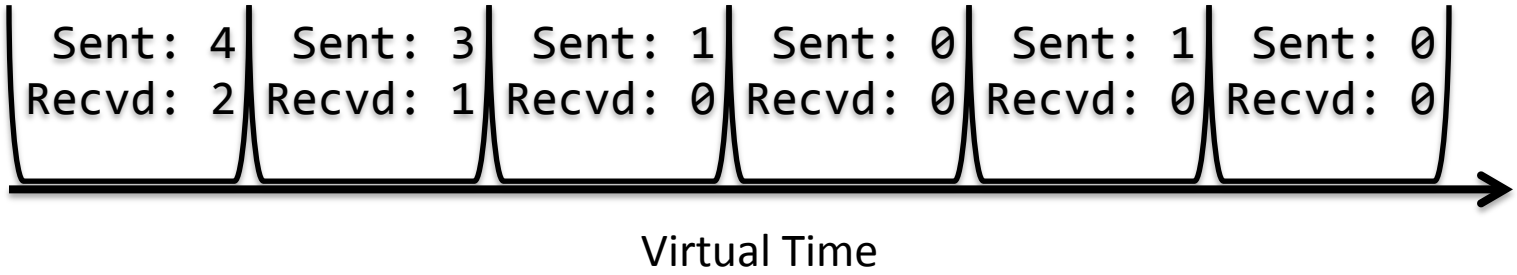
Bucketed GVT Scheme

1. Divide entire simulation timeline into buckets



Bucketed GVT Scheme

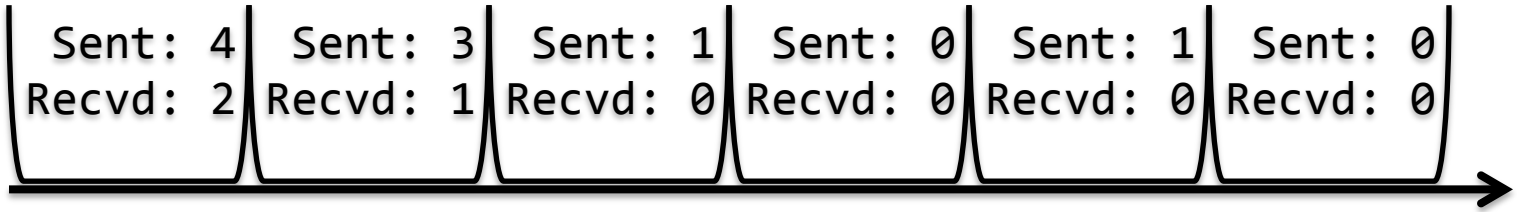
1. Divide entire simulation timeline into buckets
2. Monitor incoming/outgoing events



PPL UIUC

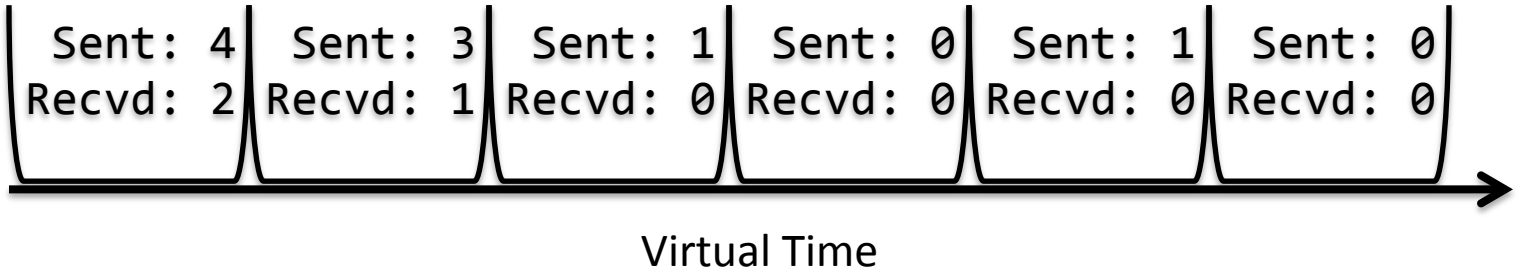
Bucketed GVT Scheme

1. Divide entire simulation timeline into buckets
2. Monitor incoming/outgoing events
3. As buckets get passed, advance GVT



Bucketed GVT Scheme

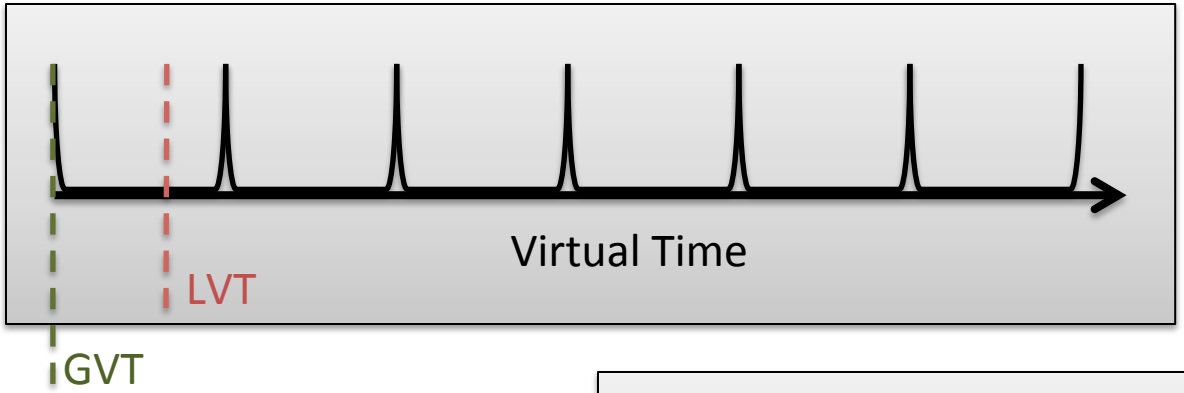
1. Divide entire simulation timeline into buckets
2. Monitor incoming/outgoing events
3. As buckets get passed, advance GVT



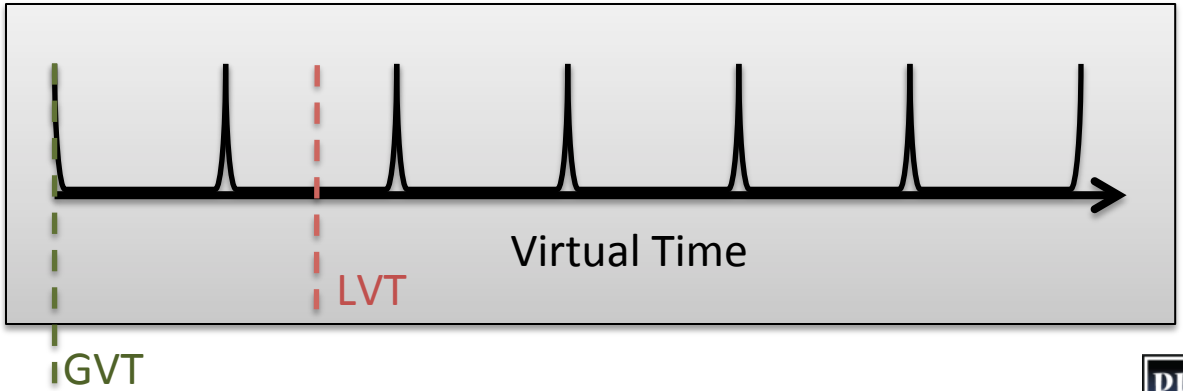
How do we know when a bucket is passed?

PPL UIUC

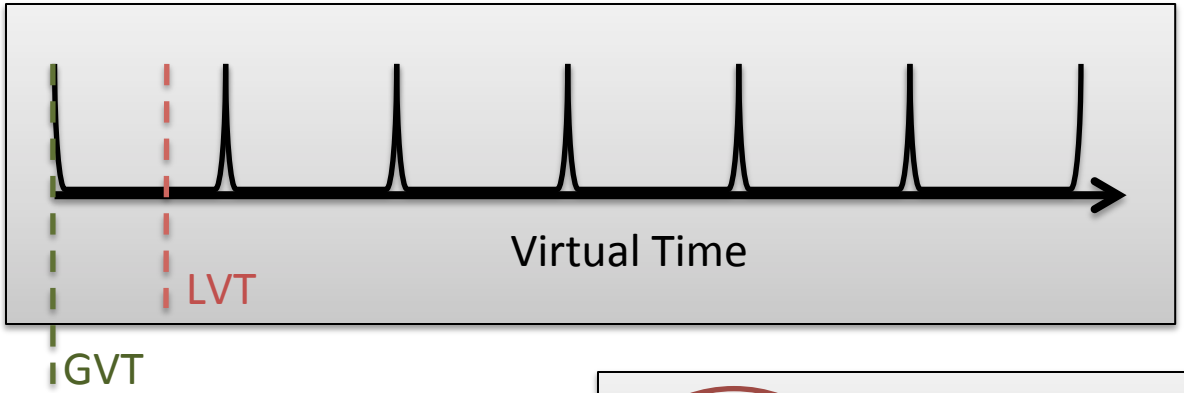
Bucketed GVT Scheme



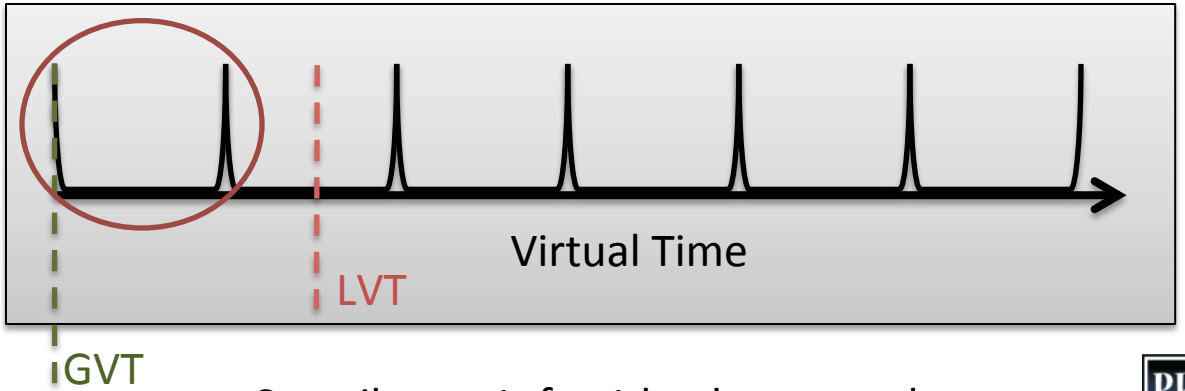
Each PE contributes to a min reduction when it passes a bucket boundary



Bucketed GVT Scheme

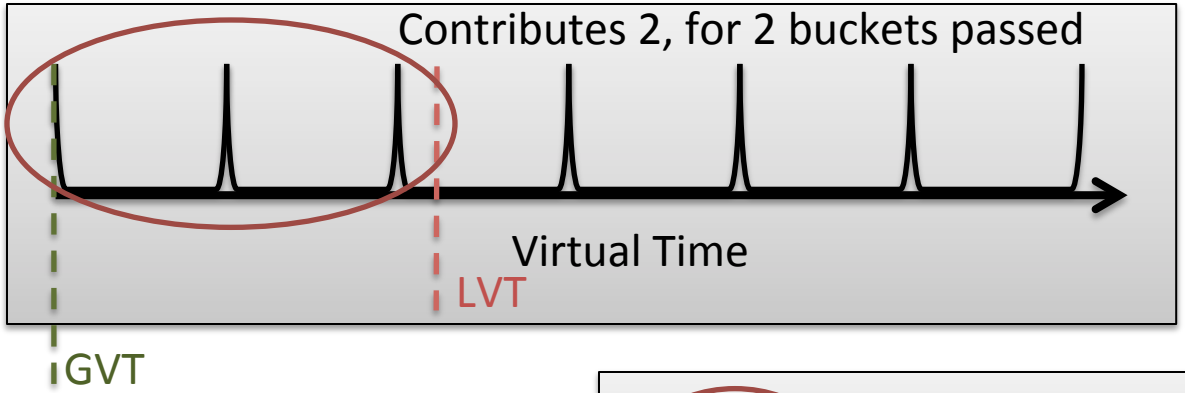


Each PE contributes to a min reduction when it passes a bucket boundary

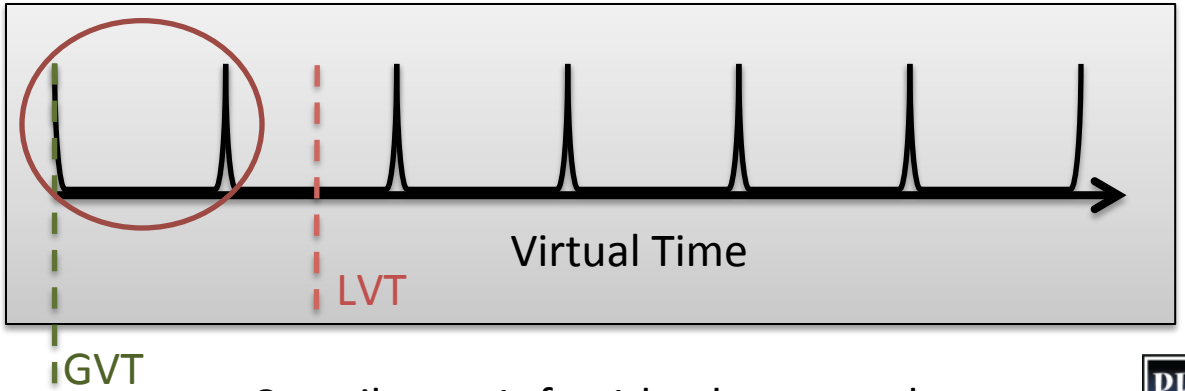


Contributes 1, for 1 bucket passed

Bucketed GVT Scheme



Each PE contributes to a min reduction when it passes a bucket boundary



Contributes 1, for 1 bucket passed

Bucketed GVT Scheme

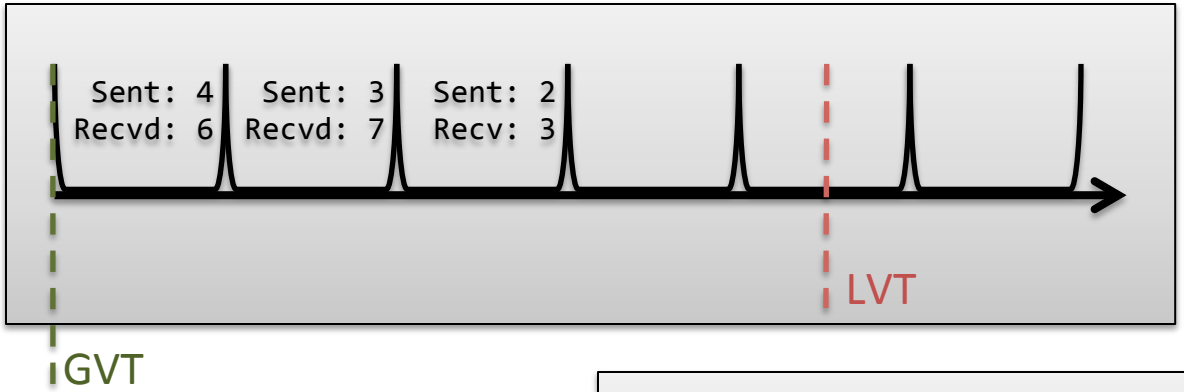
- Once the reduction completes, we know everyone has passed at least some buckets
- Start a series of “tuple” reductions of bucket counts, and buckets passed
- Similar to completion detection with extra information

Bucketed GVT Scheme

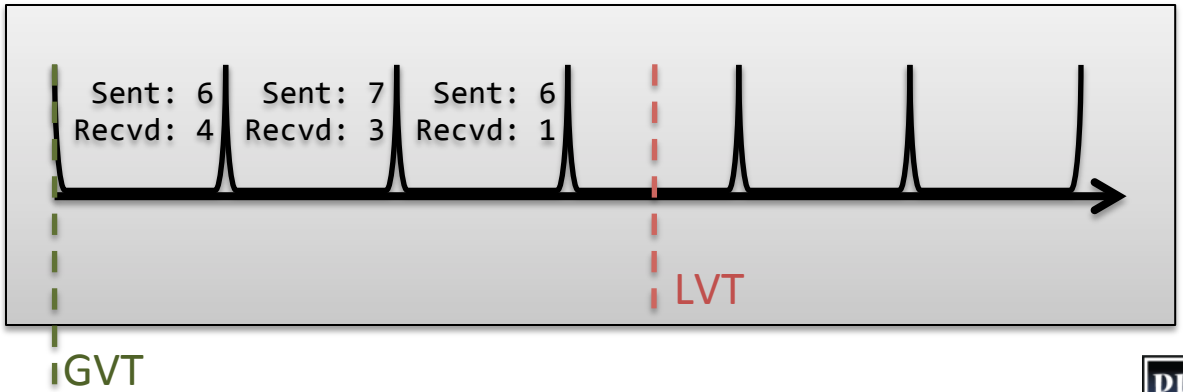
- Result of count reductions for n buckets
  - Sent counts: $[s_1, s_2, \ldots, s_n]$
  - Received counts: $[r_1, r_2, \ldots, r_n]$
  - New min bucket passed: k
- Find x such that $s_i == r_i$ for all $i \leq x$ and $x \leq k$
- Advance GVT x buckets, k-x keep reducing

PPL UIUC

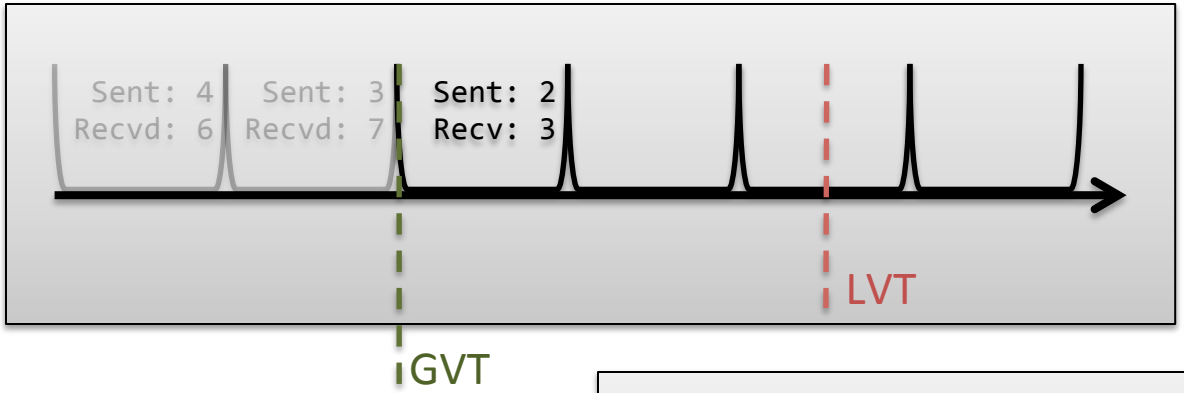
Bucketed GVT Scheme



First 2 buckets counts match, and all PEs have passed at least 3 buckets

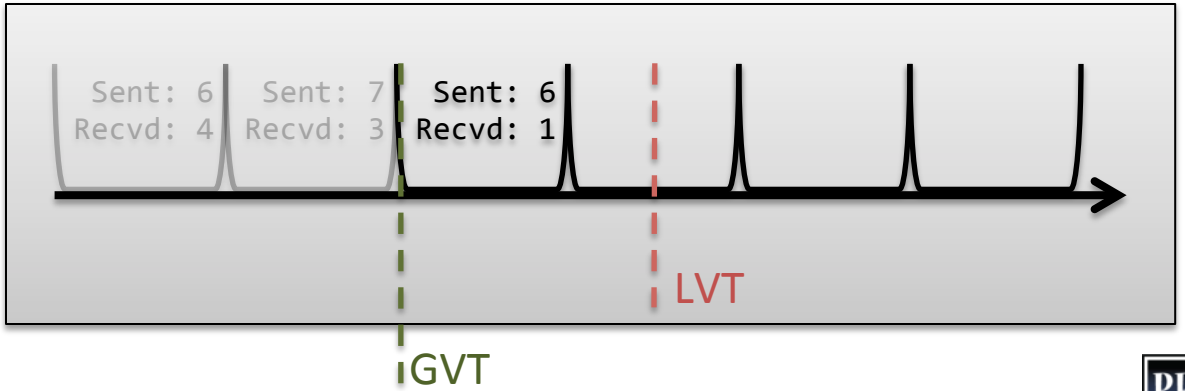


Bucketed GVT Scheme



First 2 buckets counts match, and all PEs have passed at least 3 buckets

Advance GVT 2 buckets, continue waiting on the 3rd, and possibly pull in more

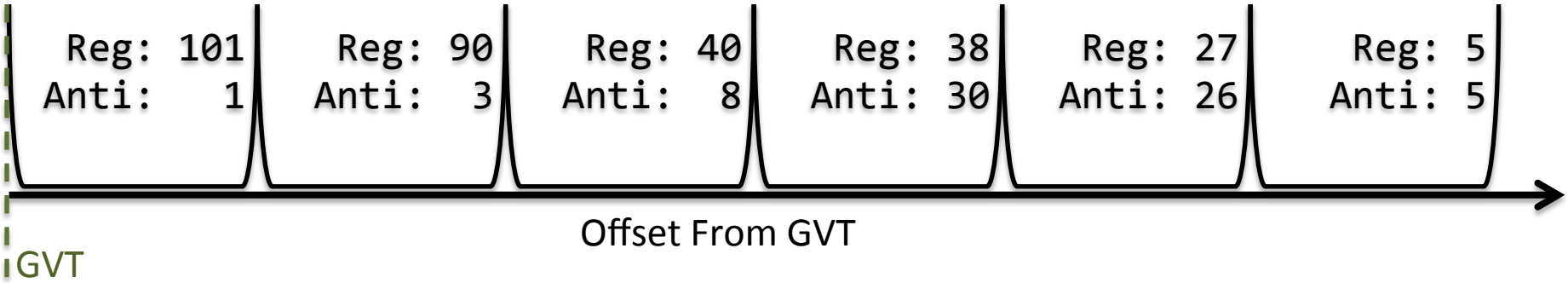


PPL UIUC

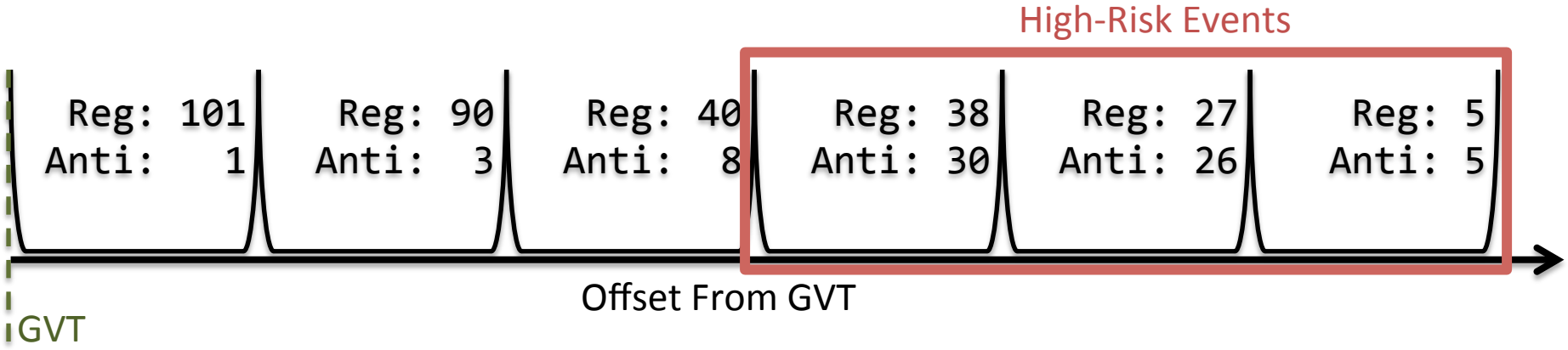
Early Results

- Comparable, or better, to Continuous
- More manageable/robust
- Easier to tune/understand
- Opportunity for adaptive event control
  - Adaptively hold back high-risk events
  - Reduce overall communication load

Adaptive Event Delay



Adaptive Event Delay



Questions?

PPL UIUC