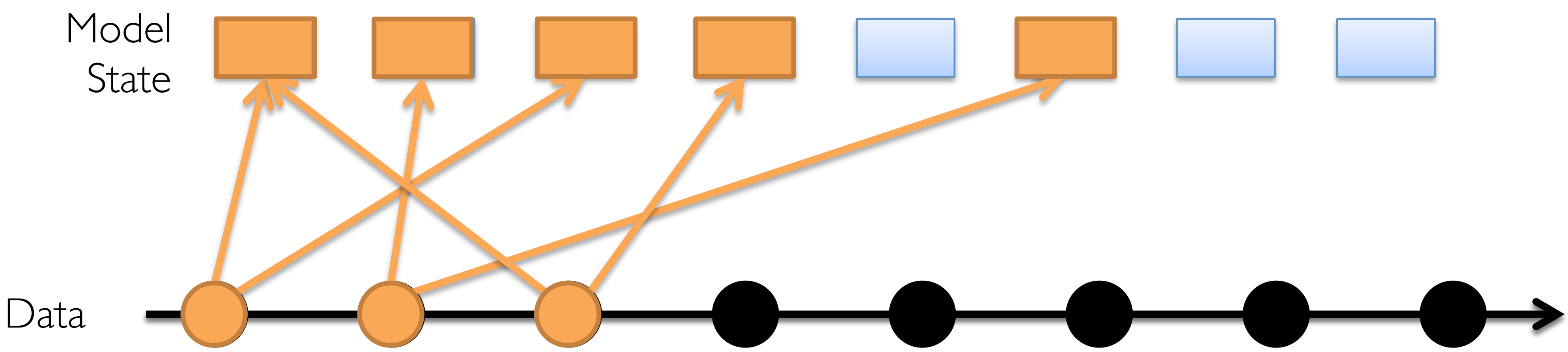
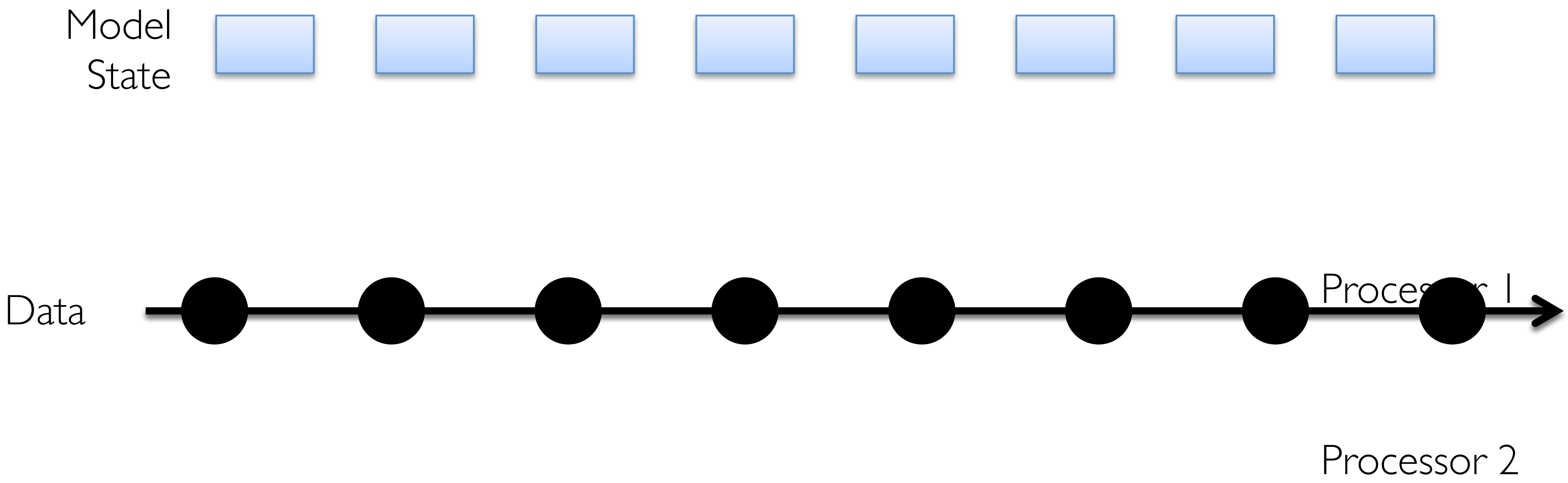
Optimistic Concurrency Control in the Design and Analysis of Parallel Learning Algorithms

Joseph E. Gonzalez
Postdoc, UC Berkeley AMPLab
Co-founder, GraphLab Inc.
jegonzal@eecs.berkeley.edu

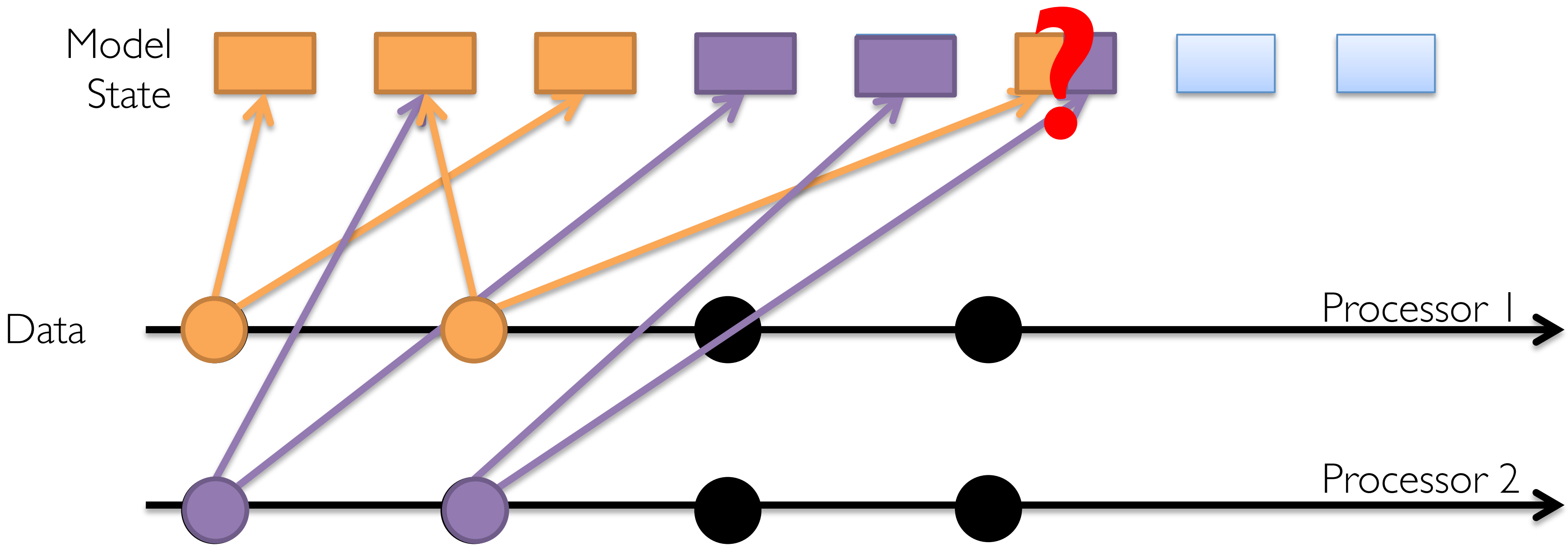
Serial Inference



Parallel Inference



Parallel Inference



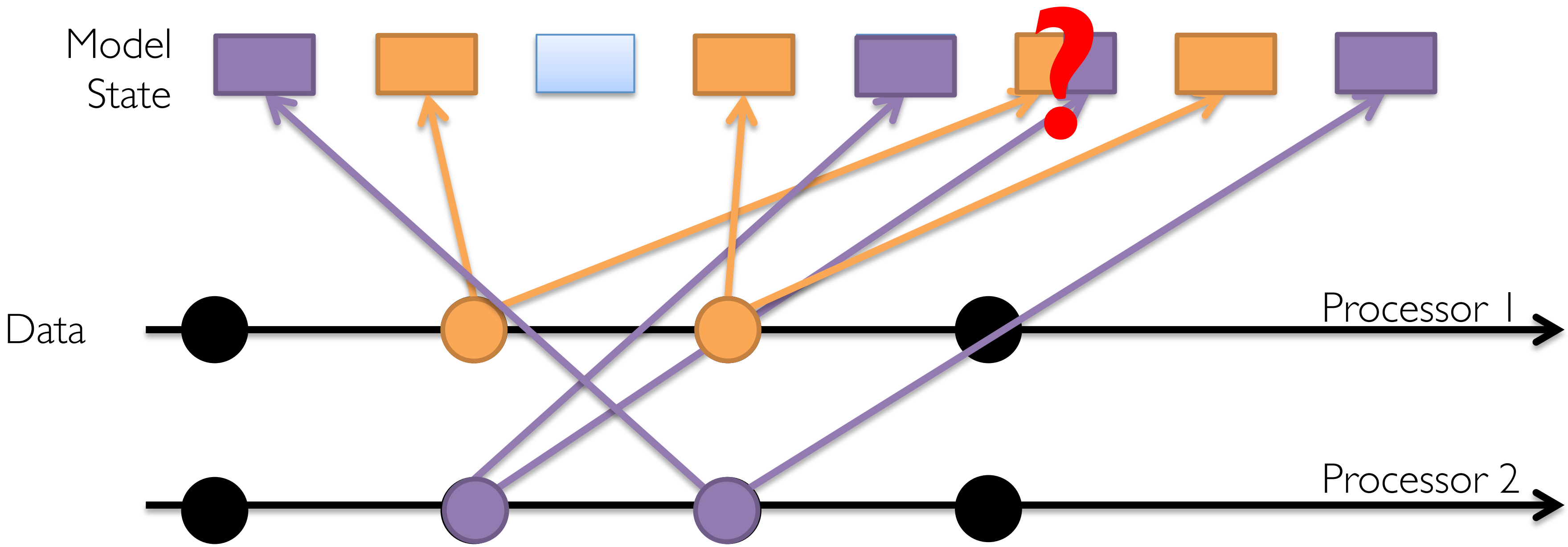
Correctness:

serial equivalence

Concurrency:

more machines = less time

Coordination Free Parallel Inference

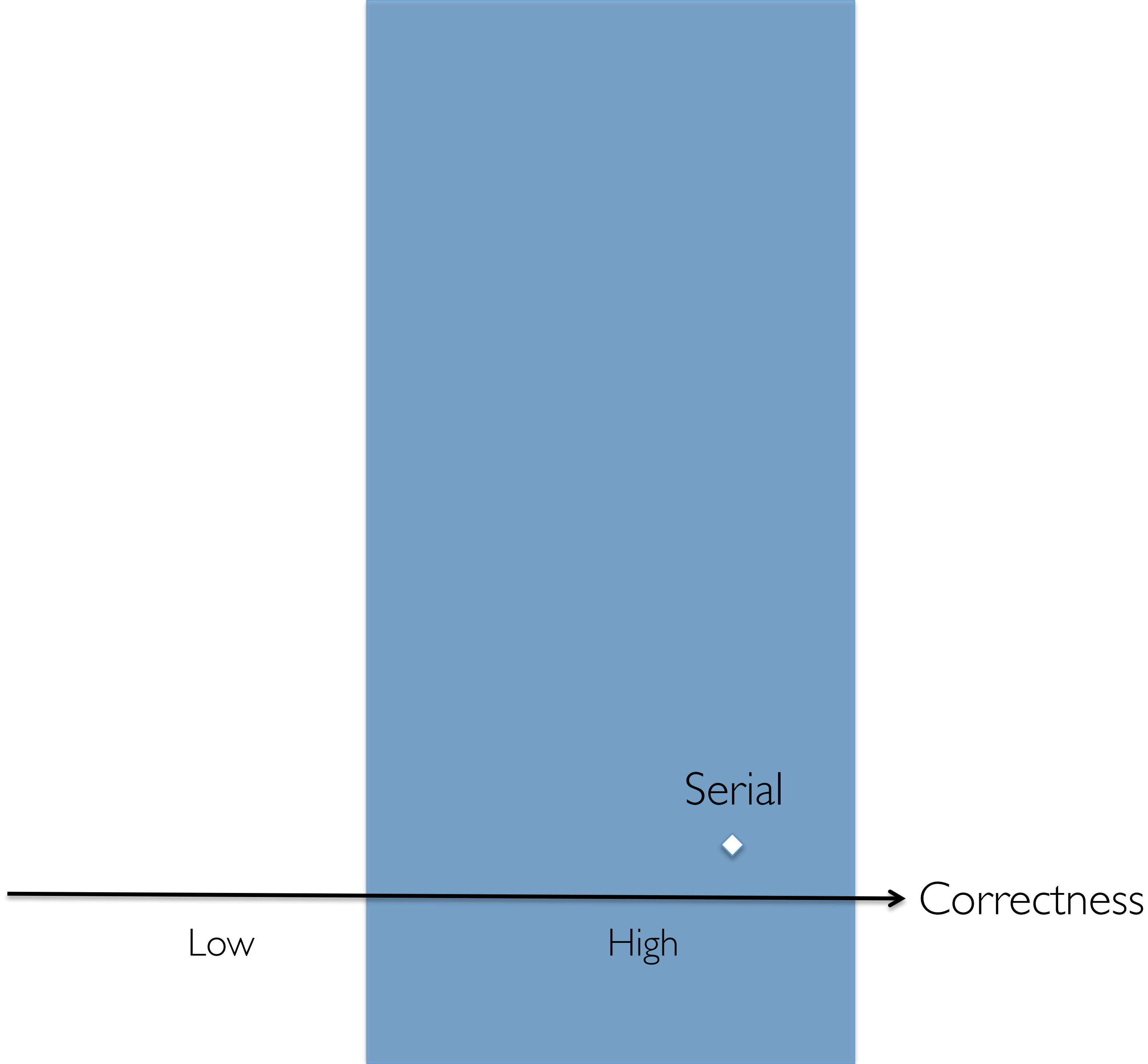


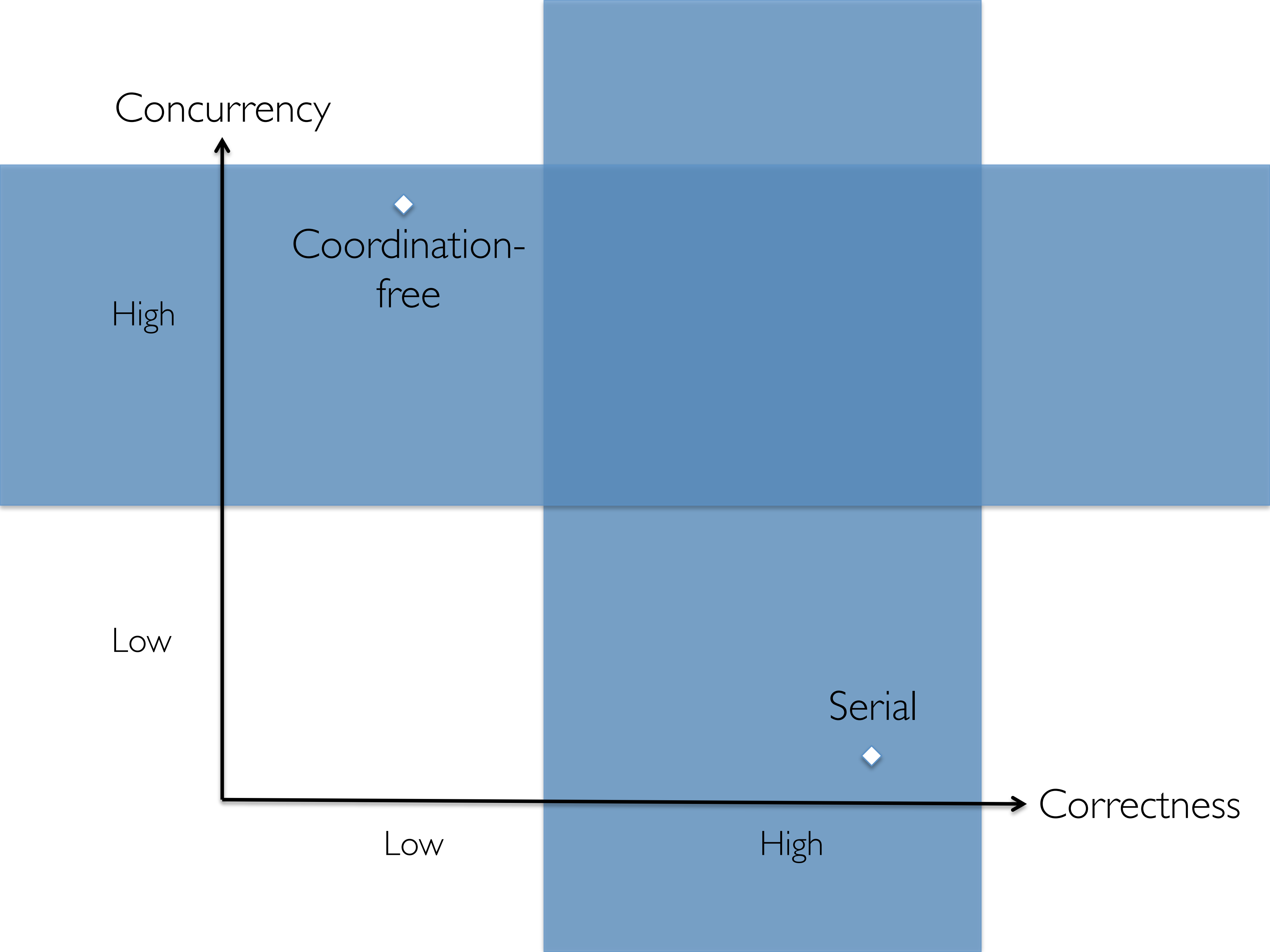
Correctness? Concurrency:

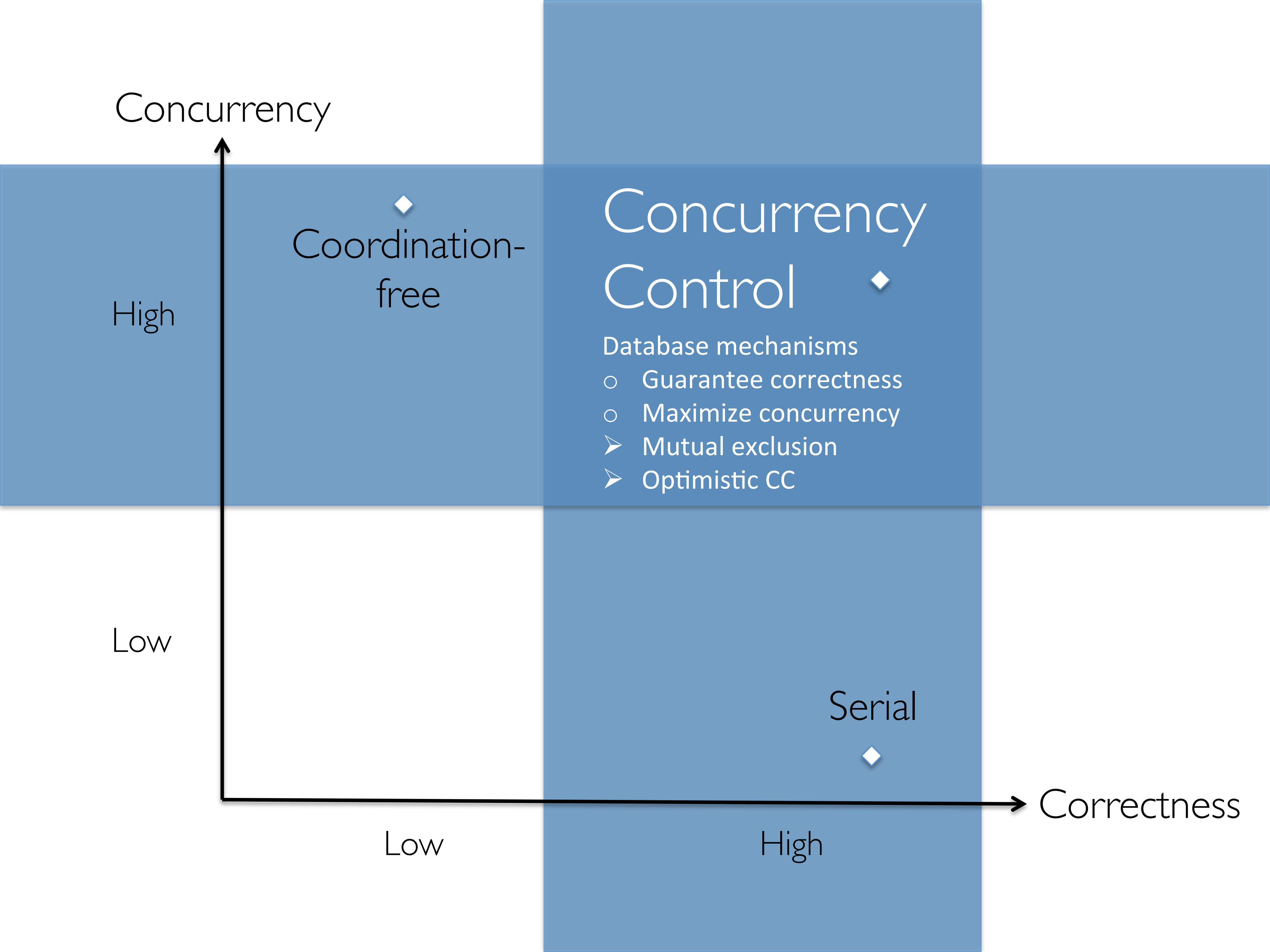
Keep Calm and Carry On.

Depends on Assumptions

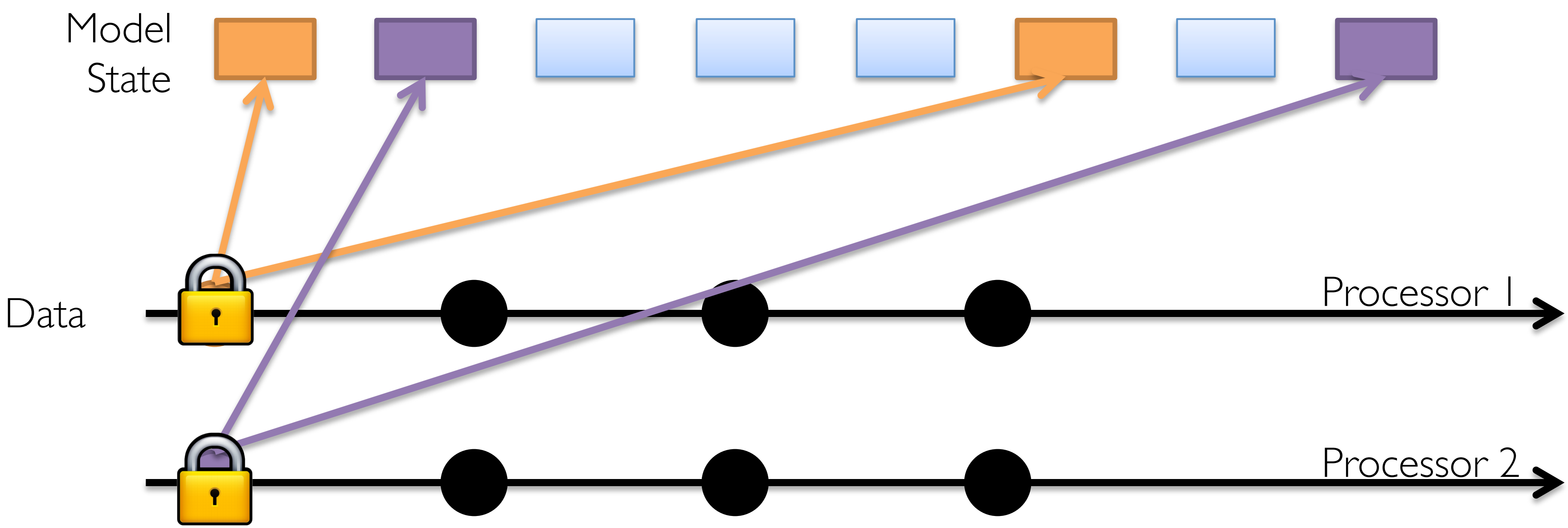
(almost) free


Serial
Correctness
Low
High


Concurrency
High
Low
Coordination-free
Serial
Correctness
Low
High

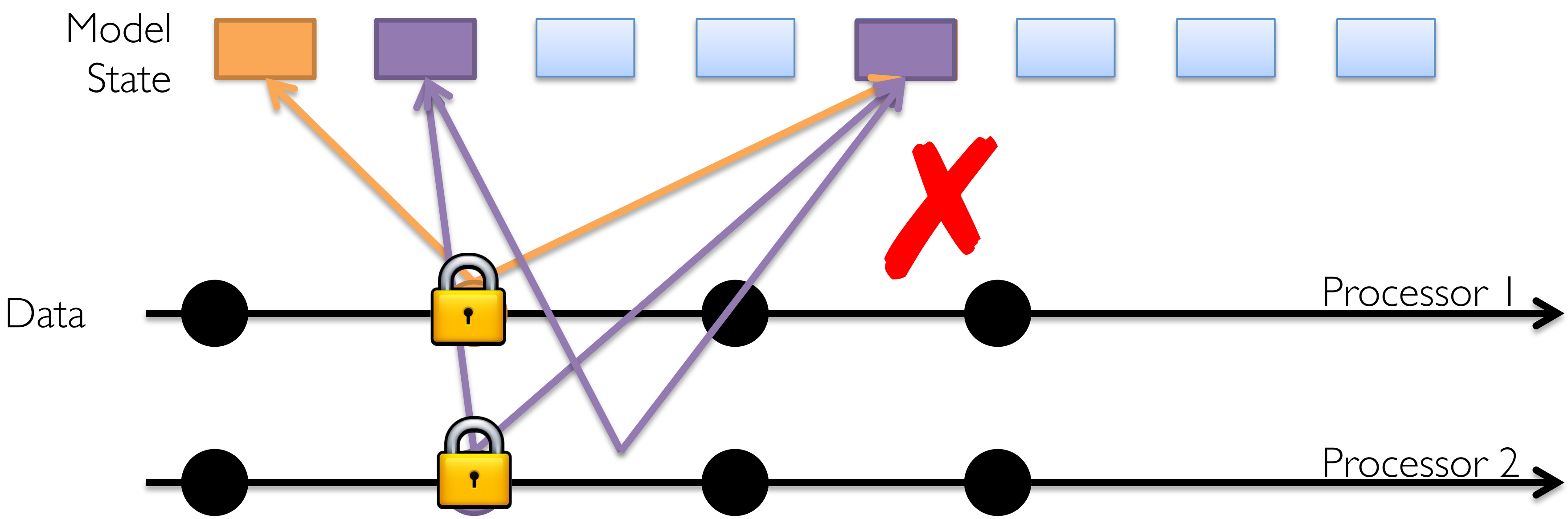

Concurrency
Coordination-free
Concurrency Control
Database mechanisms
Guarantee correctness
Maximize concurrency
Mutual exclusion
Optimistic CC
High
Low
Serial
Correctness
Low
High

Mutual Exclusion Through Locking



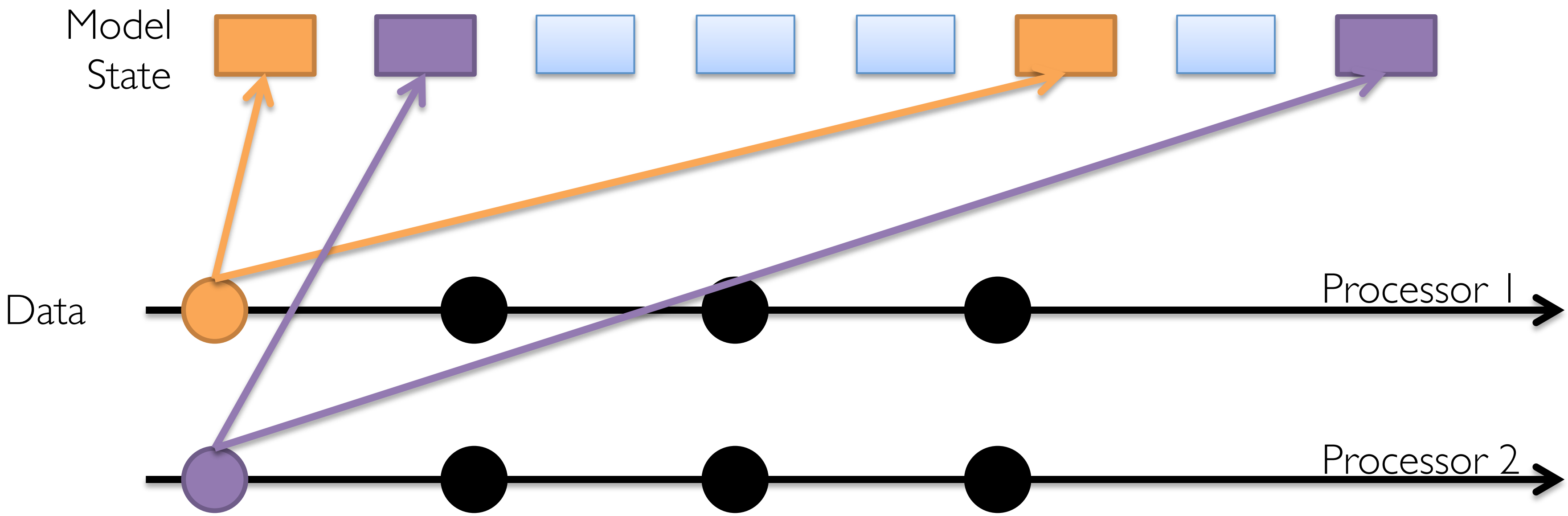
Introducing locking (scheduling) protocols to prevent potential conflicts.

Mutual Exclusion Through Locking



Enforce serialization of computation that could conflict.

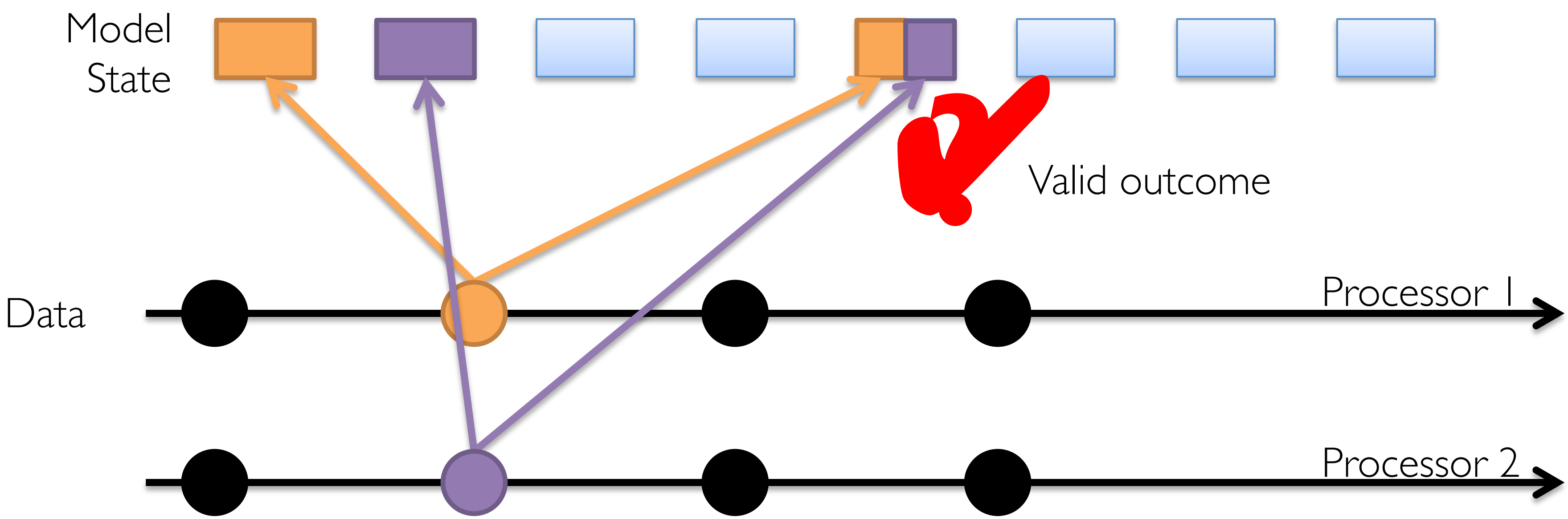
Optimistic Concurrency Control



Allow computation to proceed without blocking.

Kung & Robinson. *On optimistic methods for concurrency control.*
ACM Transactions on Database Systems 1981

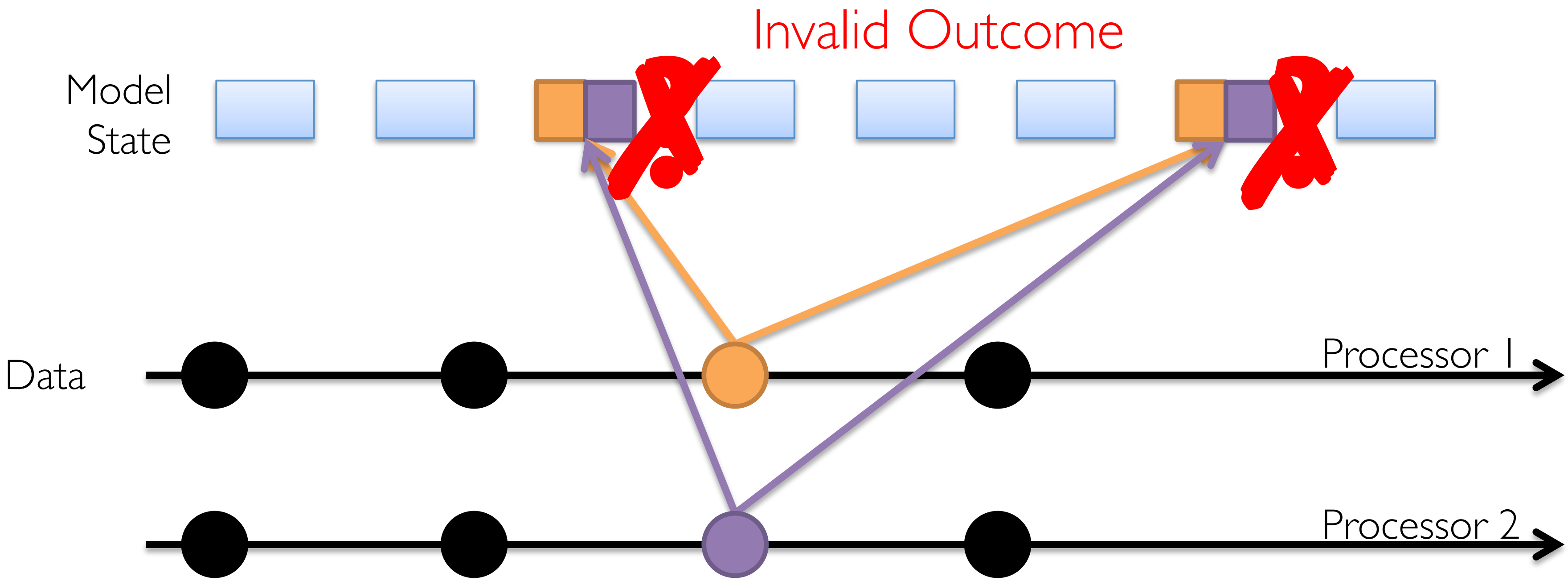
Optimistic Concurrency Control



Validate potential conflicts.

Kung & Robinson. *On optimistic methods for concurrency control.* ACM Transactions on Database Systems 1981

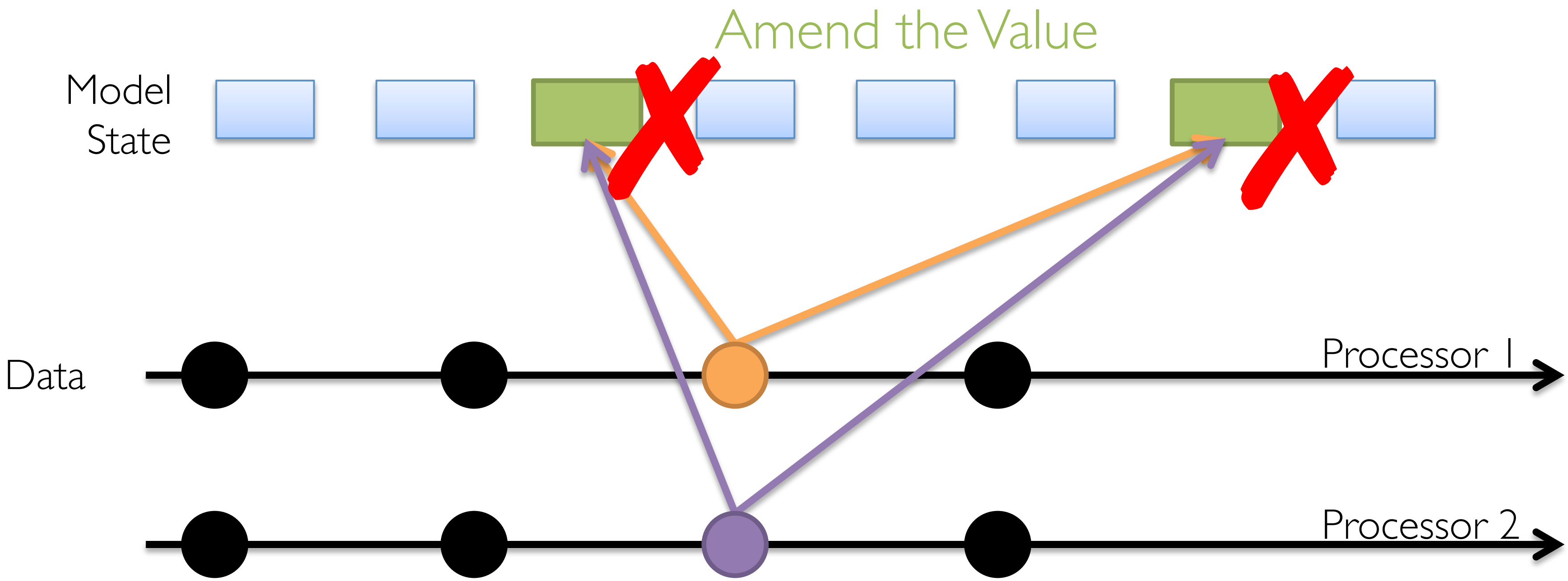
Optimistic Concurrency Control



Validate potential conflicts.

Kung & Robinson. *On optimistic methods for concurrency control.* ACM Transactions on Database Systems 1981

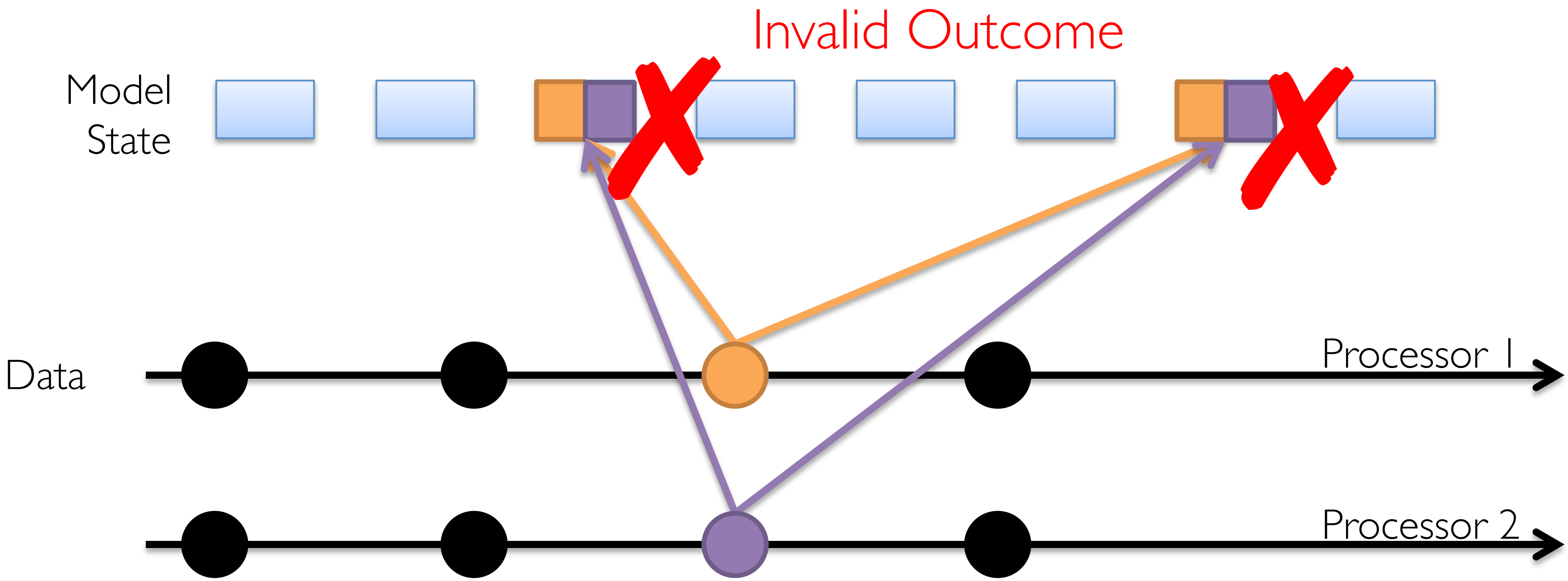
Optimistic Concurrency Control



Take a compensating action.

Kung & Robinson. *On optimistic methods for concurrency control.* ACM Transactions on Database Systems 1981

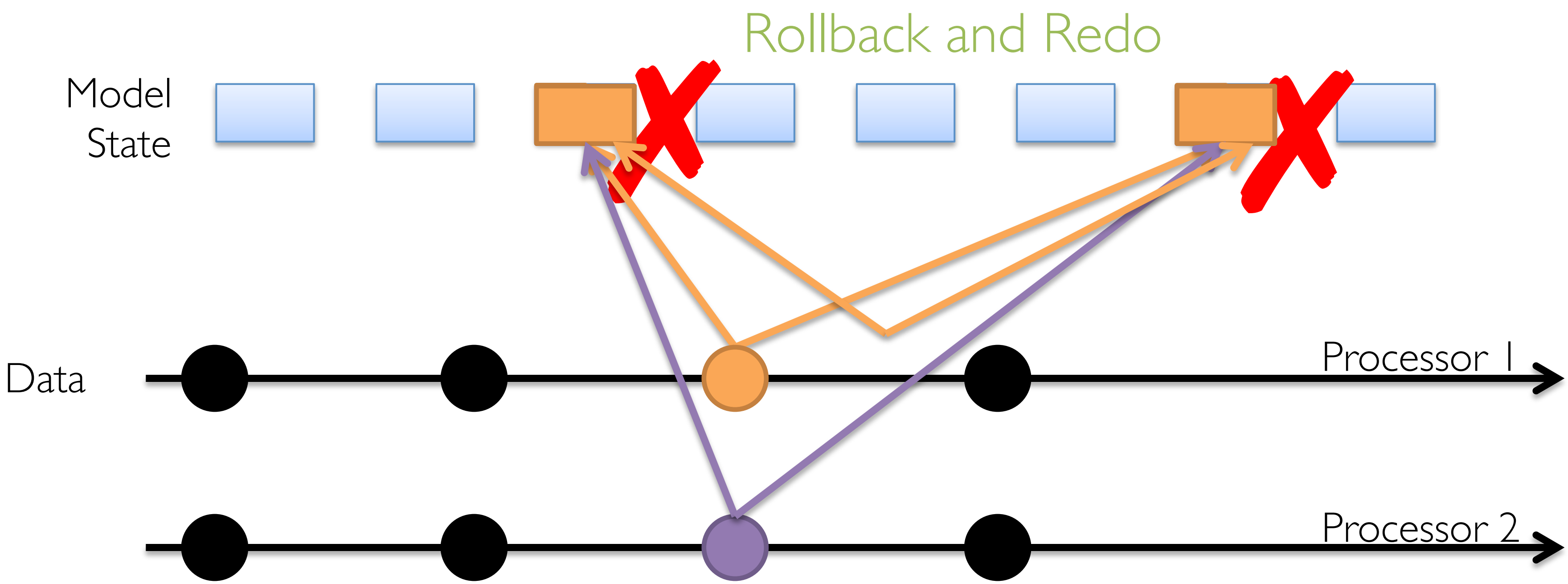
Optimistic Concurrency Control



Validate potential conflicts.

Kung & Robinson. *On optimistic methods for concurrency control.* ACM Transactions on Database Systems 1981

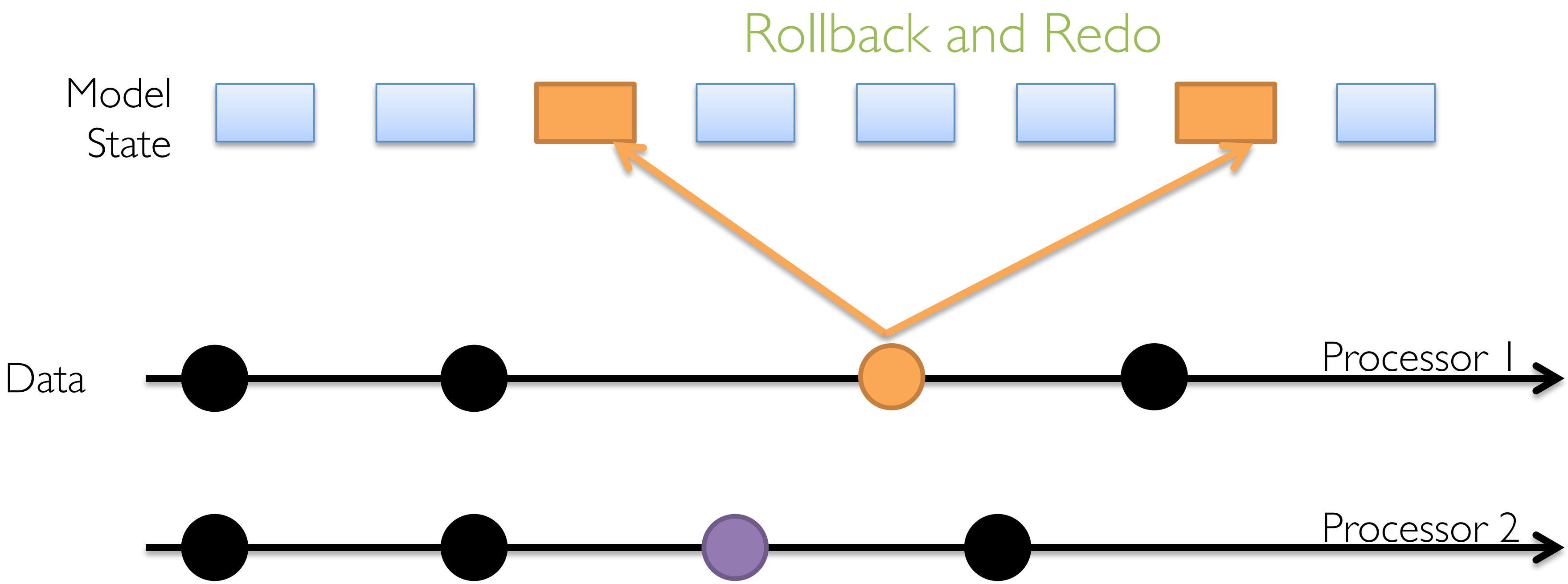
Optimistic Concurrency Control



Take a compensating action.

Kung & Robinson. *On optimistic methods for concurrency control.* ACM Transactions on Database Systems 1981

Optimistic Concurrency Control



Non-Blocking Computation → Concurrency

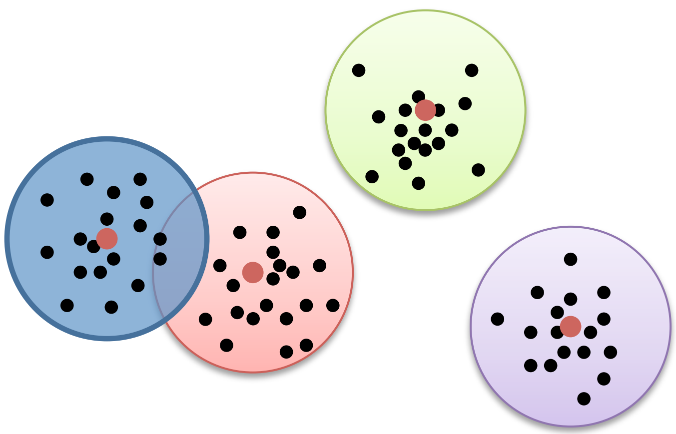
Validation: *Identify Errors*
Resolution: *Correct Errors*
→ Correctness

Optimistic Concurrency Control for Machine Learning

Non-parametric Clustering

Distributed DP-Means

[NIPS'13]



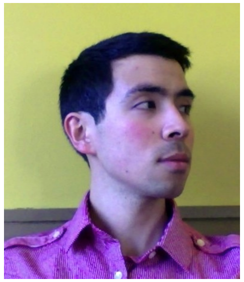
Submodular Optimization

Double Greedy Submodular Maximization

[NIPS'14]

Optimistic Concurrency Control for Submodular Maxmization



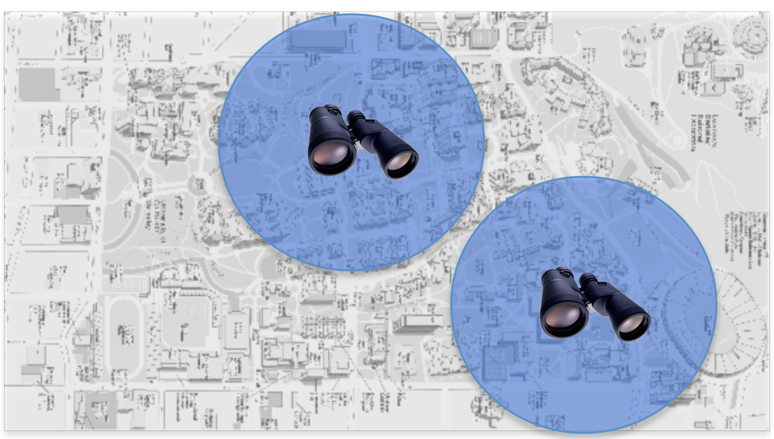

Xinghao Pan, Stefanie Jegelka, Joseph Gonzalez, Joseph Bradley, Michael I. Jordan

Submodular Set Functions

Diminishing Returns Property

F: $2^V \rightarrow \mathrm{R}$, such that for all $A \subset B \subseteq \mathcal{V}$ and $e \notin B$

$$F(A \cup e) - F(A) \quad \geq \quad F(B \cup e) - F(B)$$



Submodular Examples

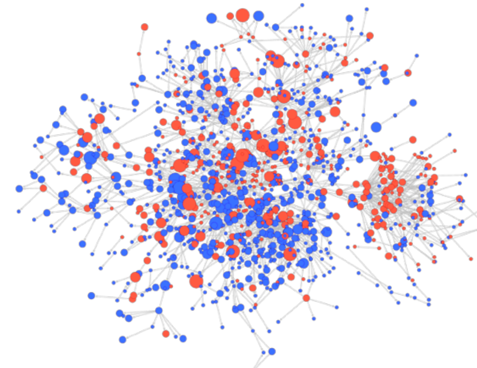
Sensing

$F(S) = \text{area covered by } S$

$$F(S) = I(Y; X_S)$$
$$= \text{reduction in uncertainty}$$

(Krause & Guestrin 2005)

Network Analysis



$$F(S) = \mathbb{E}[\#\text{ active nodes at end}]$$

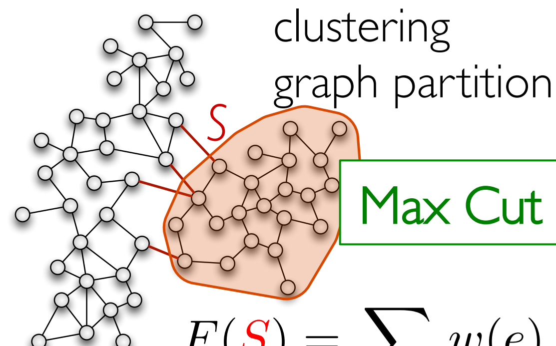
(Kempe, Kleinberg, Tardos 2003, Mossel & Roch 2007)

Document Summarization

(Lin & Bilmes 2011)

Graph Algorithms

clustering
graph partitioning



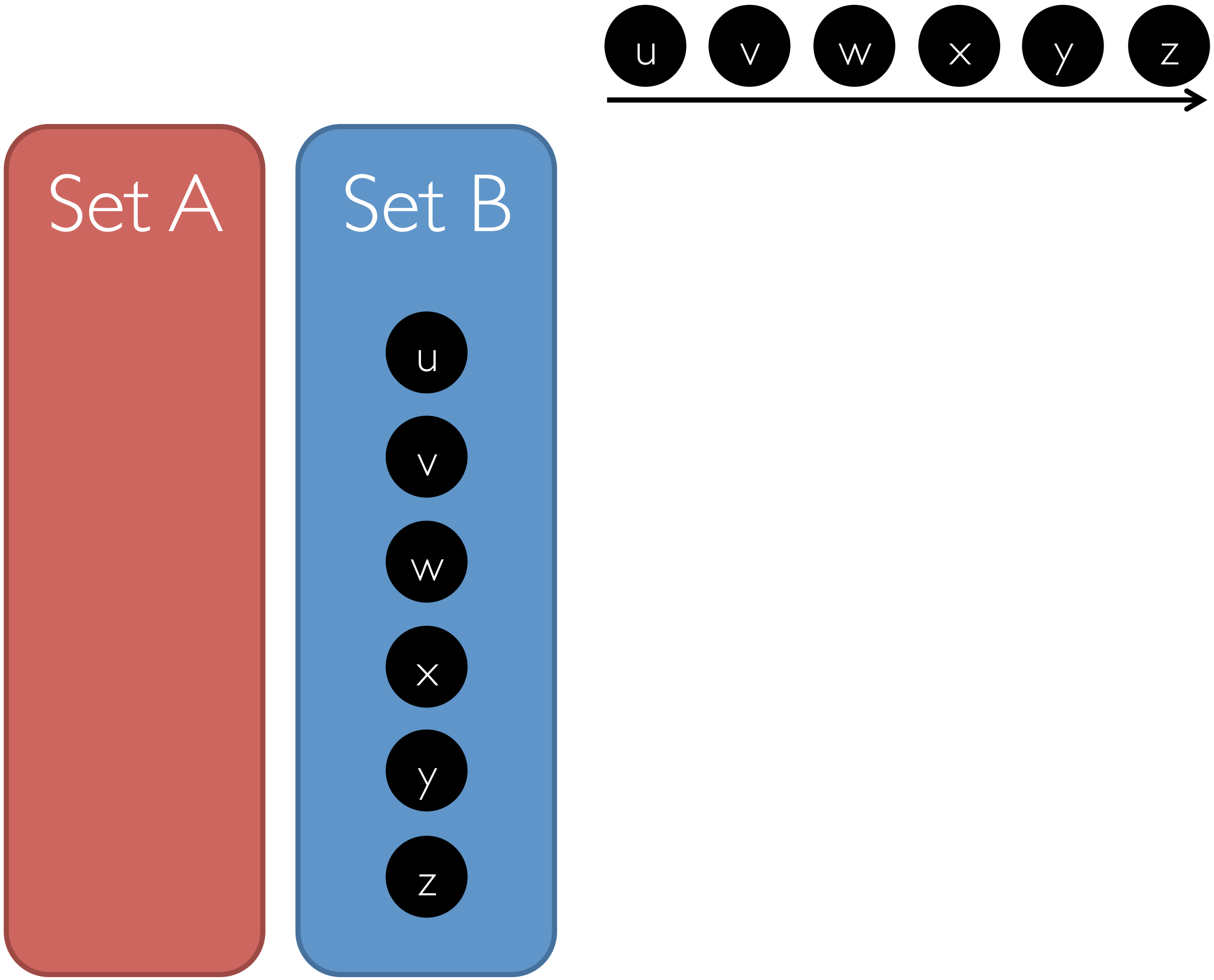
$$F(S) = \sum_{e \text{ cut}} w(e)$$

Submodular Maximization

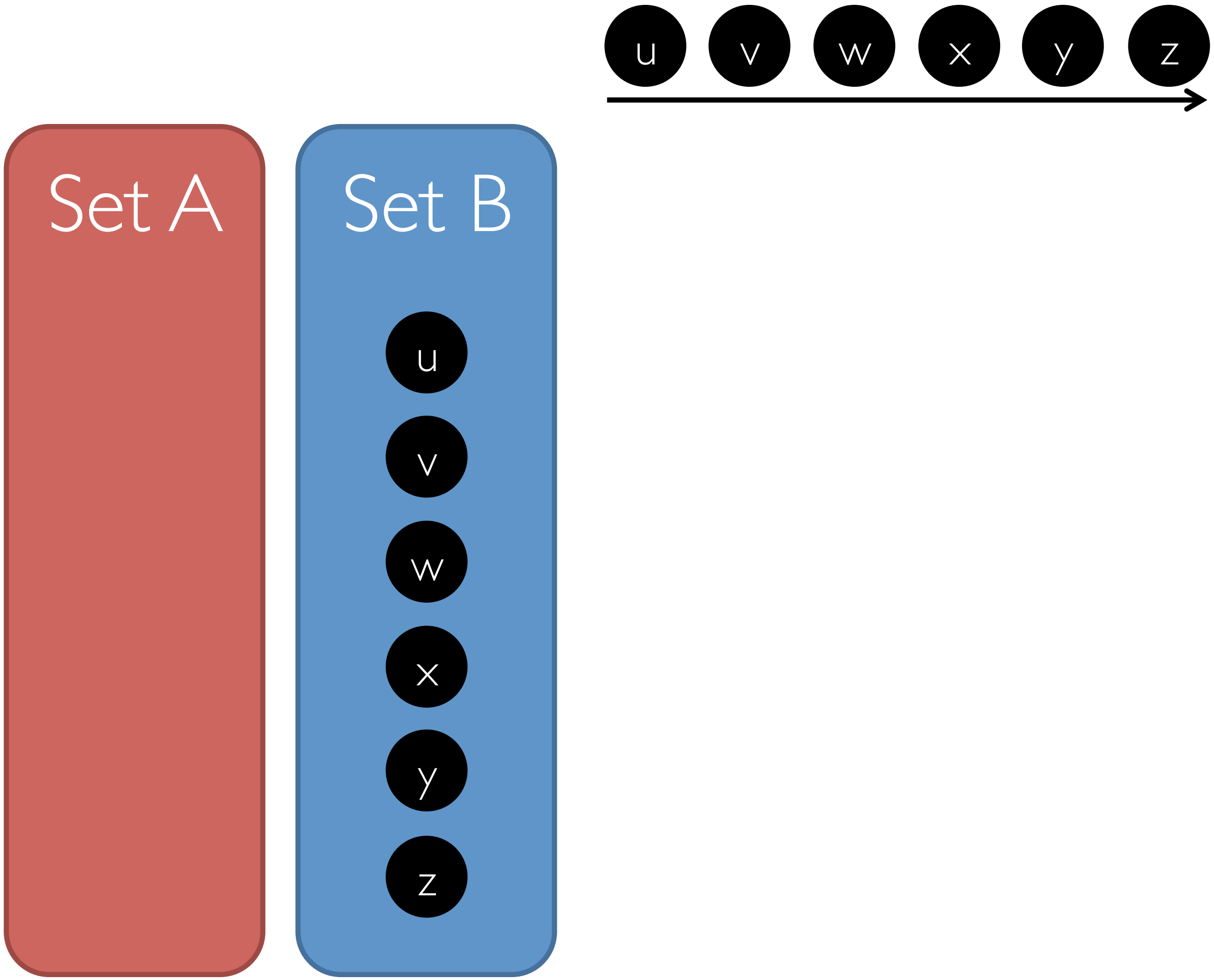
max F(A), A ⊆ V

	Monotone (increasing) functions [Positive marginal gains]	Non-monotone functions
Sequential	*Greedy (Nemhauser et al, 1978)* (1-1/e) - approximation Optimal polytime	*Double Greedy (Buchbinder et al, 2012)* ½ - approximation Optimal polytime
Parallel / Distributed	*GreeDi (Mirzasoleiman et al, 2013)* $(1-1/e)^2 / p$ – approximation 1 MapReduce round *(Kumar et al, 2013)* $1 / (2+\varepsilon)$ – approximation $O(1/\varepsilon)$ MapReduce rounds	Concurrency Control Double Greedy Optimal ½ - approximation Bounded overhead Coordination Free Double Greedy Bounded error Minimal overhead

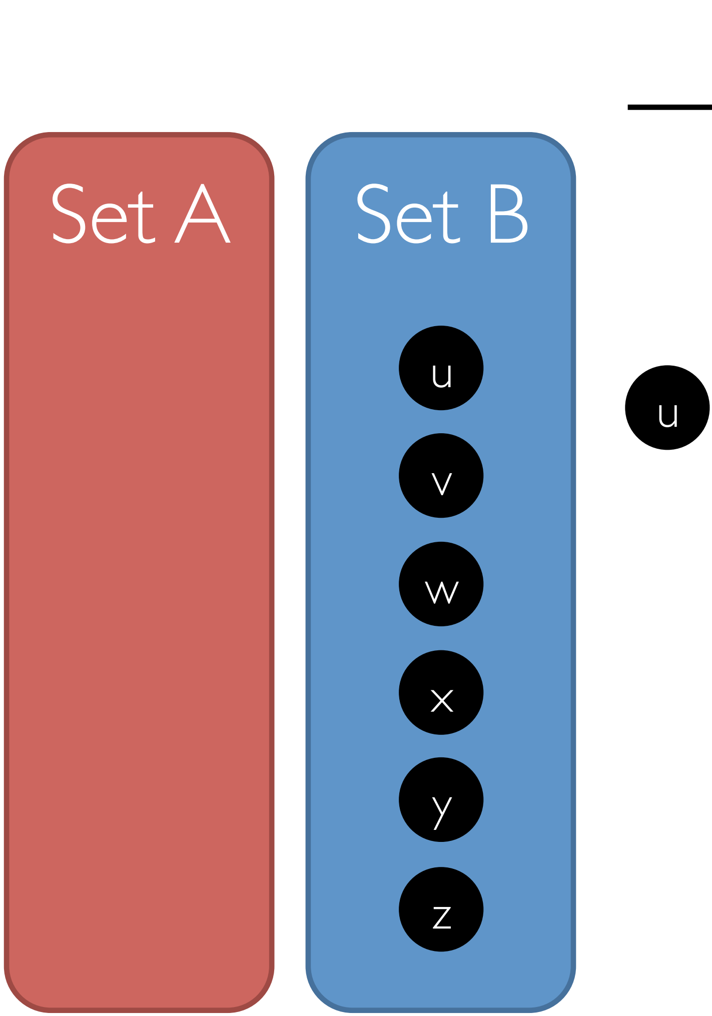
Double Greedy Algorithm



Double Greedy Algorithm



Double Greedy Algorithm

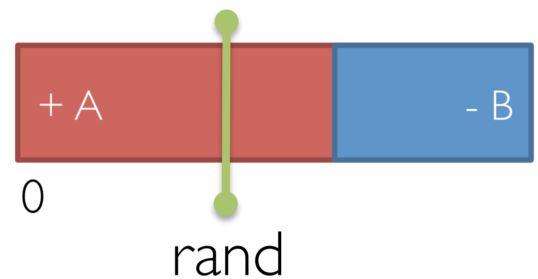


Marginal gains

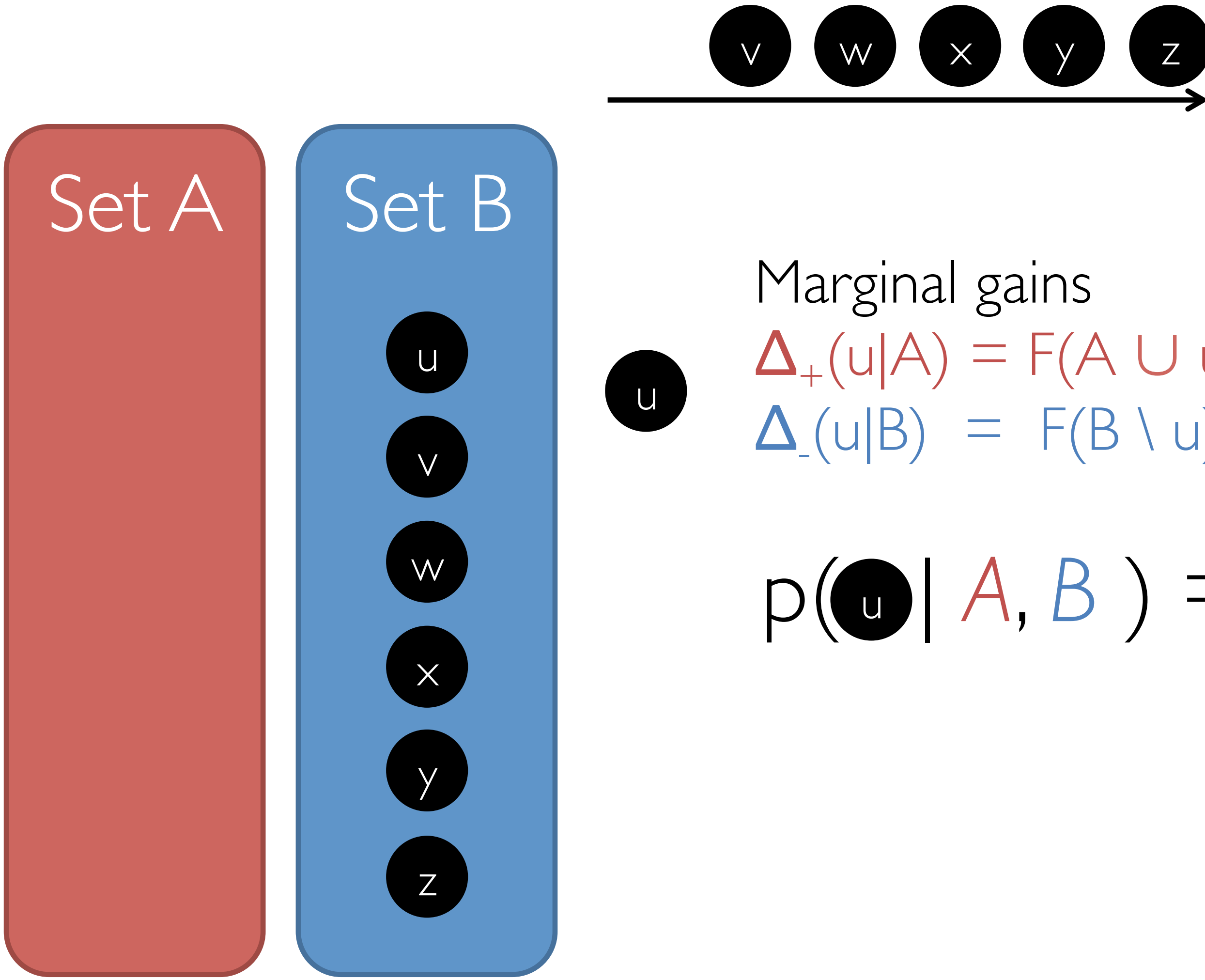
$\Delta_+(u|A) = F(A \cup u) - F(A),$

$\Delta_-(u|B) = F(B \setminus u) - F(B).$

$p(u \mid A, B) =$



Double Greedy Algorithm

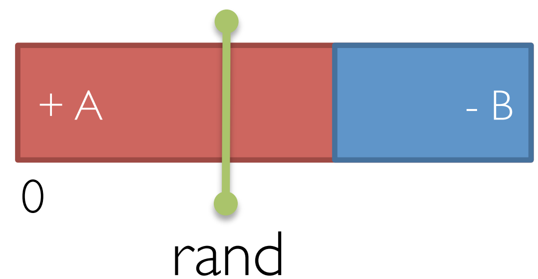


Marginal gains

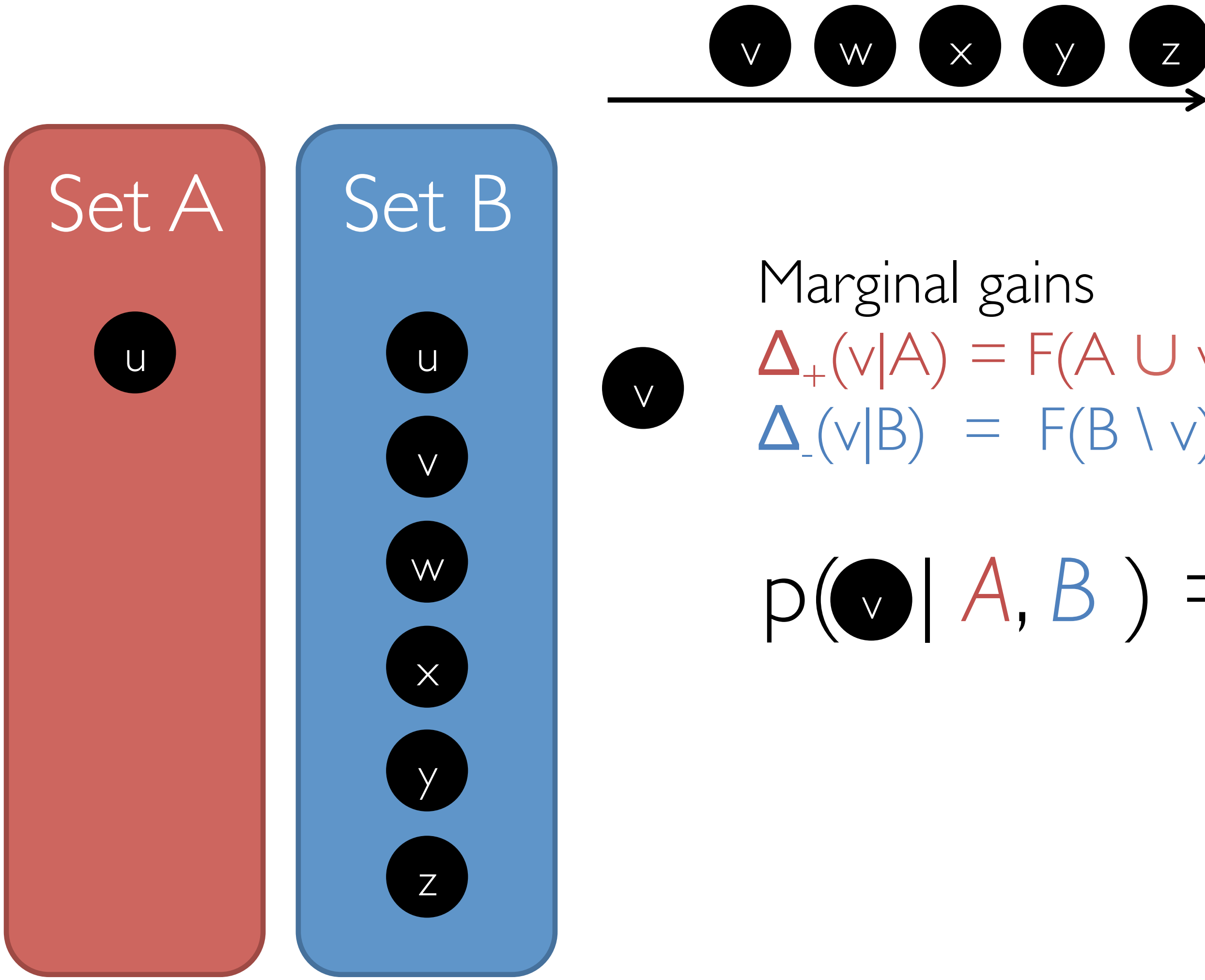
$\Delta_{+}(u|A) = F(A \cup u) - F(A),$

$\Delta_{-}(u|B) = F(B \setminus u) - F(B).$

$p(u \mid A, B) =$



Double Greedy Algorithm

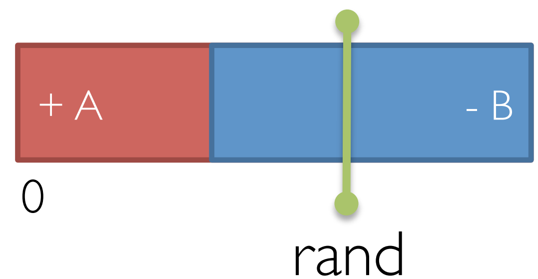


Marginal gains

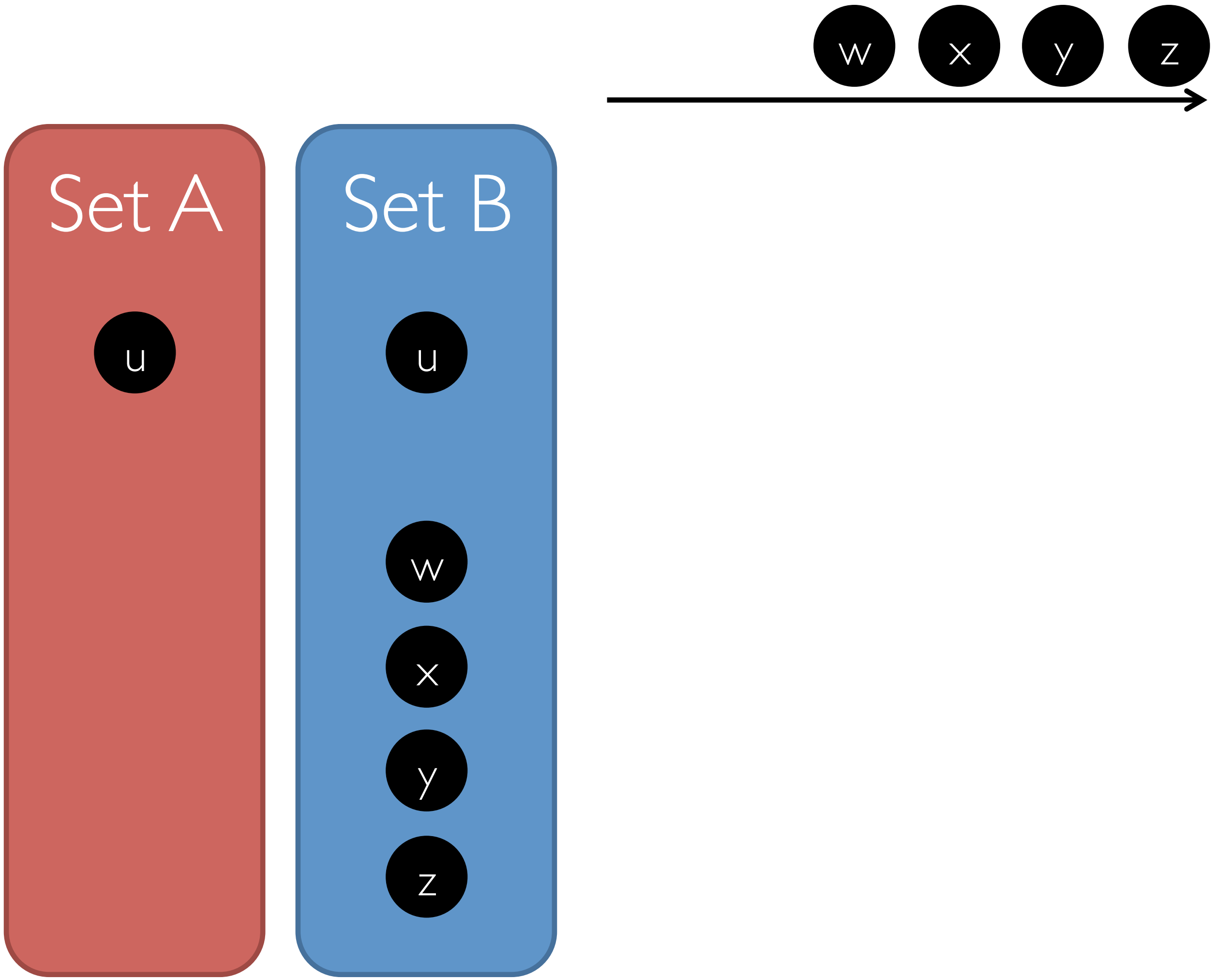
$\Delta_+(v|A) = F(A \cup v) - F(A),$

$\Delta_-(v|B) = F(B \setminus v) - F(B).$

$p(v \mid A, B) =$

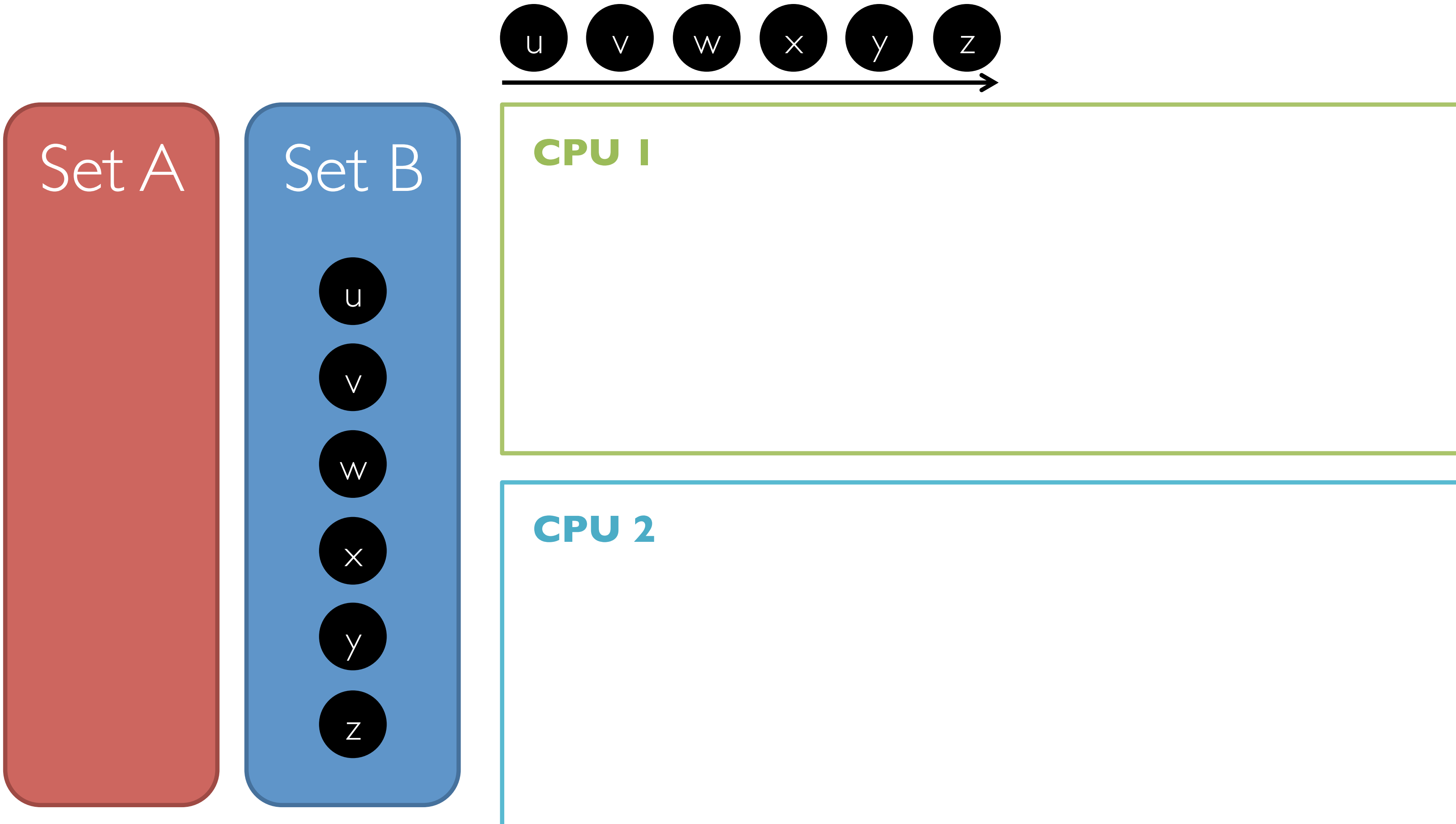


Double Greedy Algorithm

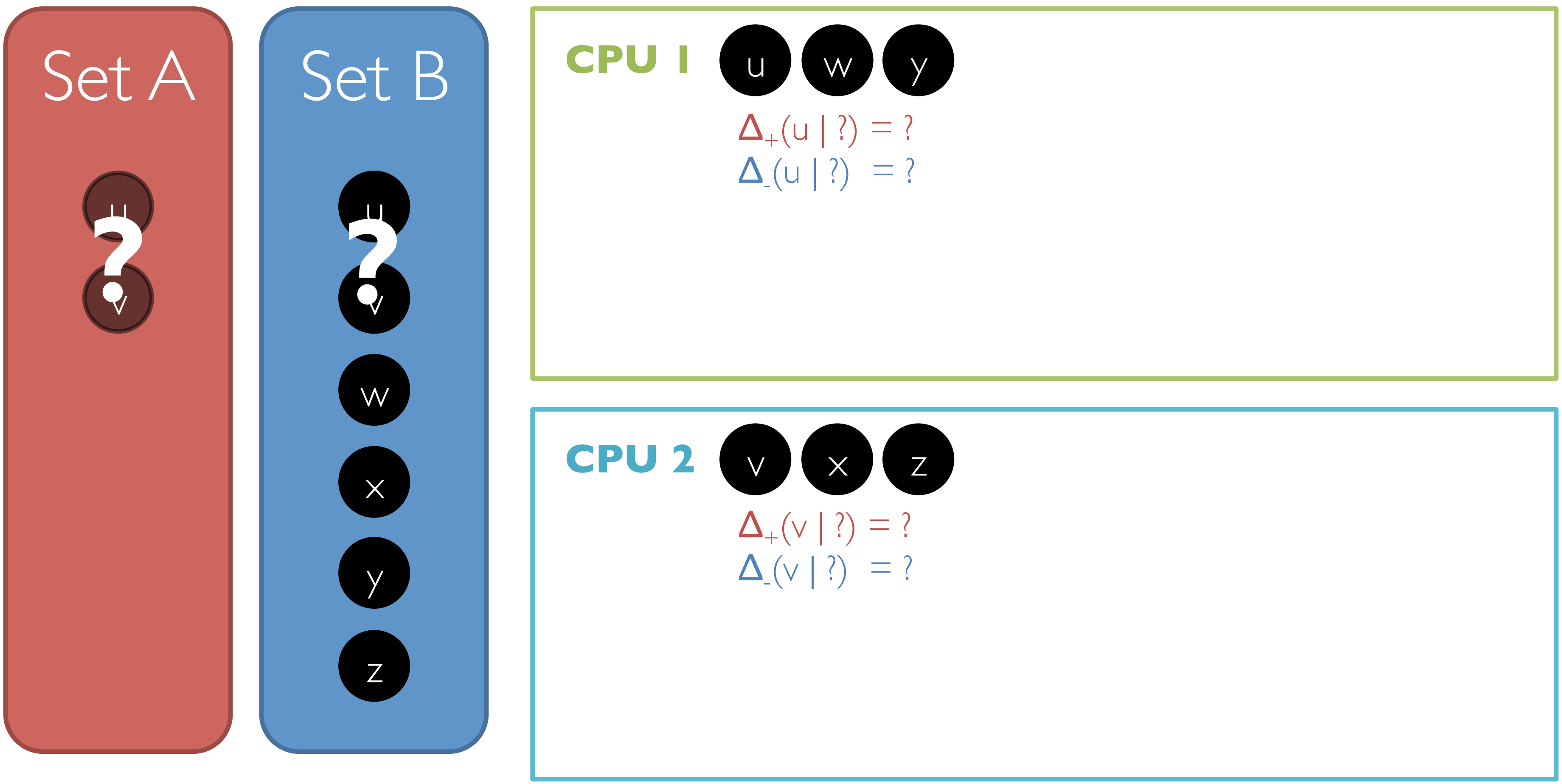


Return A

Parallel Double Greedy Algorithm

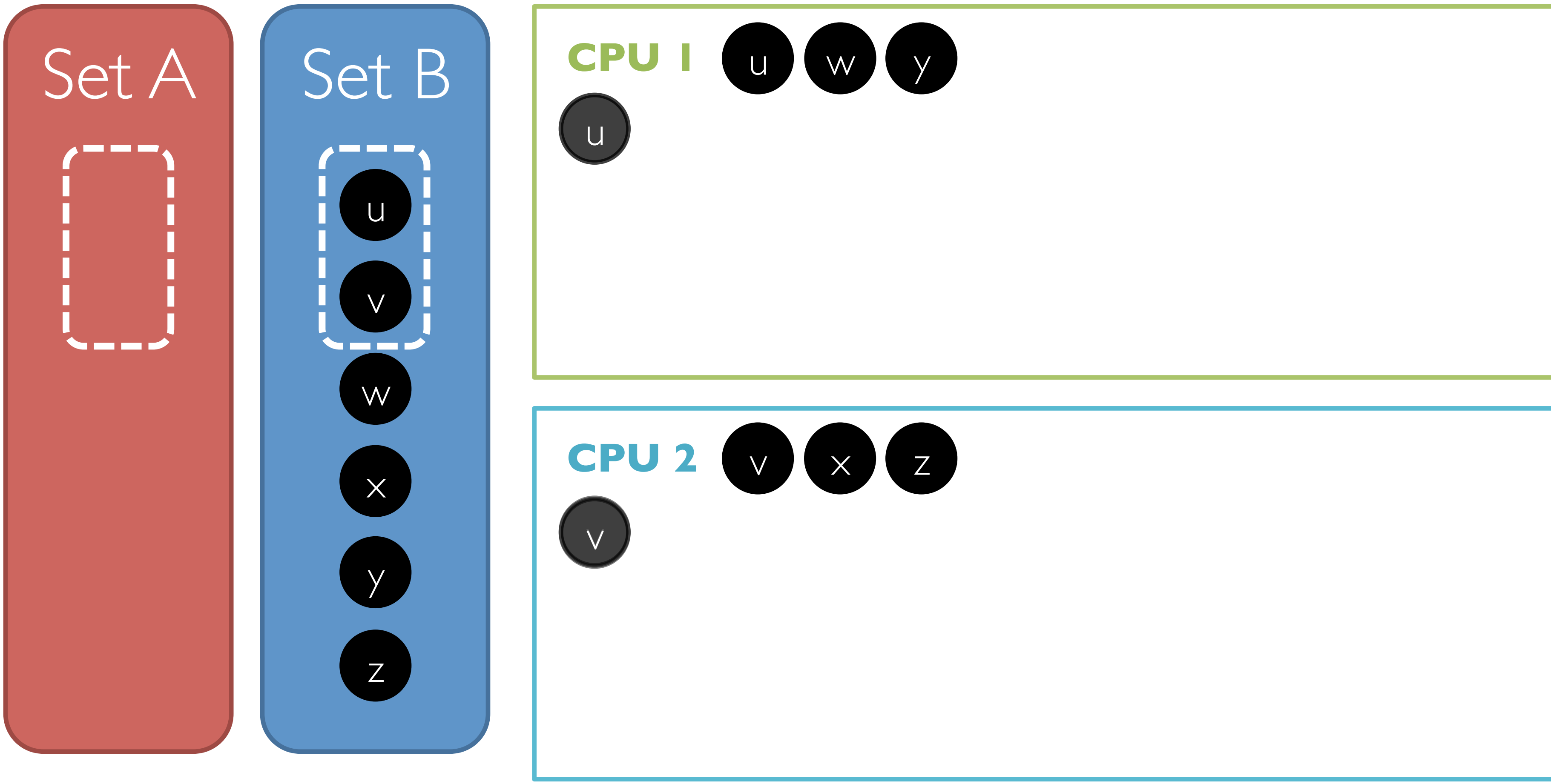


Parallel Double Greedy Algorithm



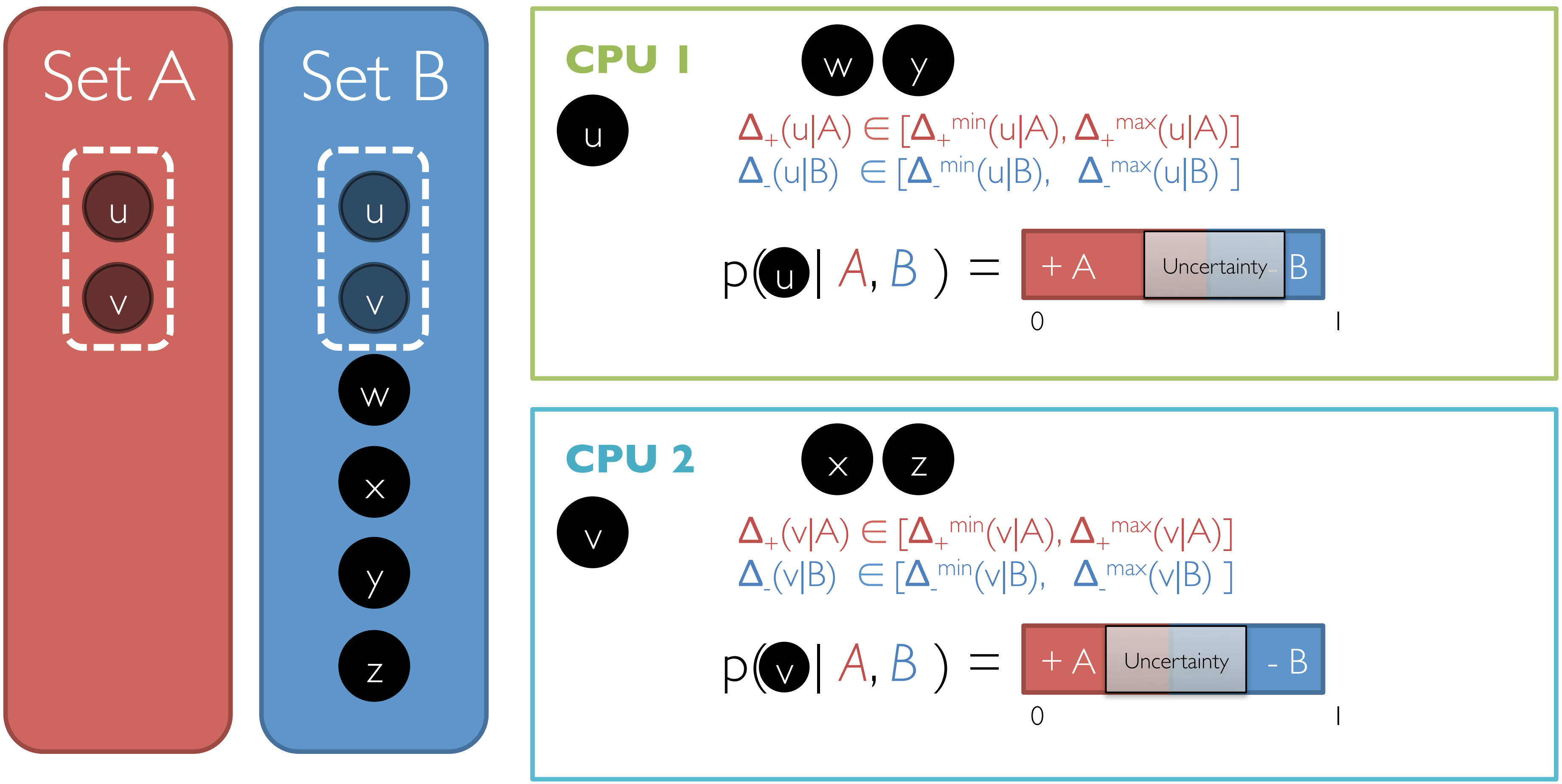
Concurrency Control Double Greedy

Maintain bounds on A, B ➔ Enable threads to make decisions locally



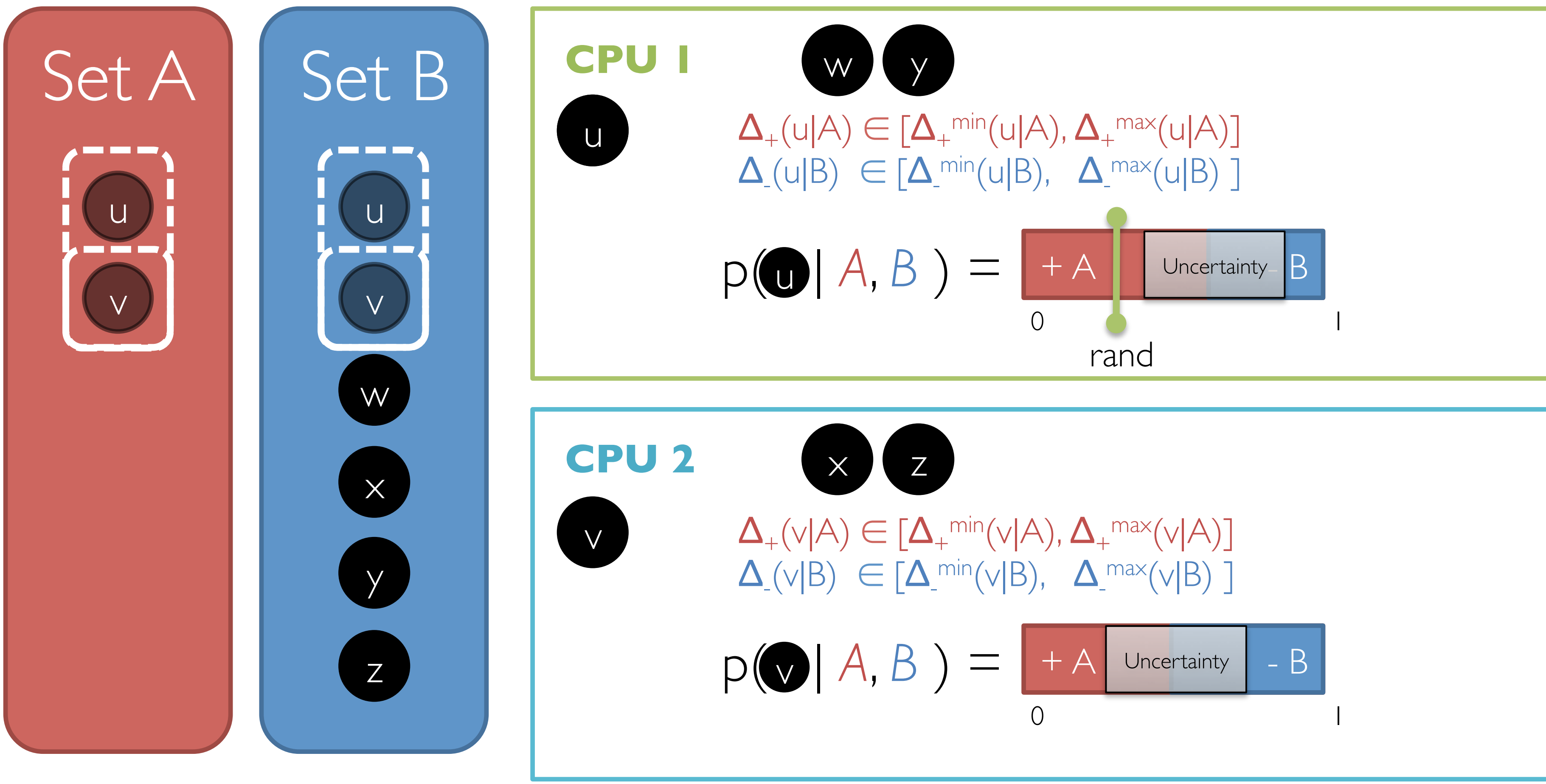
Concurrency Control Double Greedy

Maintain bounds on A, B ➔ Enable threads to make decisions locally



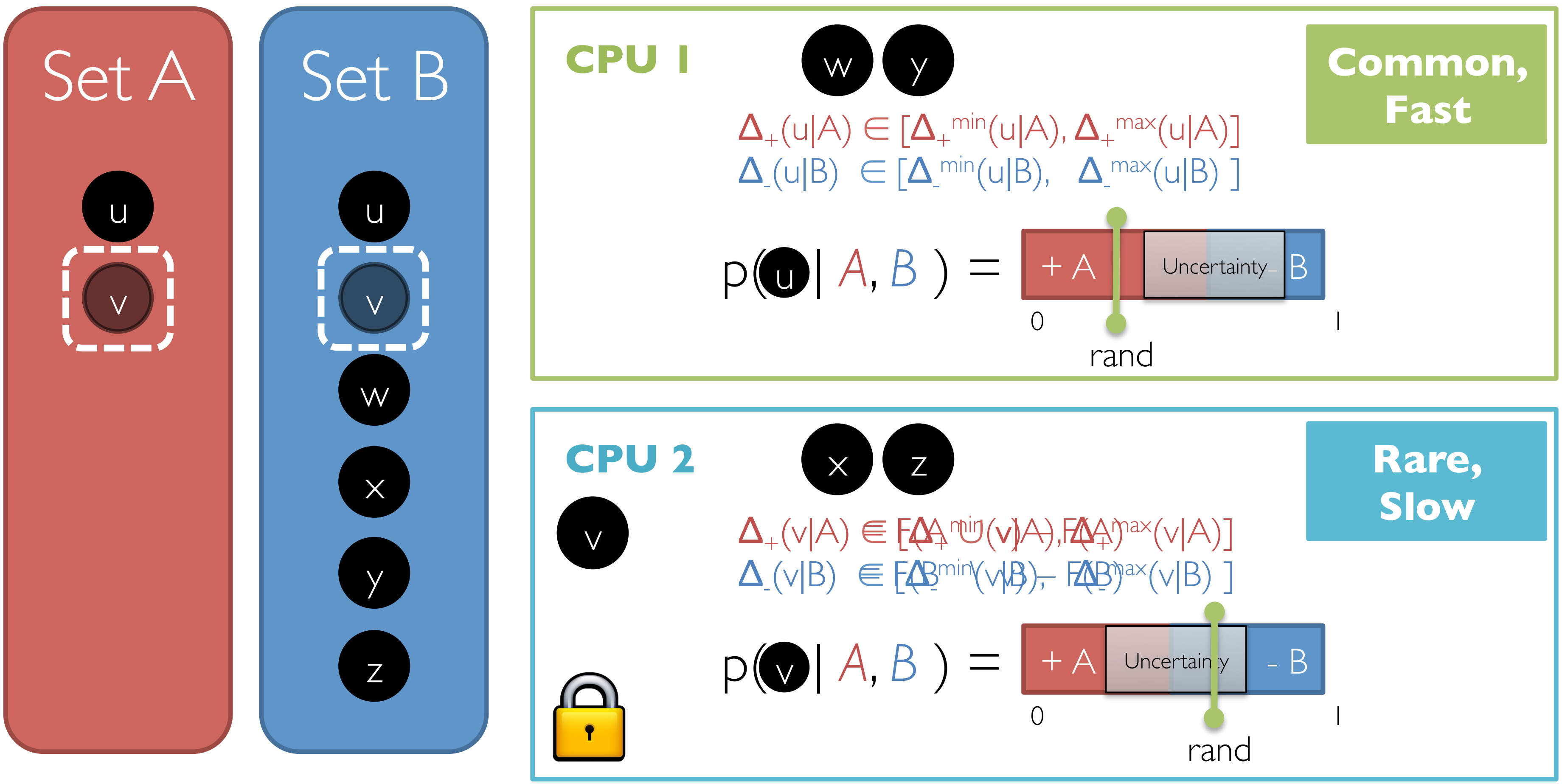
Concurrency Control Double Greedy

Maintain bounds on A, B ➔ Enable threads to make decisions locally



Concurrency Control Double Greedy

Maintain bounds on A, B ➔ Enable threads to make decisions locally



Properties of CC Double Greedy

Correctness

Theorem: CC double greedy is serializable.

Corollary: CC double greedy <u>preserves optimal approximation guarantee</u> of ½OPT.

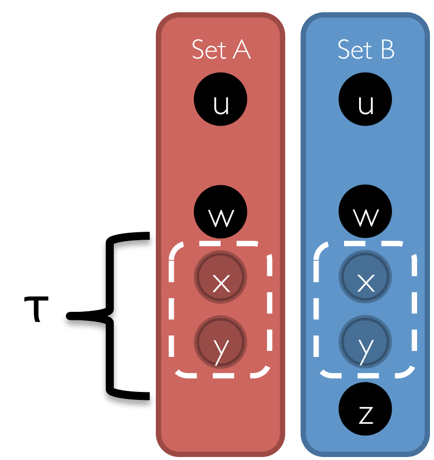
Concurrency

Lemma: CC has bounded overhead.

Expected number of blocked elements

set cover with costs: $< 2\tau$

sparse max cut: $< 2\tau\,|E|\,/\,|V|$



Change in Analysis

Coordination Free:

Provably fast and correct under key assumptions.

Concurrency Control:

Provably correct and fast under key assumptions.

Correctness	Scalability
Easy Proof	Challenging Proof

Empirical Validation

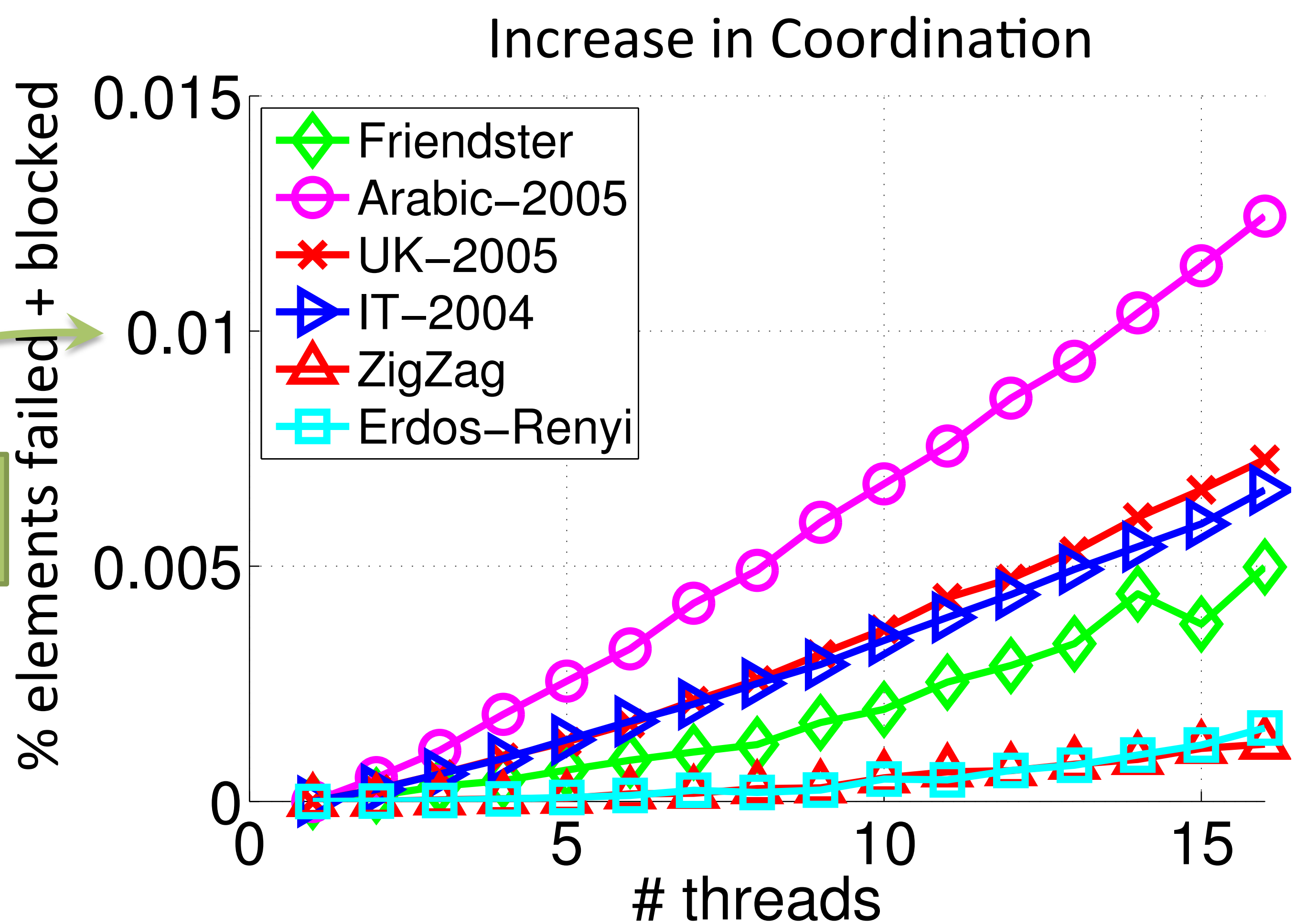
Multicore up to 16 threads

Set cover, Max graph cut

Real and synthetic graphs

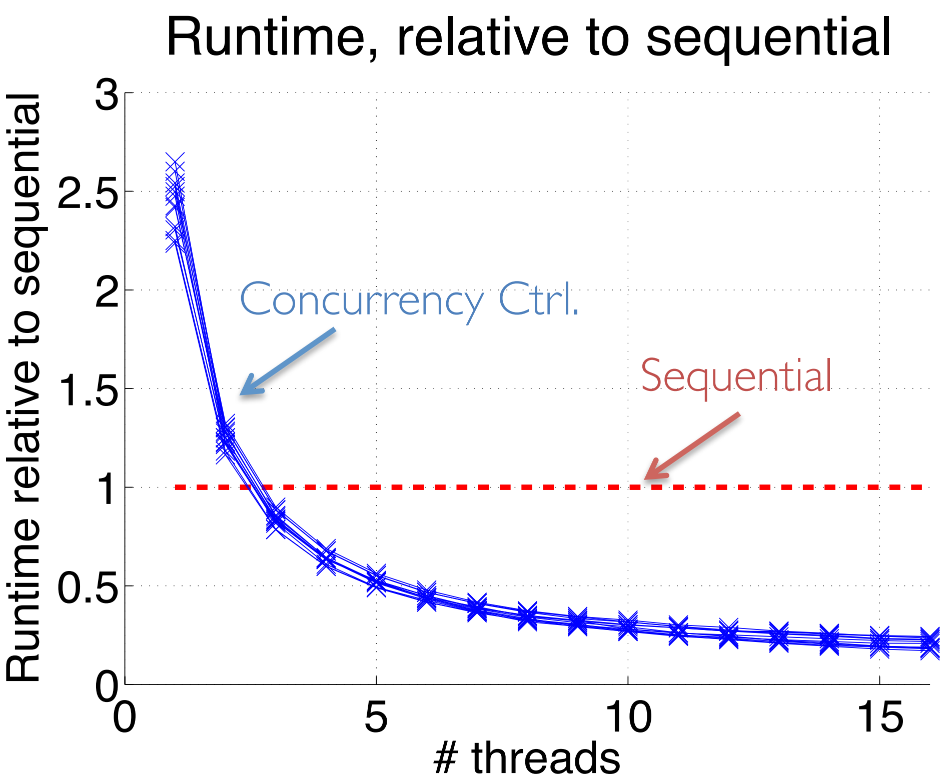
Graph		Vertices	Edges
IT-2004	Italian web-graph	41 Million	1.1 Billion
UK-2005	UK web-graph	39 Million	0.9 Billion
Arabic-2005	Arabic web-graph	22 Million	0.6 Billion
Friendster	Social sub-network	10 Million	0.6 Billion
Erdos-Renyi	Synthetic random	20 Million	2.0 Billion
ZigZag	Synthetic expander	25 Million	2.0 Billion

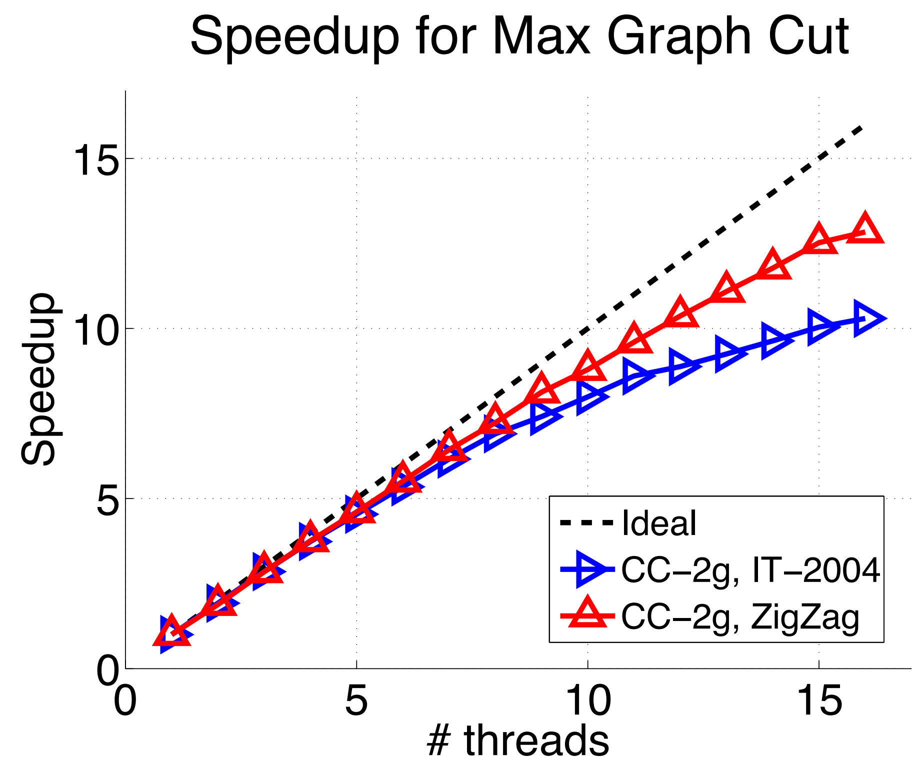
CC Double Greedy Coordination



Only 0.01% needs blocking!

Runtime and Strong-Scaling





Conclusion

	Scalability	Approximation
Sequential Double Greedy	Always slow	**Always optimal**
Concurrency Control Double Greedy	Usually fast	**Always optimal**
Coordination Free Double Greedy	**Always fast**	Near optimal

Paper @ NIPS 2014:

Parallel Double Greedy Submodular Maximization.

BACKUP SLIDES

Concurrency Control

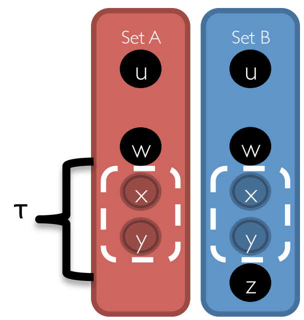
Correctness

Theorem: serializable.
preserves optimal
approximation bound
½OPT.

Concurrency

Lemma:
bounded overhead.

set cover with costs: 2τ
sparse maxcut: $2|E|\tau \,/\, |V|$



from same dependencies
(uncertainty region)

Coordination Free

Correctness

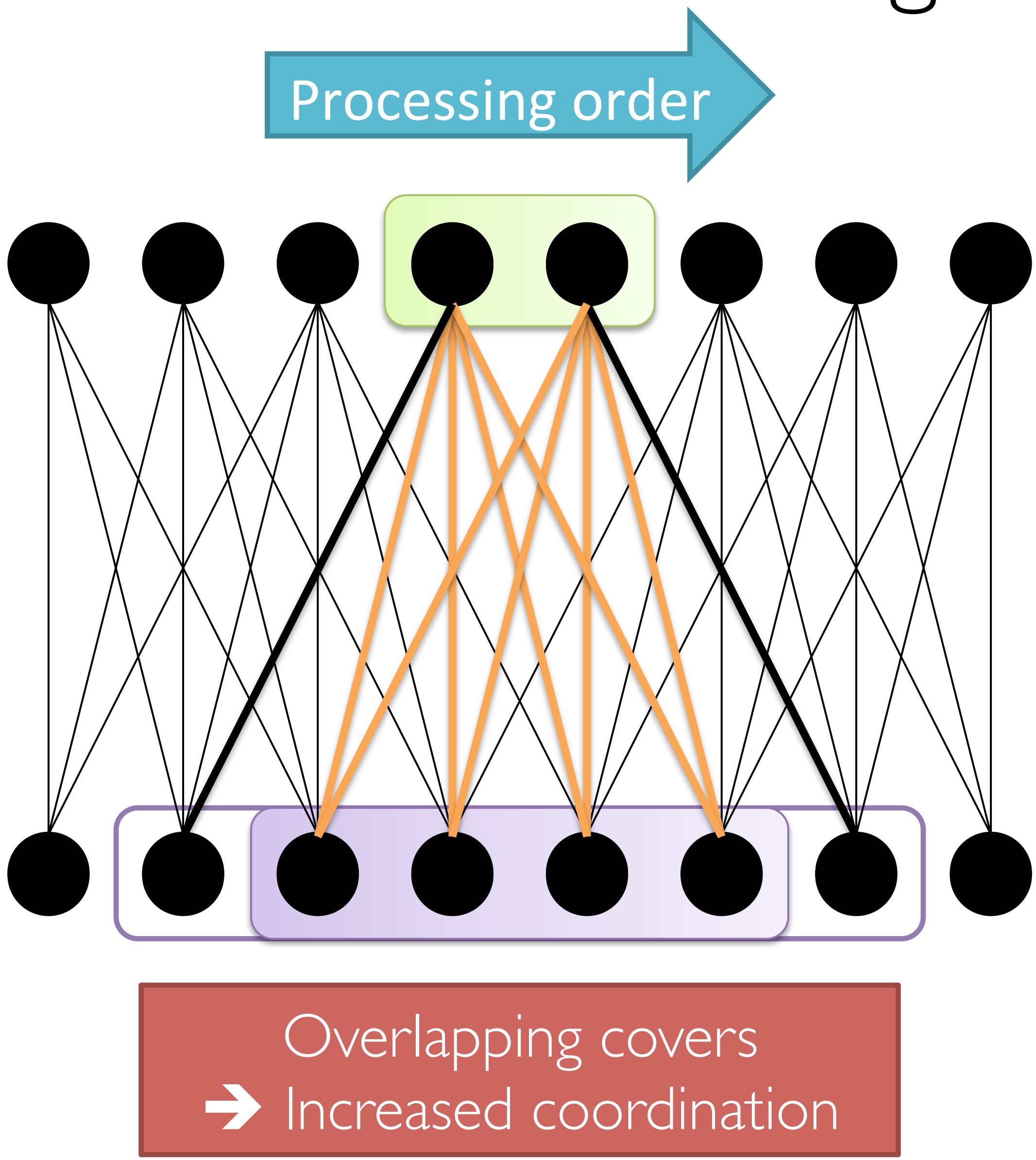
Lemma:
approximation bound
½OPT - error

set cover with costs: $\geq \tau$
sparse maxcut: $|E|\tau \,/\, 2|V|$

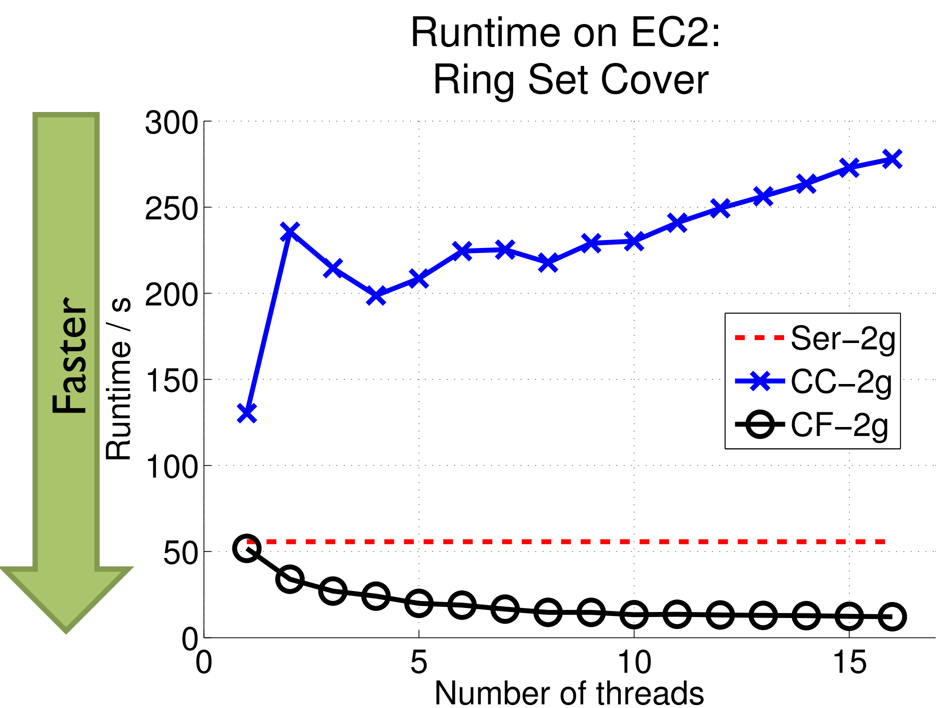
Concurrency

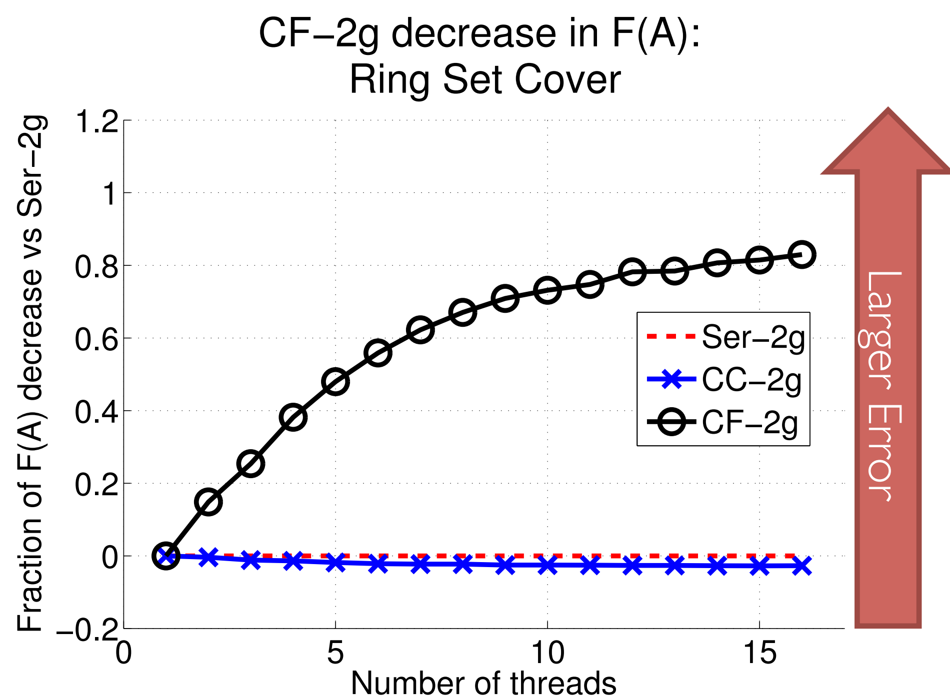
no
overhead

Adversarial Setting



Adversarial Setting – Ring Set Cover





Coord Free	Always fast	Possibly wrong
Conc Ctrl	Possibly slow	Always optimal