CoRAL: Confined Recovery in Distributed Asynchronous Graph Processing

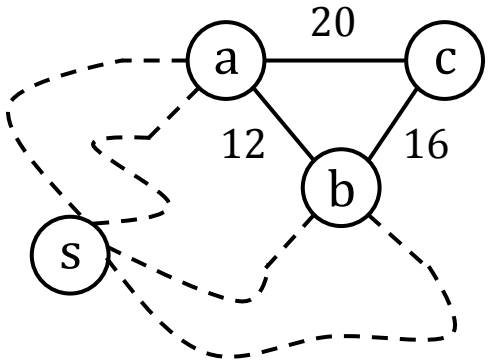
Keval Vora, Chen Tian, Rajiv Gupta and Ziang Hu

UNIVERSITY OF CALIFORNIA
UCRIVERSIDE

HUAWEI

Graph Processing

- Iterative graph algorithms

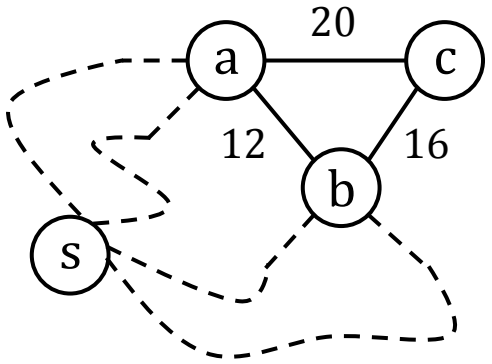


$$v.path \leftarrow \min_{e \in \text{inEdges}(v)} (e.source.path + e.weight)$$

Iter	s	a	b	c
0	0	∞	∞	∞
…				
i	0	5	10	35
i+1	0	4	6	25
i+2	0	1	5	22
…				

Graph Processing

- Iterative graph algorithms
- Asynchronous
- Highly parallel execution



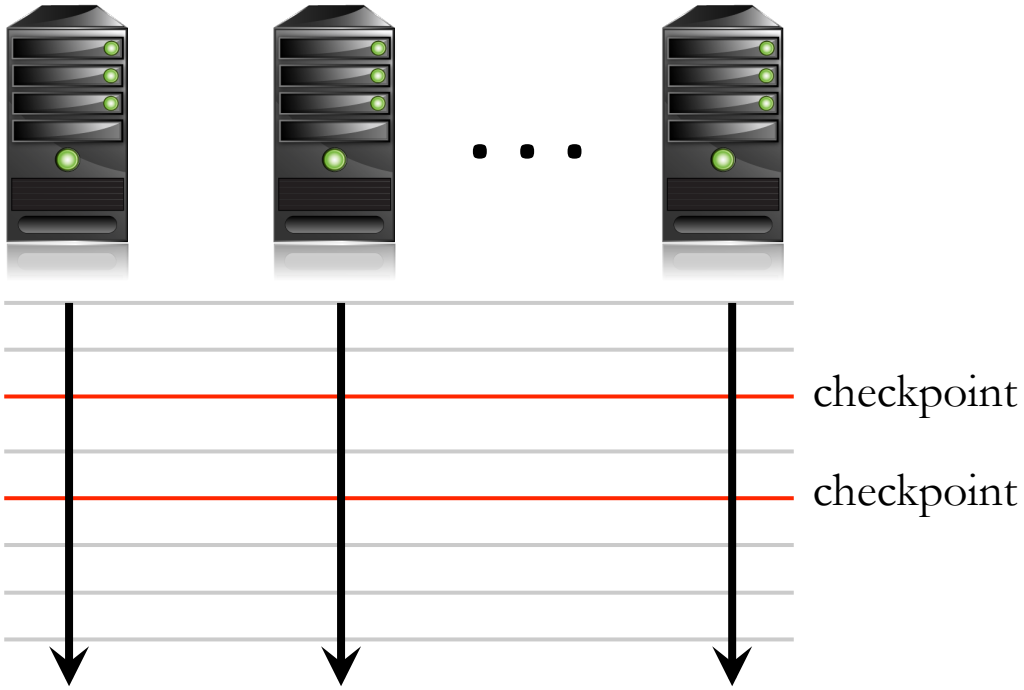
Iter	s	a	b	c
i	0	5	10	35
i+1	0	4	6	25
i+2			5	21

Iter	s	a	b	c
i	0	5	10	35
i+1	0	4	6	25
i+2		1		21

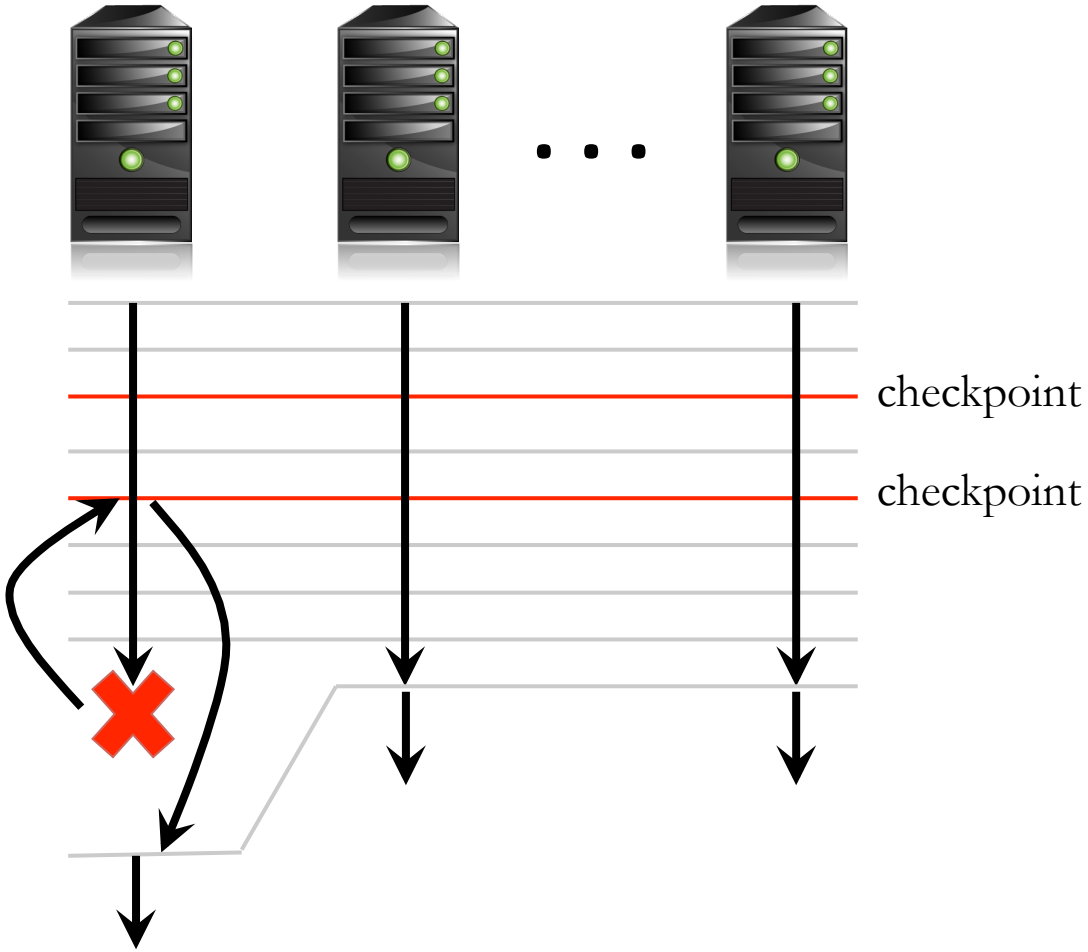
Iter	s	a	b	c
0	0	∞	∞	∞
…				
i	0	5	10	35
i+1	0	4	6	25
i+2				22

Iter	s	a	b	c
i	0	5	10	35
i+1	0	4	6	25
i+2	0	1	5	21

Fault Tolerance



Fault Tolerance

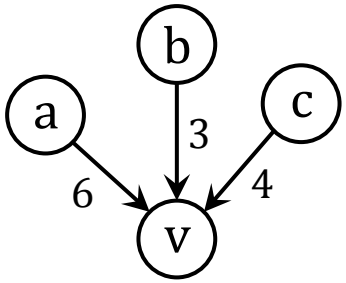


Outline

- Progressive Reads
- Single machine failure
- Multiple machine failure
- Locally consistent checkpointing

Progressive Reads

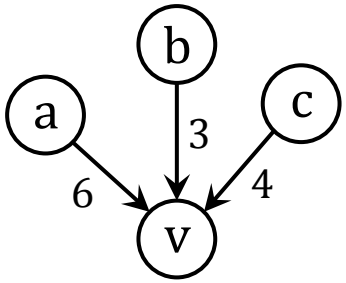
$$v.path \leftarrow \min_{e \in \text{inEdges}(v)} (e.source.path + e.weight)$$



Iter	a	b	c	v
0	∞	∞	∞	∞
..	..	..	..	..
i-1	5	10	13	18
i	4	7	6	10
i+1		5	6	10
i+2			5	

Progressive Reads

$$v.path \leftarrow \min_{e \in \text{inEdges}(v)} (e.source.path + e.weight)$$

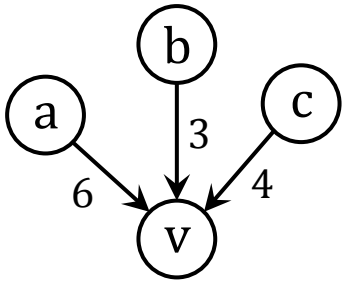


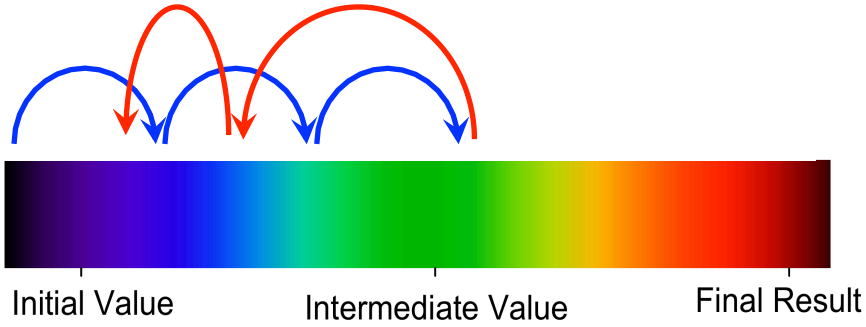
Iter	a	b	c	v
0	∞	∞	∞	∞
..	..	..	..	..
i-1	5	10	13	18
i	4	7	6	10
i+1		5	6	10
i+2			5	8

Progressive Reads

- Convergence
- Correctness

$$v.path \leftarrow \min_{e \in \text{inEdges}(v)} (e.source.path + e.weight)$$

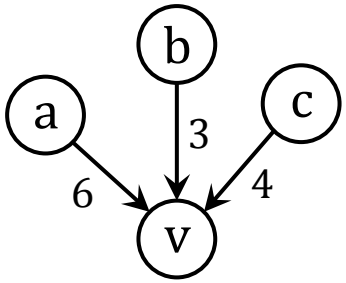




Iter	a	b	c	v
0	∞	∞	∞	∞
..	..	..	..	..
i-1	5	10	13	18
i	4	7	6	10
i+1		5	6	10
i+2			5	8
i+3				9

Progressive Reads

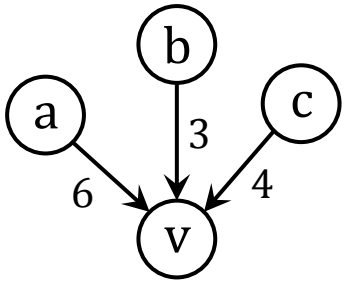
$$v.path \leftarrow \min_{e \in \text{inEdges}(v)} (e.source.path + e.weight)$$



Iter	a	b	c	v
0	∞	∞	∞	∞
..	..	..	..	..
i-1	5	10	13	18
i	4	7	6	10
i+1		5	6	10
i+2			5	8

Progressive Reads

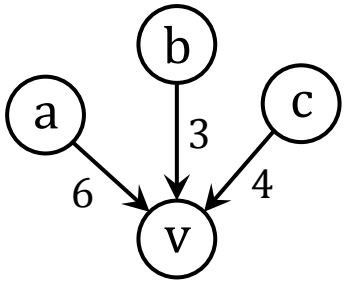
$$v.path \leftarrow \min_{e \in \text{inEdges}(v)} (e.source.path + e.weight)$$



Iter	a	b	c	v
0	∞	∞	∞	∞
..	..	..	..	..
i-1	5	10	13	18
i	4	7	6	10
i+1		5	6	10
i+2			5	8

Progressive Reads

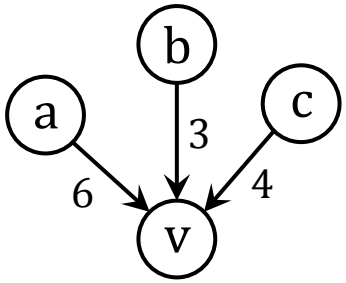
$$v.path \leftarrow \min_{e \in \text{inEdges}(v)} (e.source.path + e.weight)$$



Iter	a	b	c	v
0	∞	∞	∞	∞
..	..	..	..	..
i-1	5	10	13	18
i	4	7	6	10
i+1		5	6	10
i+2			5	8

Progressive Reads

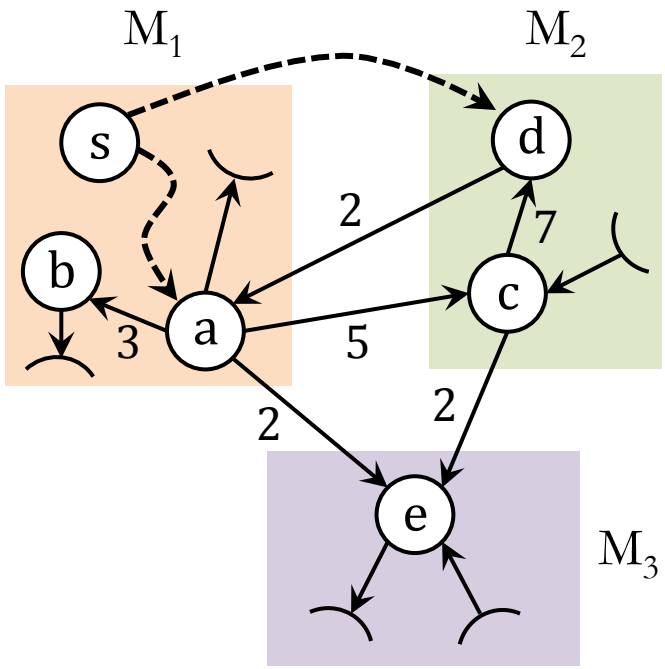
- Same or newer value [OOPSLA'14]



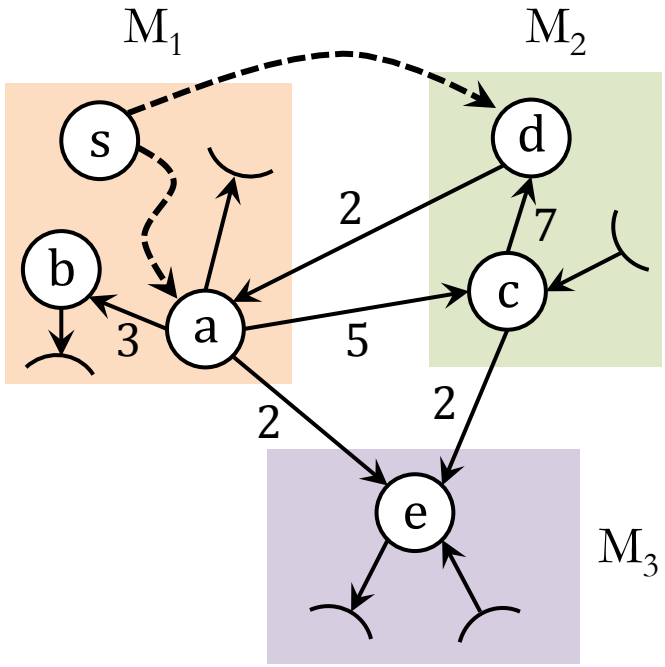
Iter	a	b	c	v
0	∞	∞	∞	∞
..	..	..	..	..
i-1	5	10	13	18
i	4	7	6	10
i+1		5	6	10
i+2			5	8
i+3				

Checkpointing & Recovery

	a	***b***	***c***	***d***	***e***
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3	22	28	30	19	27
4	21	25	27	18	24
5	20	24	26	17	22



Checkpointing & Recovery

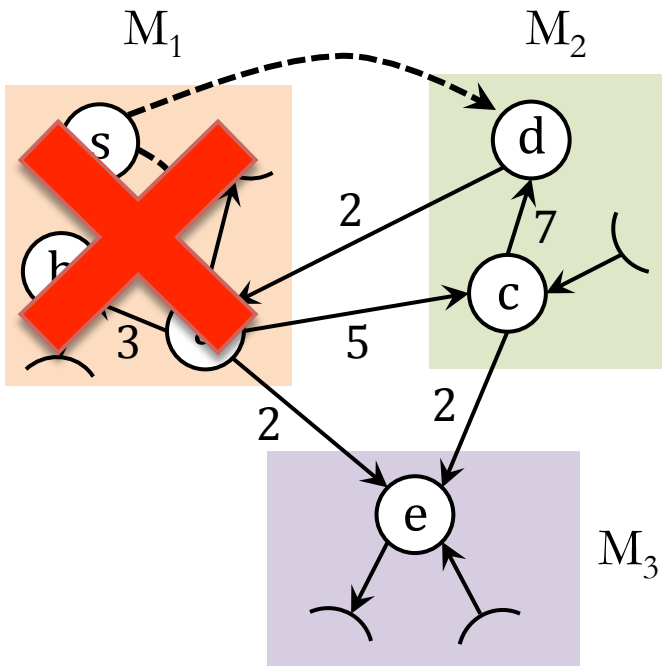


	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3	22	28	30	19	27
4	21	25	27	18	24
5	20	24	26	17	22

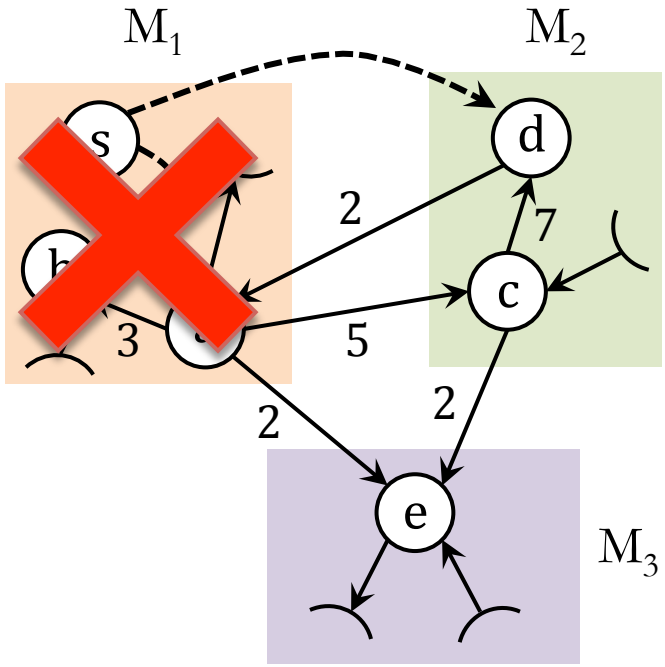
- Chandy-Lamport asynchronous snapshot [TOCS'85]

Checkpointing & Recovery

	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3			30	19	27
4			27	18	24
5			26	17	22
M_1 DIES					
2	25	36	38	20	35
3	22	28	30	18	27



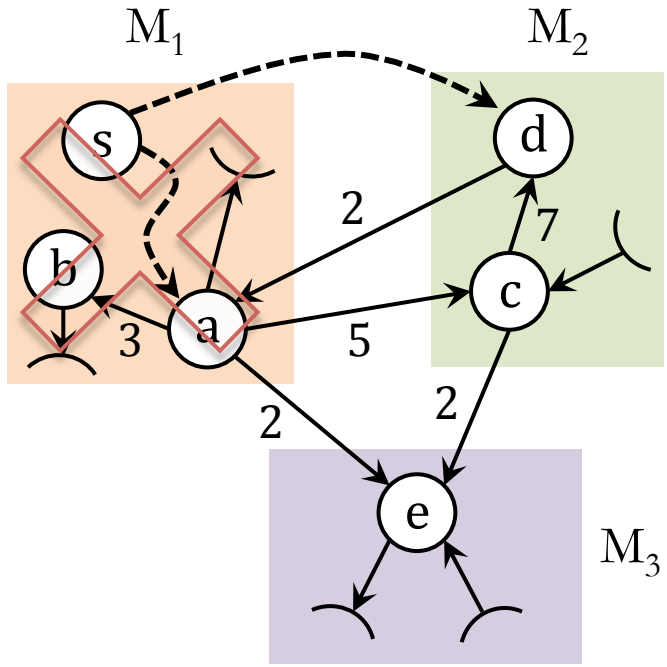
Checkpointing & Recovery



	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3			30	19	27
4			27	18	24
5			26	17	22
	M_1 DIES				
2	25	36	38	20	35
3	22	28	30	18	27

Confined Recovery

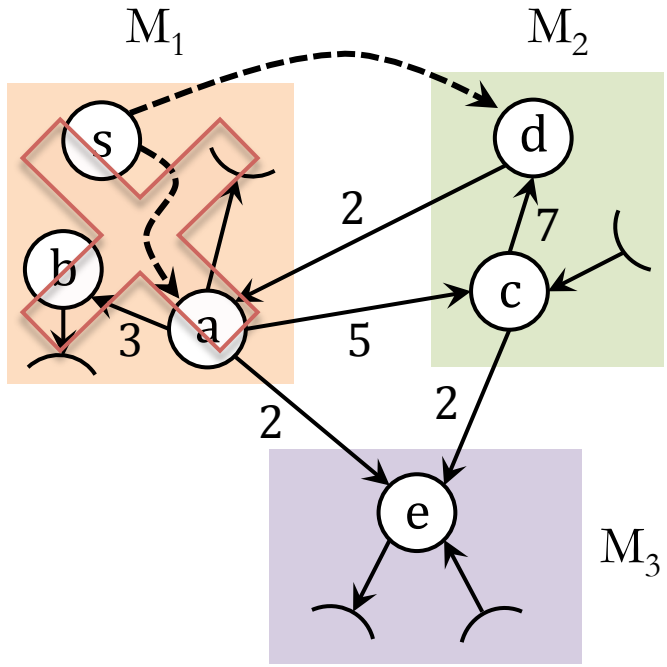
	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3			30	19	27
4			27	18	24
5			26	17	22
M_1 DIES					



Confined Recovery

- Can *a, b* join *c, d* & *e* ?

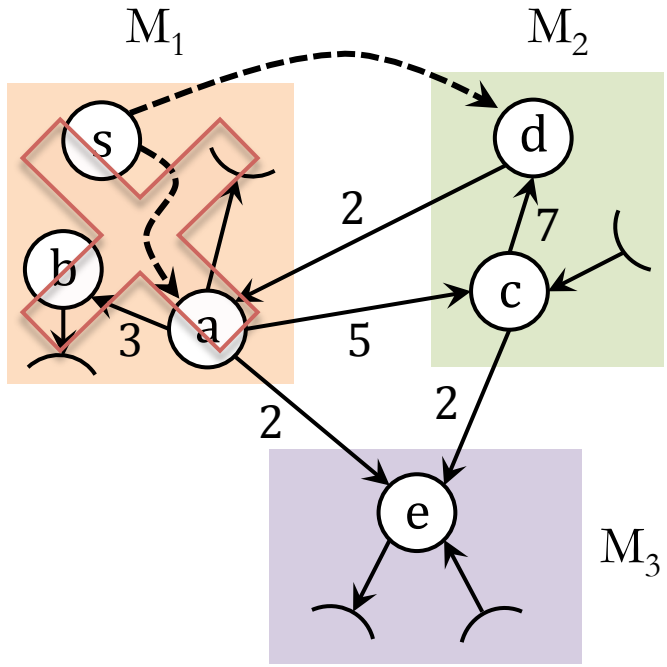
	a	***b***	***c***	***d***	***e***
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3			30	19	27
4			27	18	24
5			26	17	22
M_1 DIES					



Confined Recovery

- Can *a, b* join *c, d* & *e* ?

	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3			30	19	27
4			27	18	24
5			26	17	22
	M_1 DIES				
6	22	36	30	17	27

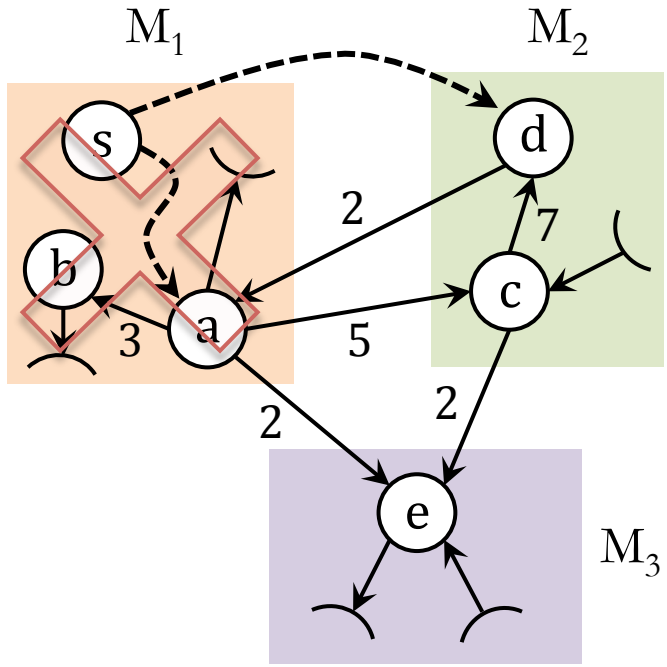


Confined Recovery

- Can *a, b* join *c, d* & *e* ?

	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3			30	19	27
4			27	18	24
5			26	17	22
	M_1 DIES				
6	22	36	30	17	27

Progressive Reads
c & *e* cannot use *a*

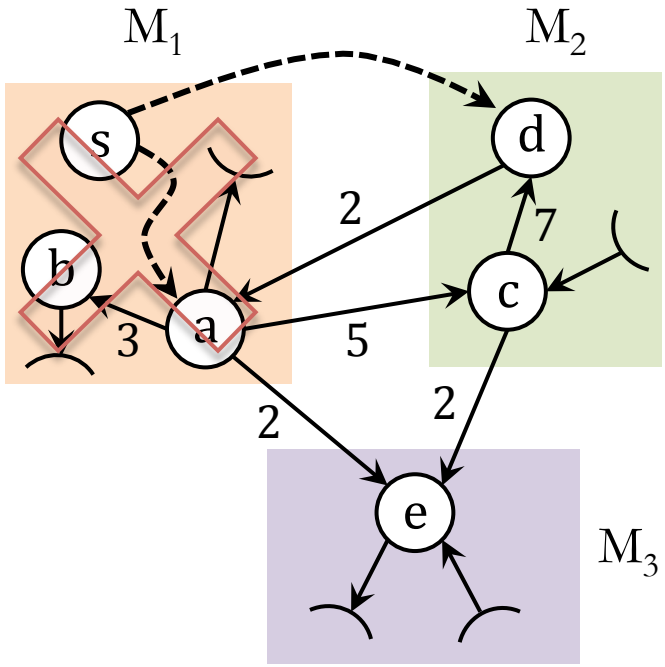


c,d,e can use *a,b* ?

Confined Recovery

- Can *a, b* **safely use** *c, d* & *e* ?

	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3			30	19	27
4			27	18	24
5			26	17	22
M_1 DIES					

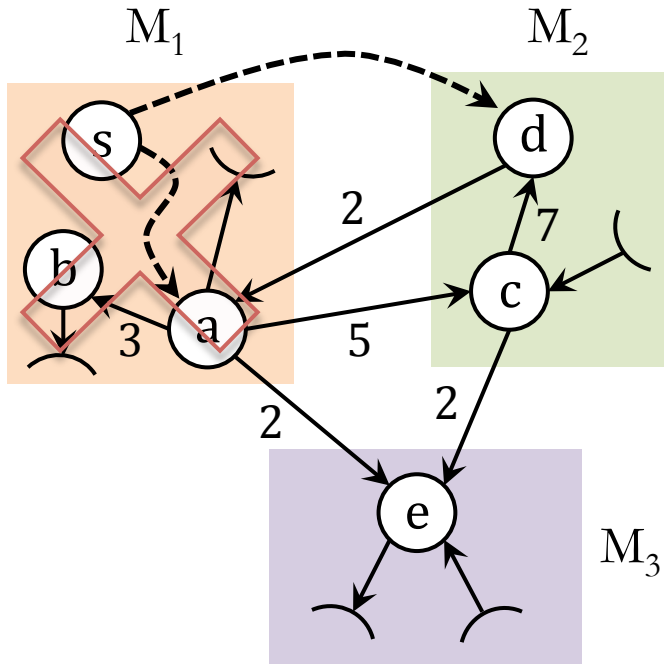


a,b can use *c,d,e* ?	
c,d,e can use *a,b* ?	✖

Confined Recovery

- *a, b* can safely use *c, d* & *e*

	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3			30	19	27
4			27	18	24
5			26	17	22
M_1 DIES					

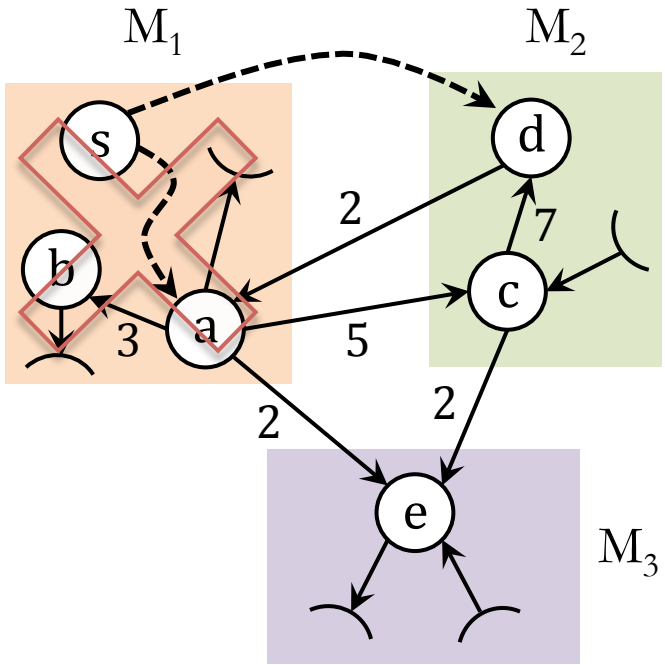


a,b can use *c,d,e* ?	✓
c,d,e can use *a,b* ?	✗

Confined Recovery

- *a, b* can safely use *c, d* & *e*

	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3	22	28	30	19	27
4	21	25	27	18	24
5	20	24	26	17	22
M_1 DIES					



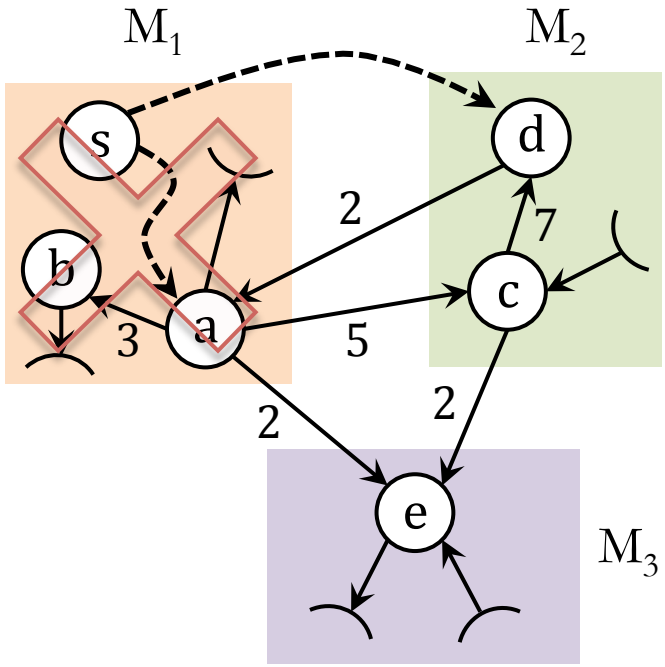
a,b can use *c,d,e* ?	✓
c,d,e can use *a,b* ?	✗

Confined Recovery

- *a, b* can safely use *c, d* & *e*

	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3			30	19	27
4			27	18	24
5			26	17	22
	M_1 DIES				

	a	*b*
6	19	36
7	19	22
CONVERGED		

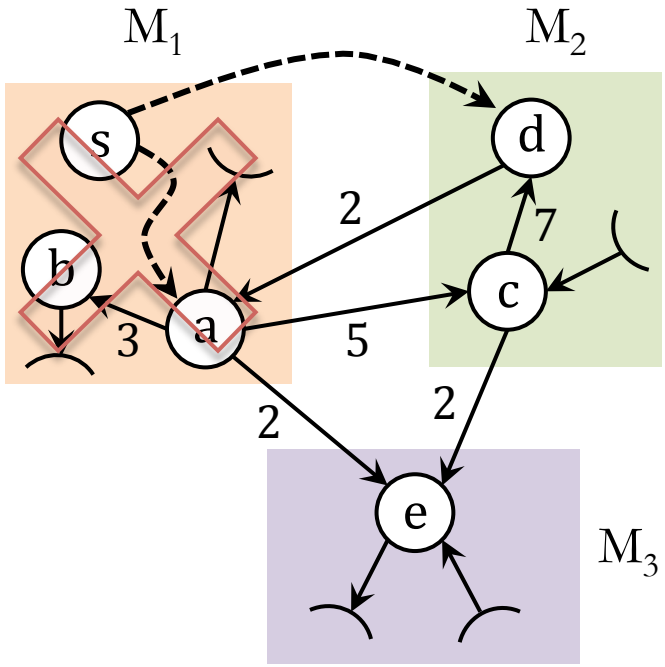


a,b can use *c,d,e* ?	✓
c,d,e can use *a,b* ?	✗

Confined Recovery

	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3			30	19	27
4			27	18	24
5			26	17	22
M_1 DIES					

	a	*b*
6	19	36
7	19	22
CONVERGED		

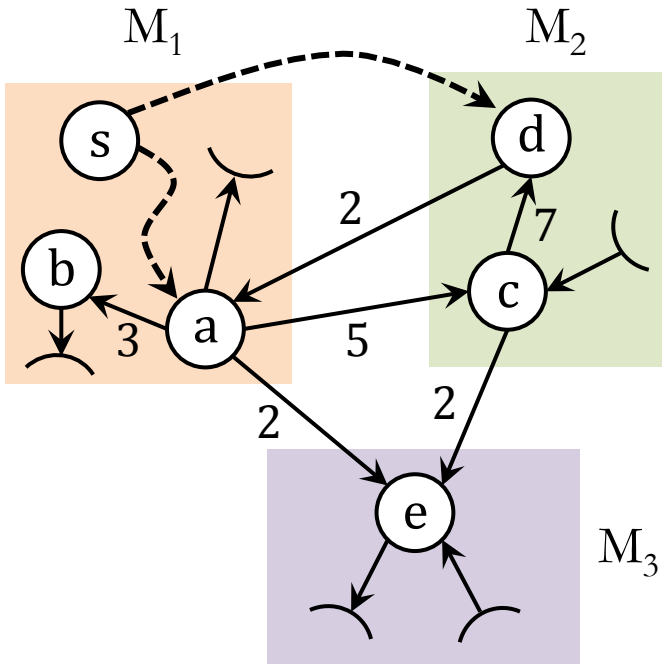


a,b can use *c,d,e* ?	✓
c,d,e can use *a,b* ?	✗

Confined Recovery

	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3			30	19	27
4			27	18	24
5			26	17	22
	M_1 DIES				

	a	*b*
6	19	36
7	19	22
	RECOVERED	

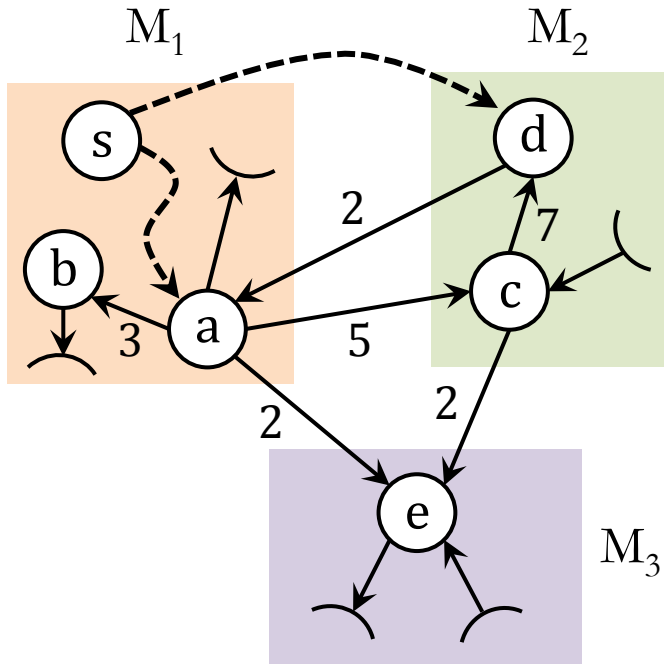


a,b can use *c,d,e* ?	✓
c,d,e can use *a,b* ?	✓

Confined Recovery

	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3			30	19	27
4			27	18	24
5			26	17	22
	M_1 DIES				
8	19	22	24	17	21

	a	*b*
6	19	36
7	19	22
	RECOVERED	



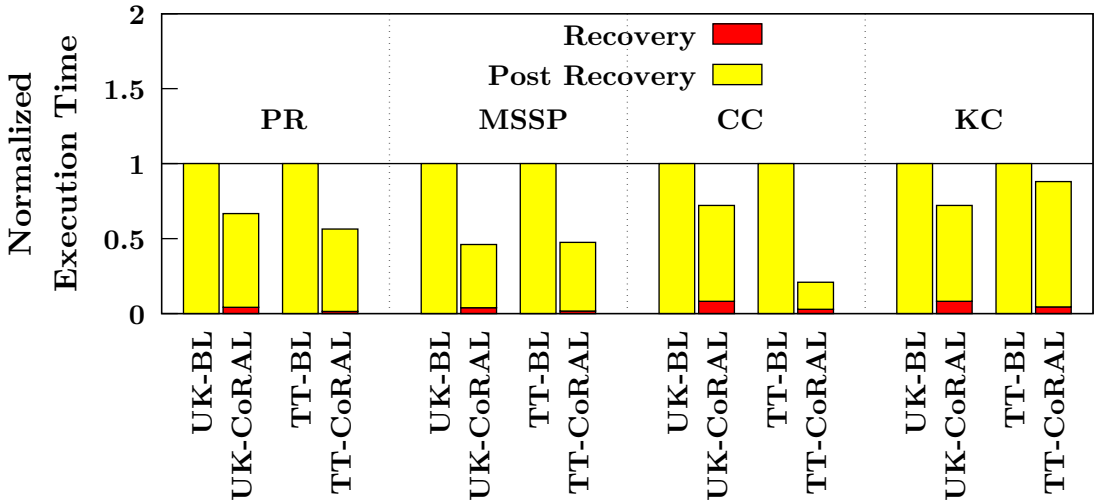
a,b can use *c,d,e* ?	✓
c,d,e can use *a,b* ?	✓

Experimental Setup

- 16-node EC2 cluster: 8-core/64GB nodes
- Asynchronous programs
  - PageRank (PR), MultipleSourceShortestPaths (MSSP), ConnectedComponents (CC), KCoreDecomposition (KC)

Graphs	**#Edges**	**#Vertices**
Twitter (TT)	1.5B	41.7M
UKDomain (UK)	1.0B	39.5M
LiveJournal (LJ)	69M	4.8M

Confined Recovery: Performance



Graphs	#Edges	#Vertices
Twitter (TT)	1.5B	41.7M
UKDomain (UK)	1.0B	39.5M

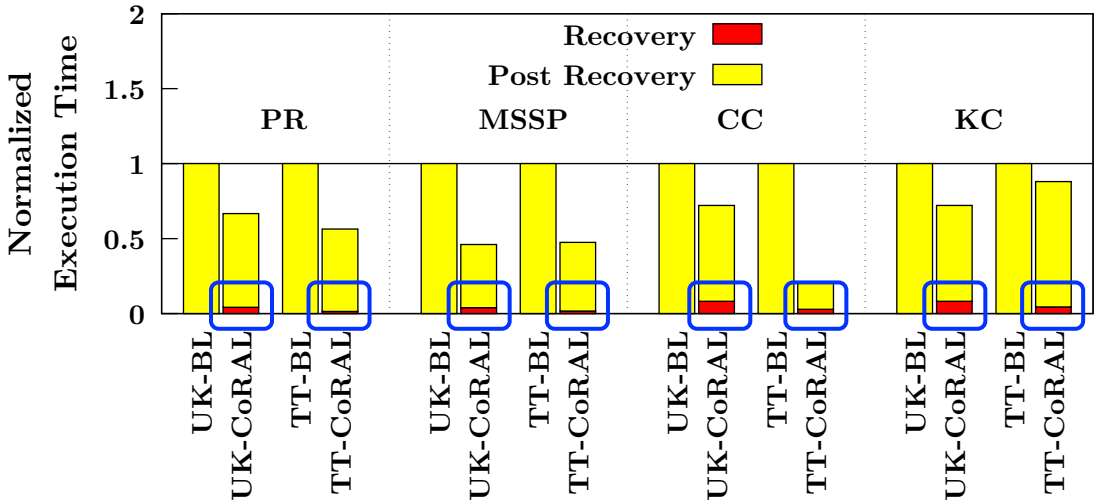
1.3-2.3x faster

Absolute times (BL)

	UK	TT
PR	5.5 min	10 min
MSSP	1.1 min	1 min
CC	2 min	2.8 min
KC	3.2 min	10.2 min

Confined Recovery: Performance

- Recovery 3-4% of total execution time



Graphs	#Edges	#Vertices
Twitter (TT)	1.5B	41.7M
UKDomain (UK)	1.0B	39.5M

1.3-2.3x faster

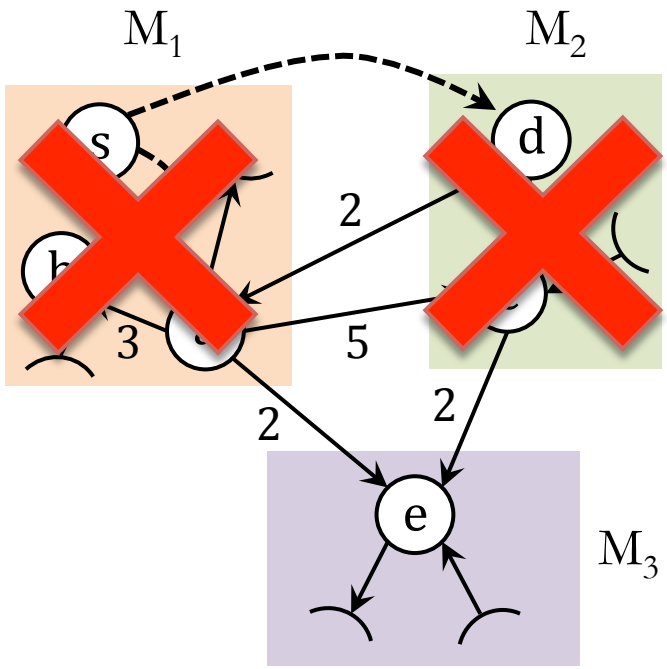
Absolute times (BL)

	UK	TT
PR	5.5 min	10 min
MSSP	1.1 min	1 min
CC	2 min	2.8 min
KC	3.2 min	10.2 min

Confined Recovery

	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3					27
4					24
5					22
M_1 & M_2 DIE					
10	19	22	24	17	21

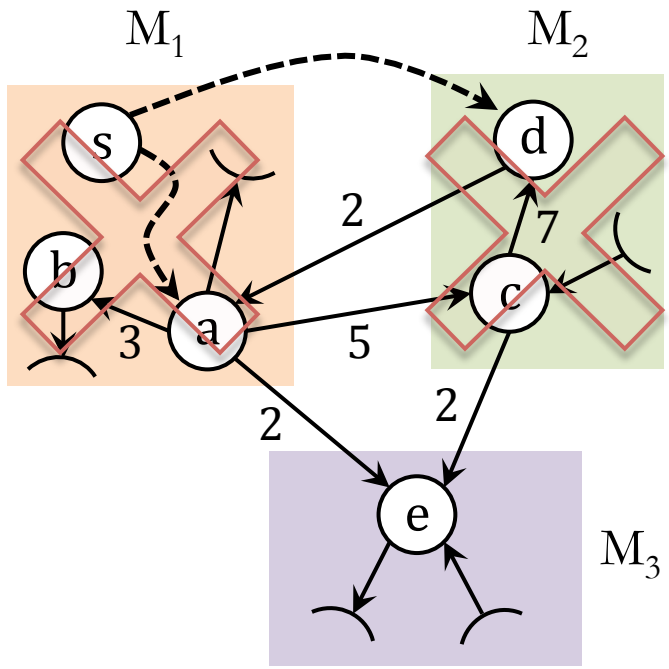
	a	*b*	*c*	*d*
6	22	28	30	18
7	20	25	27	17
8	19	23	25	17
9	19	22	24	17
CONVERGED				



Confined Recovery

- Inconsistent state

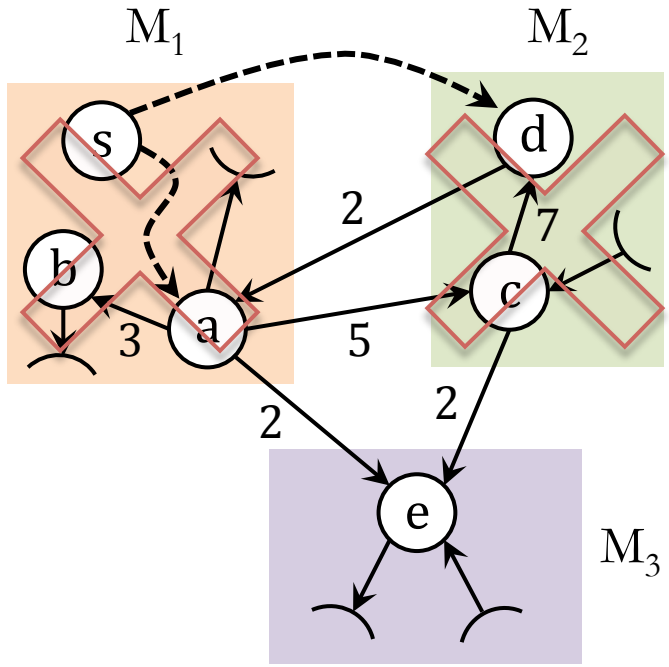
	a	***b***	***c***	***d***	***e***
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3	22	28	30	19	27
4	21	25	27	18	24
5	20	24	26	17	22
M_1 & M_2 DIE					



Confined Recovery

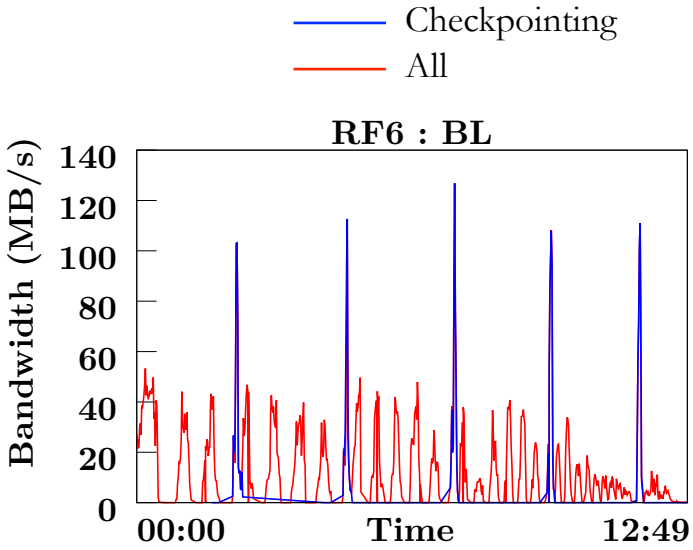
- Inconsistent state

	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3	22	28	30	19	27
4	21	25	27	18	24
5	20	24	26	17	22
M_1 & M_2 DIE					



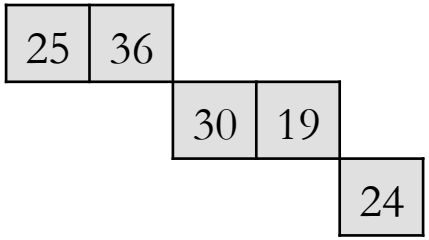
Checkpointing: Network

25	36	38	20	35

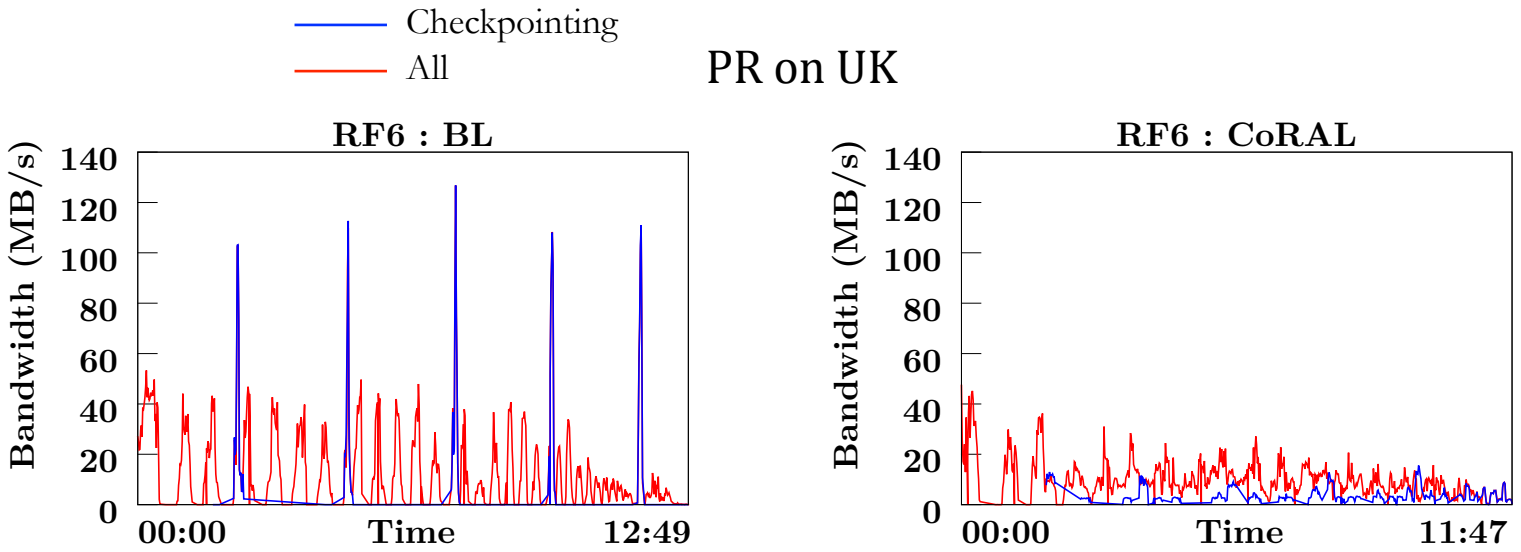


PR on UK

Checkpointing: Network



- Capturing inconsistent state
- 22-56% reduction in peak bandwidth (replication 1-6)



Confined Recovery

- Ordered recovery

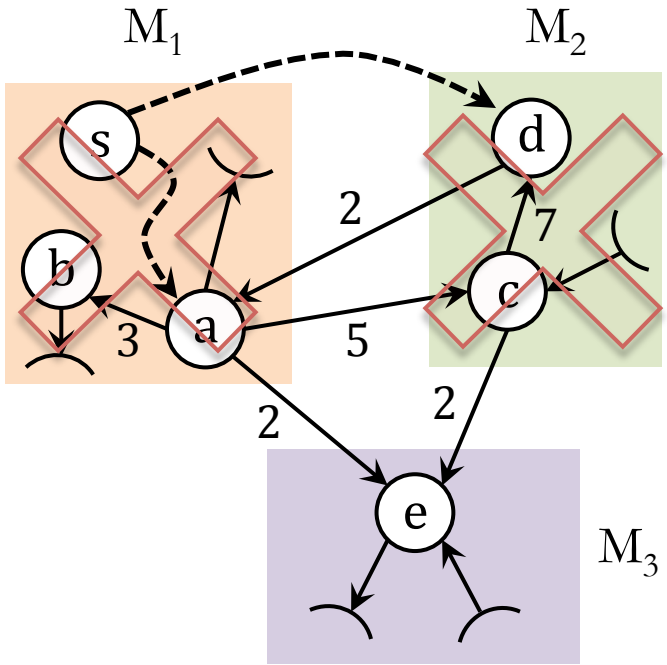
	a	*b*	*c*	*d*	*e*
0	∞	∞	∞	∞	∞
1	33	∞	∞	23	∞
2	25	36	38	20	35
3			30	19	27
4					24
5					22
M_1 & M_2 DIE					
11	19	22	24	17	21

1

	a	*b*
6	21	28
7	21	24

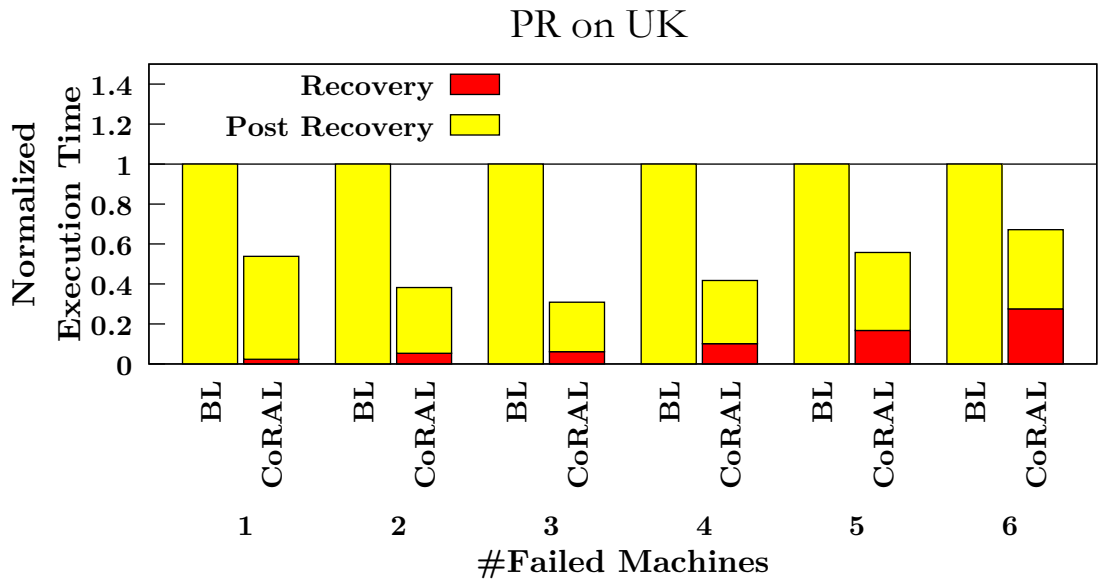
2

CONVERGED	*c*	*d*
8	21	24	26	17
9	19	24	26	17
10	19	22	24	17
RECOVERED				



Confined Recovery: Performance

- 16-node EC2 cluster: 8-core/64GB nodes



1.5-3.2x faster

Proofs

- Applicability of PR-Semantics
- Correctness of PR-Consistent recovery
- Necessity & sufficiency of PR-Ordering
- Correctness of capturing PR-Ordering

Summary

- Confined recovery
  - Construct alternate legal execution states
- Accelerates processing by up to 3.2x
- Local checkpointing
  - Reduces 99th percentile peak bandwidth by up to 51%

Thanks