Balancing Speculative Loads in Parallel Discrete Event Simulation

Eric Mikida

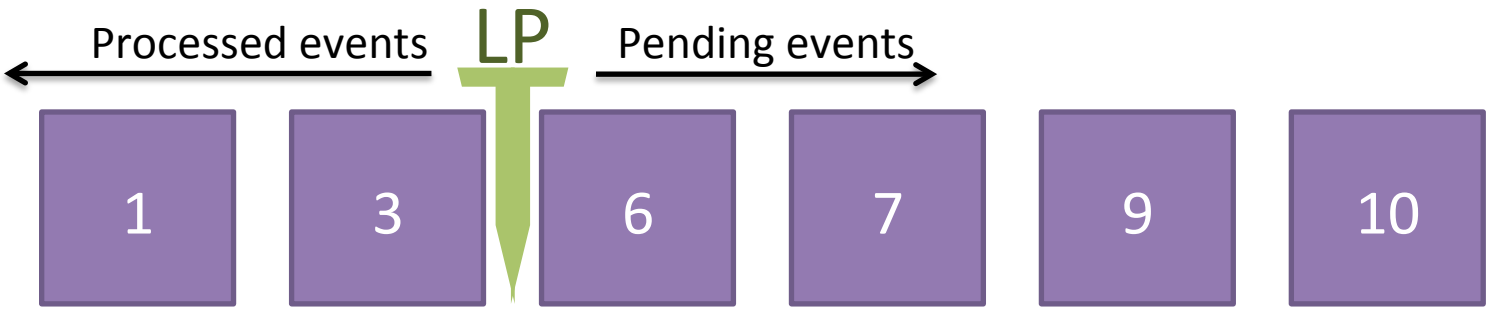
PPL UIUC

Brief PDES Description

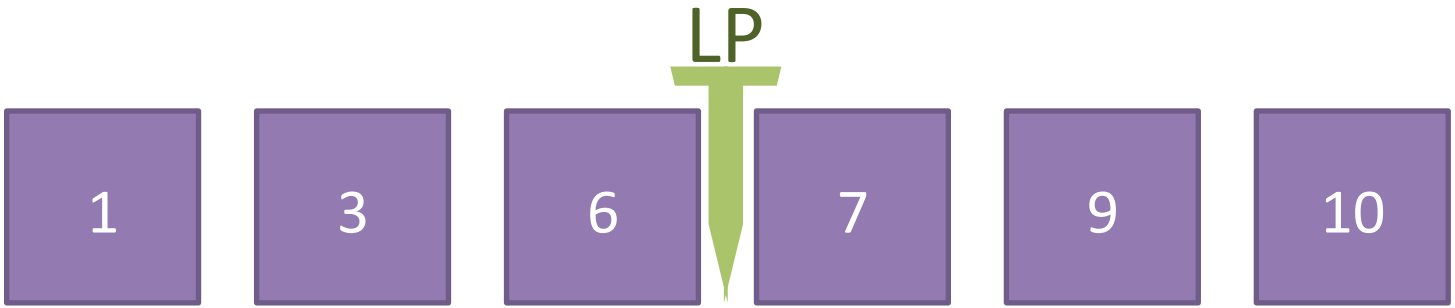
- Simulation made up of Logical Processes (LPs)
- LPs process events in timestamp order
- Synchronization is conservative or optimistic
- Periodically compute global virtual time (GVT)

PPL UIUC

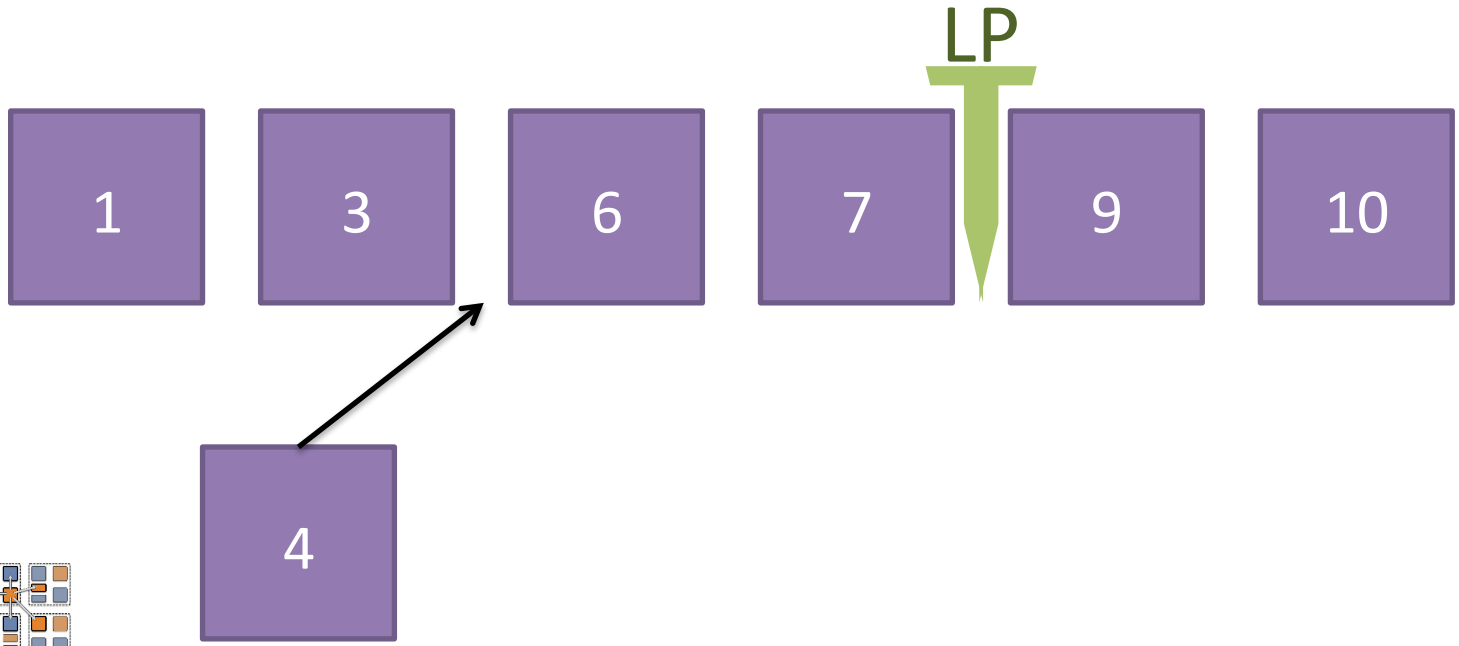
Optimistic Execution



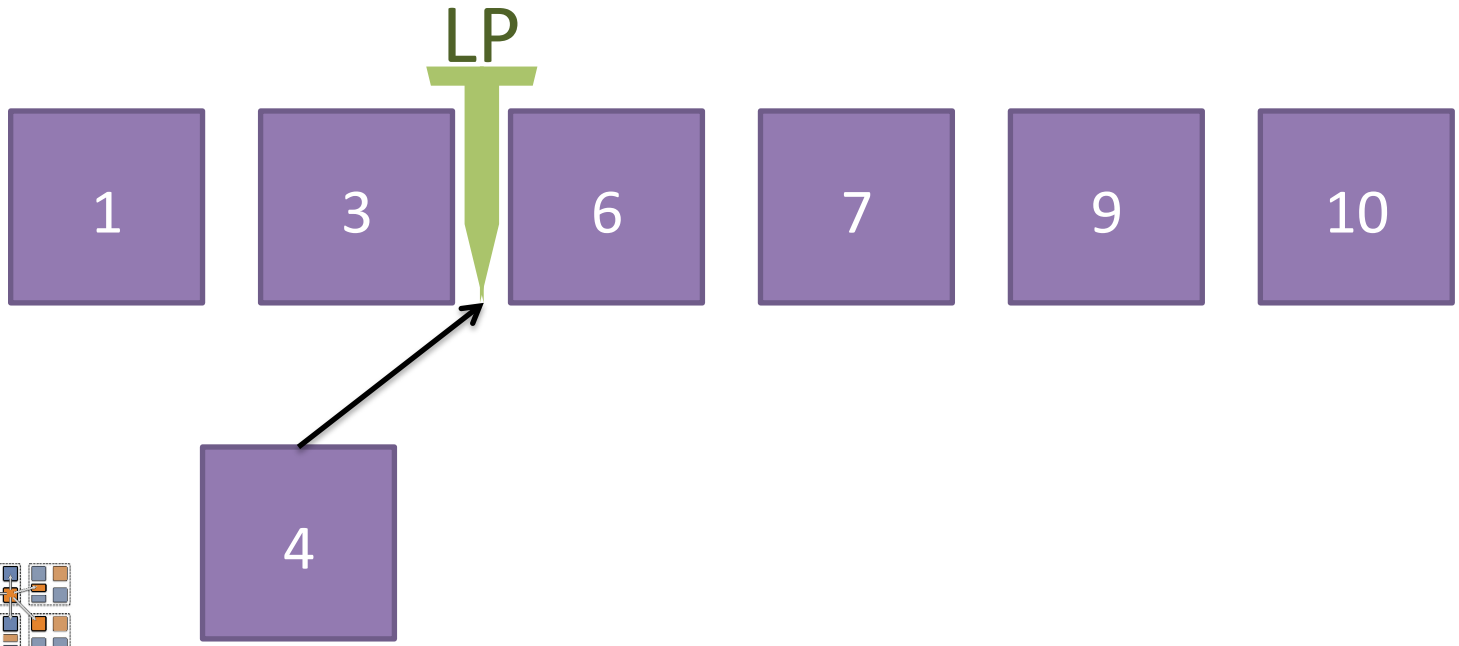
Optimistic Execution



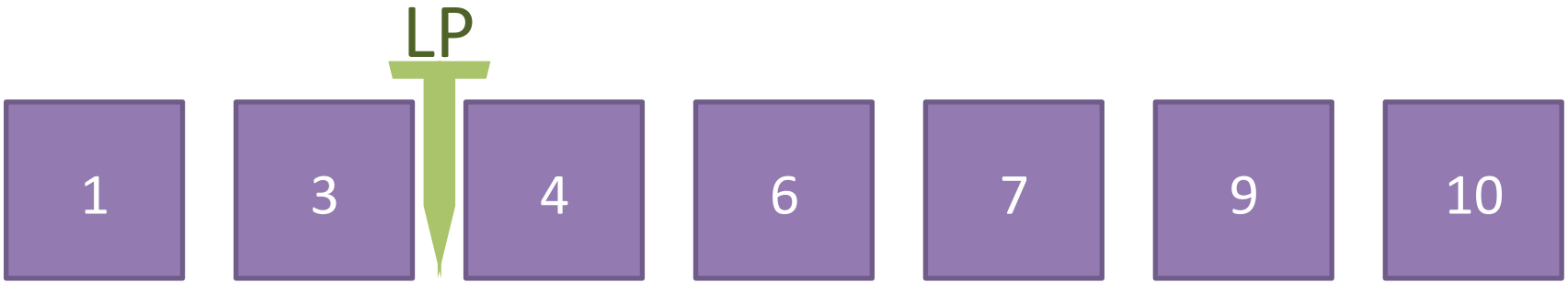
Optimistic Execution



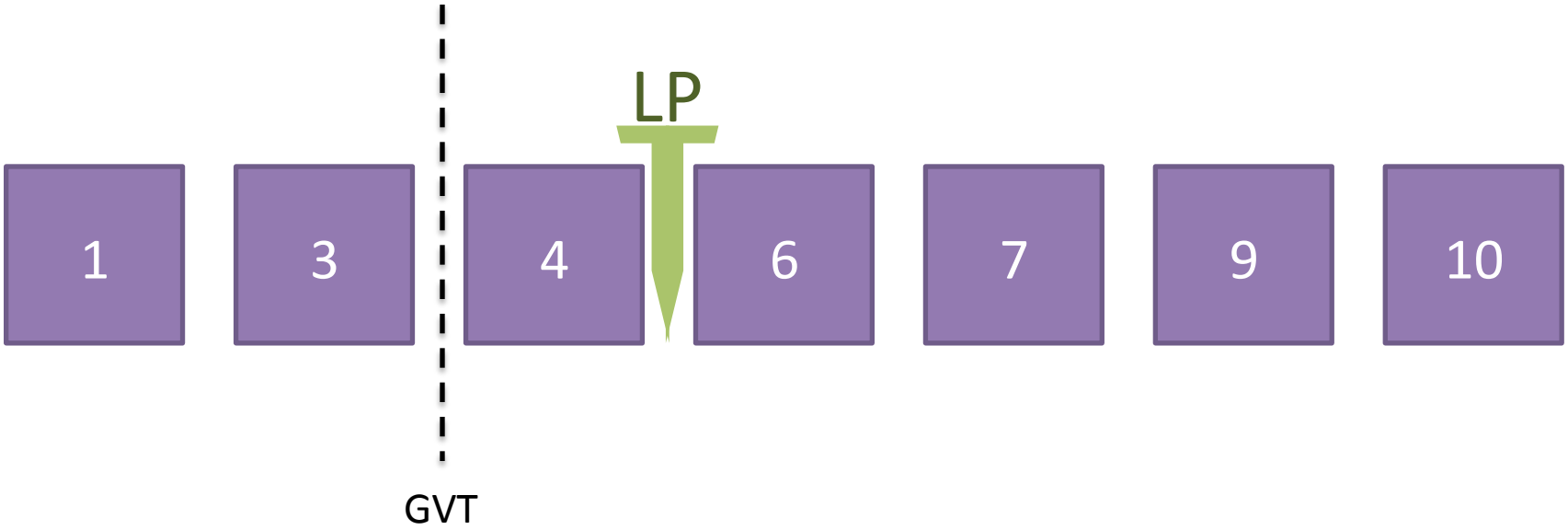
Optimistic Execution



Optimistic Execution



Optimistic Execution



PPL UIUC

Performance Metrics

Event Rate = $E_{committed}$ / s

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

Load Balance = ??

PPL UIUC

What is “load”?

- Charm++ automatically measures CPU time
  - Makes sense when all work is useful work
  - Relies on principle of persistence
  - Balances CPU time per PE

PPL UIUC

What is “load”?

- Charm++ automatically measures CPU time
  - Makes sense when all work is useful work
  - Relies on principle of persistence
  - Balances CPU time per PE

Does this make sense in a speculative setting?

PPL UIUC

Example

LP 1	LP 2	LP 3
Executed: Committed: Rolled Back:	Executed: Committed: Rolled Back:	Executed: Committed: Rolled Back:

Example

LP 1	LP 2	LP 3
Executed: 5 Committed: Rolled Back:	Executed: 5 Committed: Rolled Back:	Executed: 5 Committed: Rolled Back:

PPL UIUC

Example

LP 1	LP 2	LP 3
Executed: 5 Committed: 4 Rolled Back:	Executed: 5 Committed: 0 Rolled Back:	Executed: 5 Committed: 0 Rolled Back:

PPL UIUC

Example

LP 1	LP 2	LP 3
Executed: 5 Committed: 4 Rolled Back: 0	Executed: 5 Committed: 0 Rolled Back: 4	Executed: 5 Committed: 0 Rolled Back: 0

PPL UIUC

Example

LP 1	LP 2	LP 3
Executed: 5 Committed: 4 Rolled Back: 0	Executed: 5 Committed: 0 Rolled Back: 4	Executed: 5 Committed: 0 Rolled Back: 0

Roughly the same CPU time spent executing events

PPL UIUC

Example

LP 1	LP 2	LP 3
Executed: 5 Committed: 4 Rolled Back: 0	Executed: 5 Committed: 0 Rolled Back: 4	Executed: 5 Committed: 0 Rolled Back: 0

Roughly the same CPU time spent executing events

How does the load balancer differentiate?

How does GVT affect balance?

Count-Based GVT

- GVT computed every X events
- Doesn't attempt to bound optimism
- Can lead to poor event efficiency

Leash-Based GVT

- GVT computed every X units of virtual time
- Keeps virtual times balanced across PEs
- Can lead to poor CPU balance across PEs

PPL UIUC

Benchmarks

PHOLD

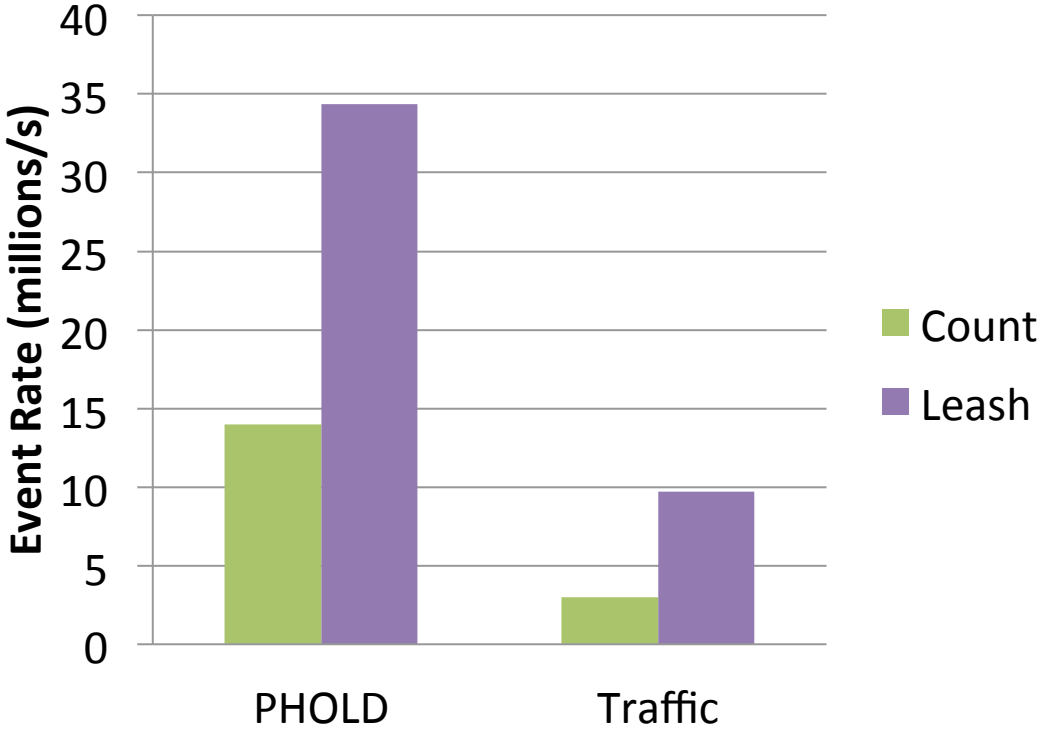
- Common PDES benchmark
- Executing an event causes a new event for a random LP
- Changing event distribution causes imbalance

Traffic

- Simulates a grid of intersections
- Events are cars arriving, leaving, and changing lanes
- Cars travel from source to destination

PPL UIUC

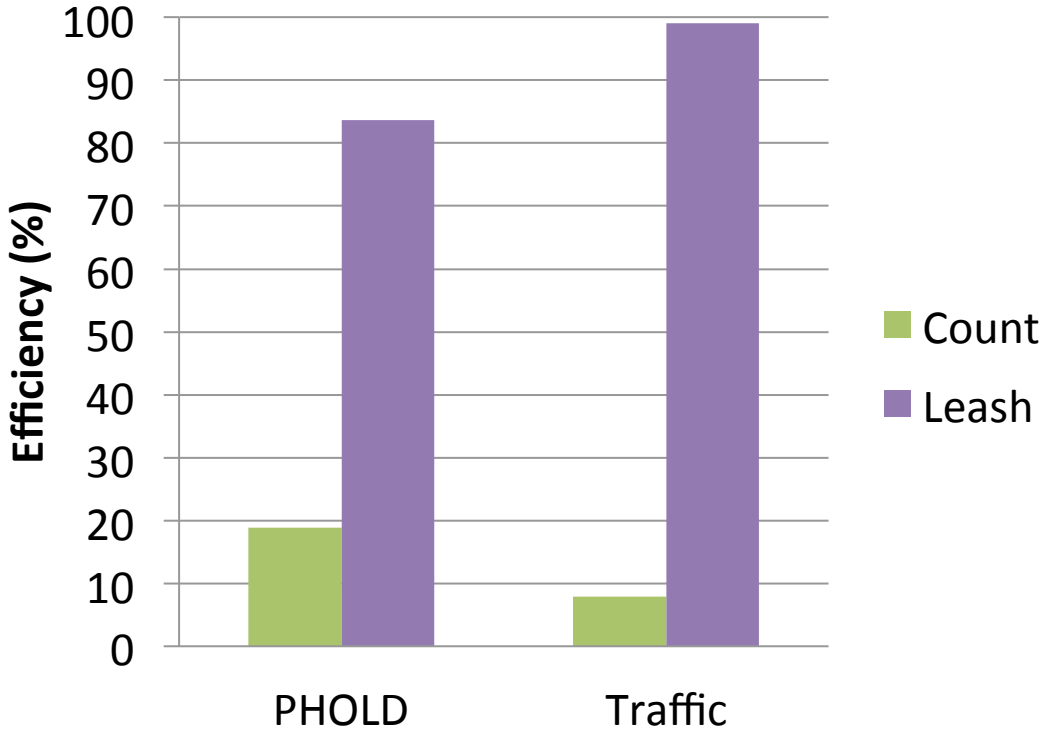
GVT Trigger Event Rates



64 nodes of Vesta (BG/Q)
PHOLD:
- 32 LPs per rank
- 50% remote events

Traffic:
- 64K intersections
- 1M cars

GVT Trigger Comparison



64 nodes of Vesta (BG/Q)
PHOLD:
- 32 LPs per rank
- 50% remote events

Traffic:
- 64K intersections
- 1M cars

Our Load Balancing Goal

- Make sure all PEs have useful work
  - Balance the CPU load
  - Only count useful work
- Maintain a high event efficiency
  - Balance rate of progress
  - Leads to less overall work

Redefine “Load” for PDES

Past-Looking Metrics

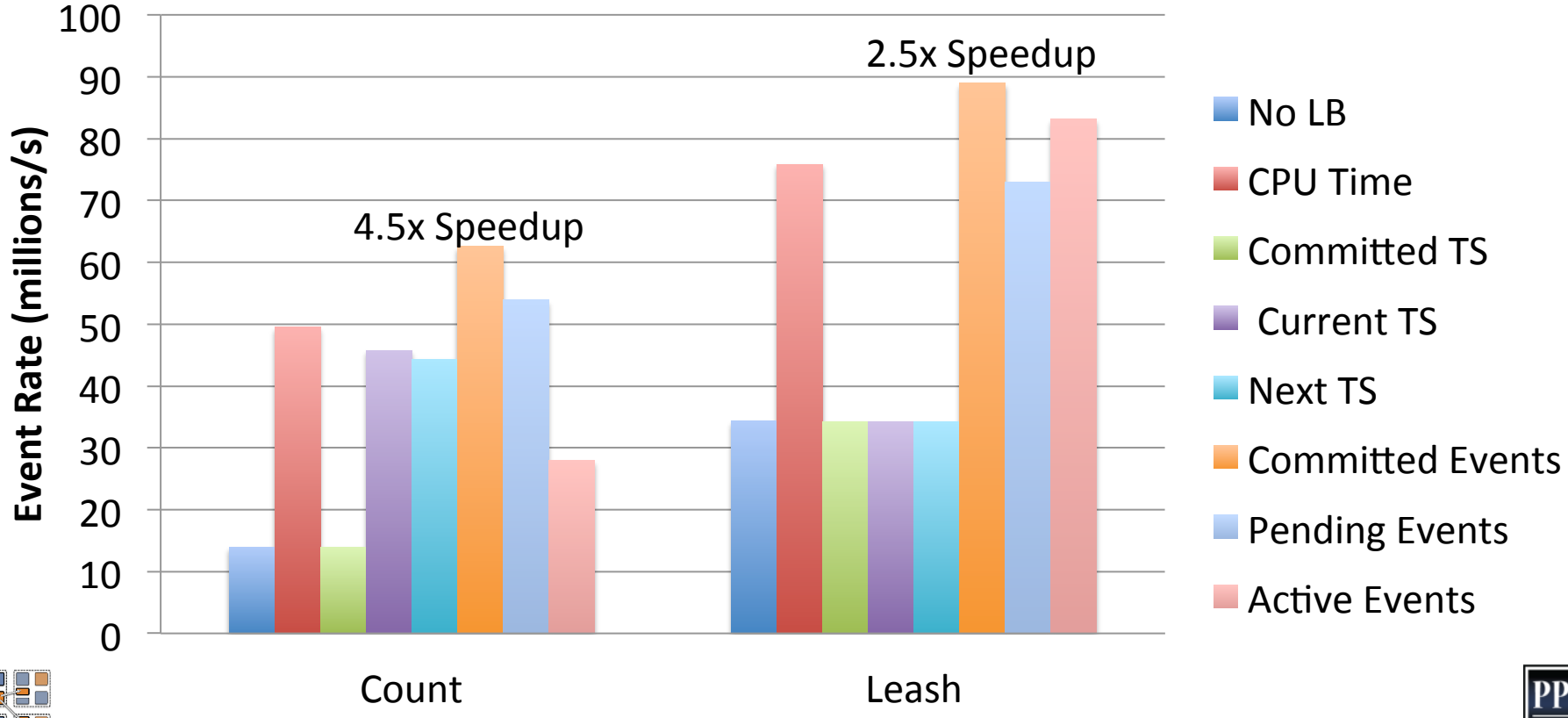
- CPU Time
- Current Timestamp
- Committed Timestamp
- Committed Events
- Potential Committed Events

Future-Looking Metrics

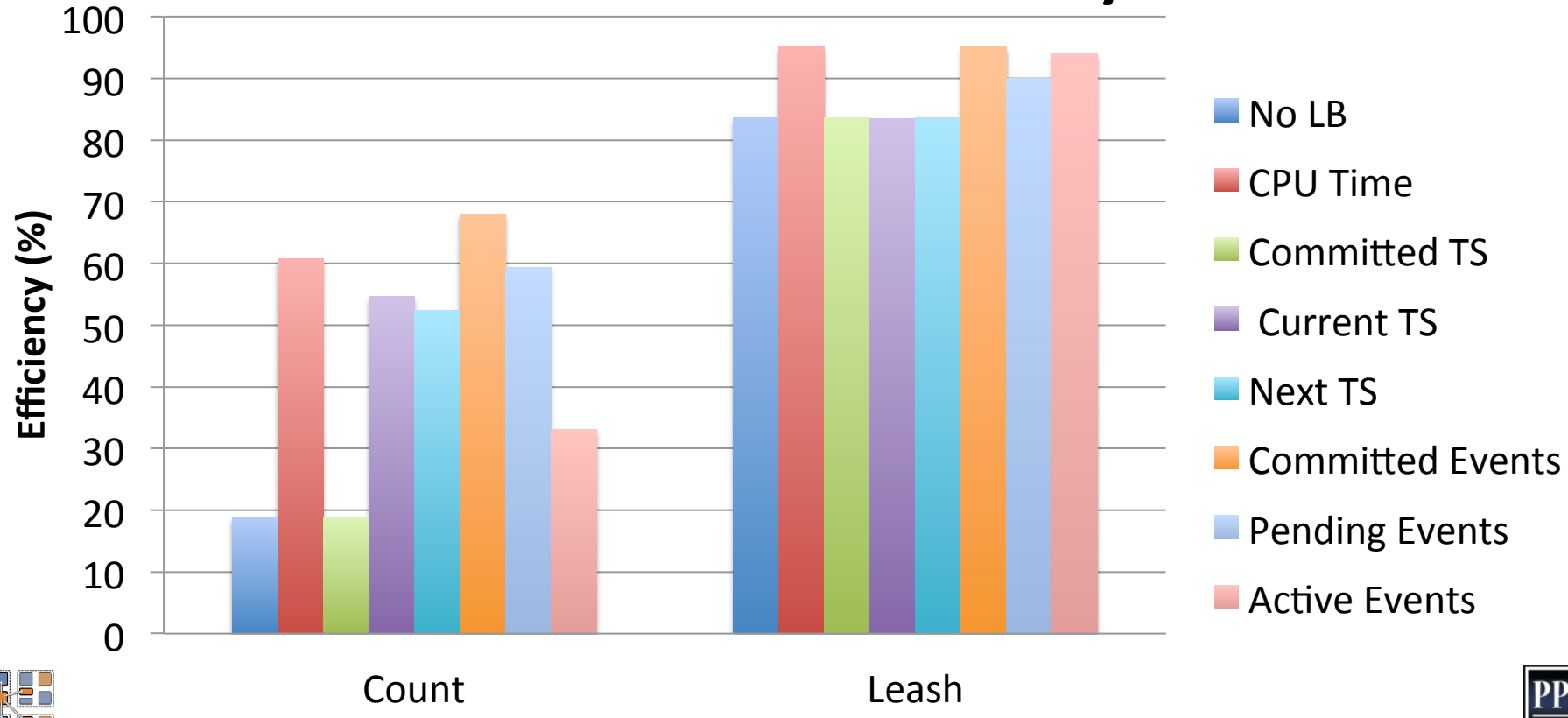
- Next Timestamp
- Pending Events
- Weighted Pending Events
- “Active” Events

PPL UIUC

PHOLD Event Rate

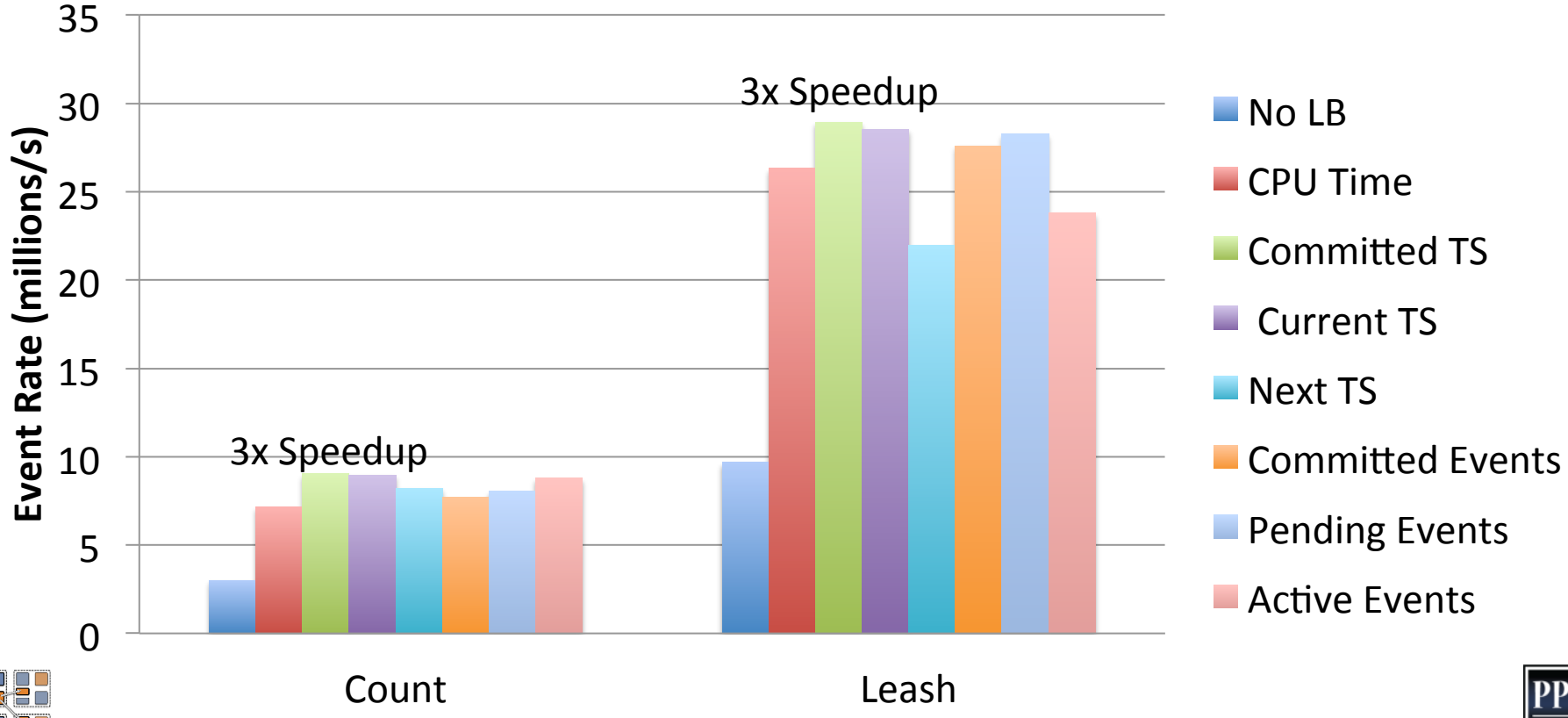


PHOLD Efficiency

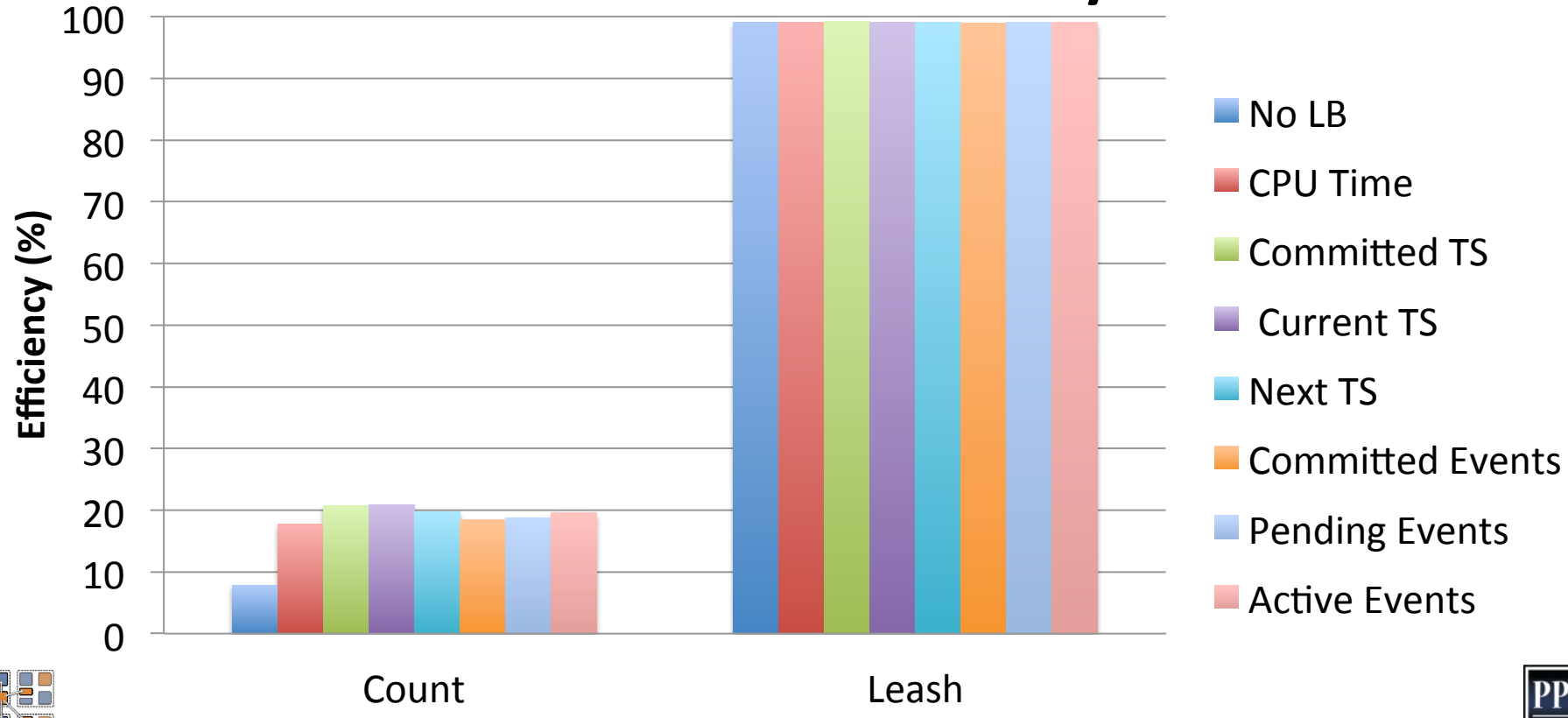


PPL UIUC

Traffic Event Rate



Traffic Efficiency



What's next?

- Better visualization/analysis tools
- More diverse set of models
- Conservative synchronization
- Vector load balancing strategies
- Adaptive load balancer
- Combine with GVT work