

CMU SCS

Carnegie Mellon Univ.  
Dept. of Computer Science  
15-415 - Database Applications

Lecture #27: Distributed DB  
(R&G ch. 22)

---

---

---

---

---

---

---

CMU SCS

General Overview

- Relational model - SQL
- Functional Dependencies & Normalization
- Physical Design; Indexing
- Query optimization
- Transaction processing
- Advanced topics
  - Spatial DB
  - Data Mining
  - Distributed Databases

C. Faloutsos CMU SCS 15-415 2

---

---

---

---

---

---

---

CMU SCS

Problem – definition

- centralized DB:

C. Faloutsos CMU SCS 15-415 3

---

---

---

---

---

---

---

CMU SCS

### Problem – definition

- Distr. DB:
- DB stored in many places
- ... connected

LA NY

C. Faloutsos CMU SCS 15-415 4

---

---

---

---

---

---

---

---

CMU SCS

### Problem – definition

now: connect to LA; exec sql select \* from EMP; ...  
connect to NY; exec sql select \* from EMPLOYEE; ...

LA NY

EMP DBMS1 DBMS2 EMPLOYEE

C. Faloutsos CMU SCS 15-415 5

---

---

---

---

---

---

---

---

CMU SCS

### Problem – definition

ideally: connect to distr-LA; exec sql select \* from EMPL;

LA NY

EMP D-DBMS DBMS1 DBMS2 EMPLOYEE

C. Faloutsos CMU SCS 15-415 6

---

---

---


---

---

---

---

---



CMU SCS

## Pros + Cons

- Pros
  - .
  - .
  - .
- Cons
  - .
  - .
  - .

C. Faloutsos CMU SCS 15-415 7

---

---

---


---

---

---

---

---



CMU SCS

## Pros + Cons

- Pros
  - Data sharing
  - reliability & availability
  - speed up of query processing
- Cons
  - software development cost
  - more bugs
  - may increase processing overhead (msg)

C. Faloutsos CMU SCS 15-415 8

---

---

---


---

---

---

---

---



CMU SCS

## Overview

- Problem – motivation
- Design issues
- Query optimization – **semijoins**
- transactions (recovery, conc. control)

C. Faloutsos CMU SCS 15-415 9

---

---

---

---

---

---

---

---

CMU SCS

## Design of Distr. DBMS

what are our choices of storing a table?

C. Faloutsos CMU SCS 15-415 10

---

---

---

---

---

---

---

CMU SCS

## Design of Distr. DBMS

- replication
- fragmentation (horizontal; vertical; hybrid)
- both

C. Faloutsos CMU SCS 15-415 11

---

---

---

---

---

---

---

CMU SCS

## Design of Distr. DBMS

ssn	name	address
123	smith	wall str.
...	...	...
234	johnson	sunset blvd

vertical fragm.

horiz. fragm.

C. Faloutsos CMU SCS 15-415 12

---

---

---

---

---

---

---

CMU SCS

## Transparency & autonomy

Issues/goals:

- naming and local autonomy
- replication and fragmentation transp.
- location transparency

i.e.:

C. Faloutsos CMU SCS 15-415 13

---

---

---

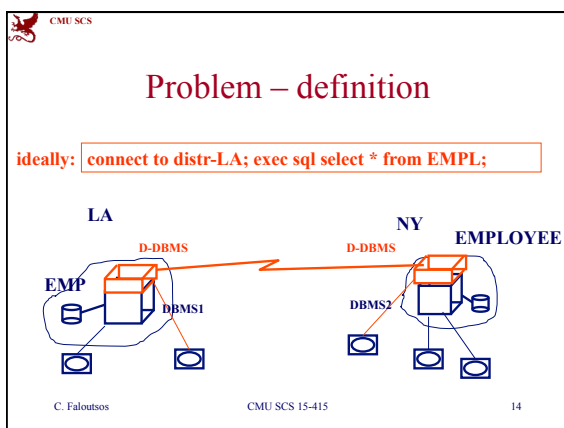
---

---

---

---

---




---

---

---

---

---

---

---

---

CMU SCS

## Overview

- Problem – motivation
- Design issues
- Query optimization – **semijoins**
- transactions (recovery, conc. control)

C. Faloutsos CMU SCS 15-415 15

---

---

---

---

---

---

---

---

CMU SCS

## Distributed Query processing

- issues (additional to centralized q-opt)
  - cost of transmission  
(cpu, disk, #bytes-transmitted, #messages-transmitted)
  - parallelism / overlap of delays
- minimize elapsed time?
- or minimize resource consumption?

C. Faloutsos CMU SCS 15-415 16

---

---

---

---

---

---

---

---

CMU SCS

## Distr. Q-opt – semijoins

S1

s#	...
s1	
s2	
s5	
s11	

S2

s#	p#
s1	p1
s2	p1
s3	p5
s2	p9

S3

SUPPLIER Join SHIPMENT = ?

C. Faloutsos CMU SCS 15-415 17

---

---

---

---

---

---

---

---

CMU SCS

## semijoins

- choice of plans?

C. Faloutsos CMU SCS 15-415 18

---

---

---

---

---

---

---

---

CMU SCS

## semijoins

- choice of plans?
- plan #1: ship SHIP -> S2; join; ship -> S3
- plan #2: ship SHIP->S3; ship SUP->S3; join
- ...
- others?

C. Faloutsos CMU SCS 15-415 19

---

---

---

---

---

---

---

---

CMU SCS

## Distr. Q-opt – semijoins

S1

s#	...
s1	
s2	
s5	
s11	

S2

s#	p#
s1	p1
s2	p1
s3	p5
s2	p9

S3

SUPPLIER Join SHIPMENT = ?

C. Faloutsos CMU SCS 15-415 20

---

---

---

---

---

---

---

---

CMU SCS

## Semijoins

- Idea: reduce the tables before shipping

S1

s#	...
s1	
s2	
s5	
s11	

S2

s#	p#
s1	p1
s2	p1
s3	p5
s2	p9

S3

SUPPLIER Join SHIPMENT = ?

C. Faloutsos CMU SCS 15-415 21

---

---

---

---

---

---

---

---

CMU SCS

## Semijoins

- How to do the reduction, cheaply?
- Eg., reduce 'SHIPMENT':

C. Faloutsos CMU SCS 15-415 22

---

---

---

---

---

---

---

---

CMU SCS

## Semijoins

- Idea: reduce the tables before shipping

S1

s#	...
s1	
s2	
s5	
s11	

(s1,s2,s5,s11)

S2

s#	p#
s1	p1
s2	p1
s3	p5
s2	p9

S3

SUPPLIER Join SHIPMENT = ?

C. Faloutsos CMU SCS 15-415 23

---

---

---

---

---

---

---

---

CMU SCS

## Semijoins

- Formally:
- SHIPMENT' = SHIPMENT  $\bowtie$  SUPPLIER
- express semijoin w/ rel. algebra

C. Faloutsos CMU SCS 15-415 24

---

---

---

---

---

---

---

---

CMU SCS

## Semijoins

- Formally:
- $\text{SHIPMENT}' = \text{SHIPMENT} \bowtie \text{SUPPLIER}$
- express semijoin w/ rel. algebra

$$R' = R \bowtie S$$

$$= \pi_R(R \bowtie S)$$

C. Faloutsos CMU SCS 15-415 25

---

---

---

---

---

---

---

---

CMU SCS

## Semijoins – eg:

- suppose each attr. is 4 bytes
- Q: transmission cost (#bytes) for semijoin  $\text{SHIPMENT}' = \text{SHIPMENT} \bowtie \text{SUPPLIER}$

C. Faloutsos CMU SCS 15-415 26

---

---

---

---

---

---

---

---

CMU SCS

## Semijoins

- Idea: reduce the tables before shipping

**SUPPLIER**

s#	...
s1	
s2	
s5	
s11	

**SHIPMENT**

s#	p#
s1	p1
s2	p1
s3	p5
s2	p9

4 bytes

$\text{SUPPLIER} \Join \text{SHIPMENT} = ?$

C. Faloutsos CMU SCS 15-415 27

---

---

---


---

---

---

---

---



**Semijoins – eg:**

- suppose each attr. is 4 bytes
- Q: transmission cost (#bytes) for semijoin  
SHIPMENT' = SHIPMENT semijoin SUPPLIER
- A: 4\*4 bytes

C. Faloutsos CMU SCS 15-415 28

---

---


---

---

---

---

---



**Semijoins – eg:**

- suppose each attr. is 4 bytes
- Q1: give a plan, with semijoin(s)
- Q2: estimate its cost (#bytes shipped)

C. Faloutsos CMU SCS 15-415 29

---

---


---

---

---

---

---



**Semijoins – eg:**

- A1:
  - reduce SHIPMENT to SHIPMENT'
  - SHIPMENT' -> S3
  - SUPPLIER -> S3
  - do join @ S3
- Q2: cost?

C. Faloutsos CMU SCS 15-415 30

---

---

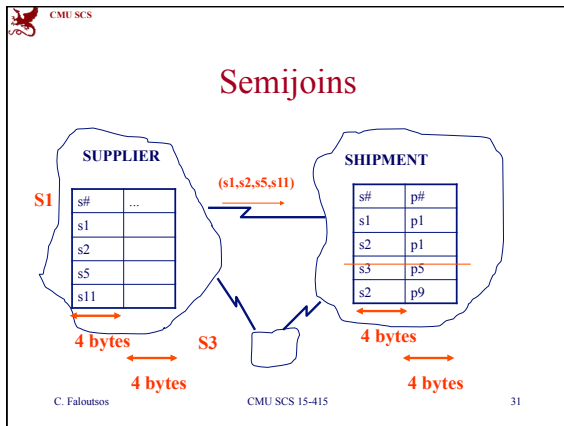
---

---

---

---

---




---

---

---

---

---

---

---

---

**Semijoins – eg:**

- A2:
  - 4\*4 bytes - reduce SHIPMENT to SHIPMENT'
  - 3\*8 bytes - SHIPMENT' -> S3
  - 4\*8 bytes - SUPPLIER -> S3
  - 0 bytes - do join @ S3

**72 bytes TOTAL**

C. Faloutsos      CMU SCS 15-415      32

---

---

---

---

---

---

---

---

**Other plans?**

C. Faloutsos      CMU SCS 15-415      33

---

---

---


---

---

---

---

---



CMU SCS

## Other plans?

P2:

- reduce SHIPMENT to SHIPMENT'
- reduce SUPPLIER to SUPPLIER'
- SHIPMENT' -> S3
- SUPPLIER' -> S3

C. Faloutsos CMU SCS 15-415 34

---

---


---

---

---

---

---



CMU SCS

## Other plans?

P3:

- reduce SUPPLIER to SUPPLIER'
- SUPPLIER' -> S2
- do join @ S2
- ship results -> S3

C. Faloutsos CMU SCS 15-415 35

---

---


---

---

---

---

---



CMU SCS

## A brilliant idea: 'Bloom-joins'

- (not in the book – not in the final exam)
- how to ship the projection, say, of SUPPLIER.s#, even cheaper?
- A: Bloom-filter [Lohman+] =
  - quick&dirty membership testing

C. Faloutsos CMU SCS 15-415 36

---

---

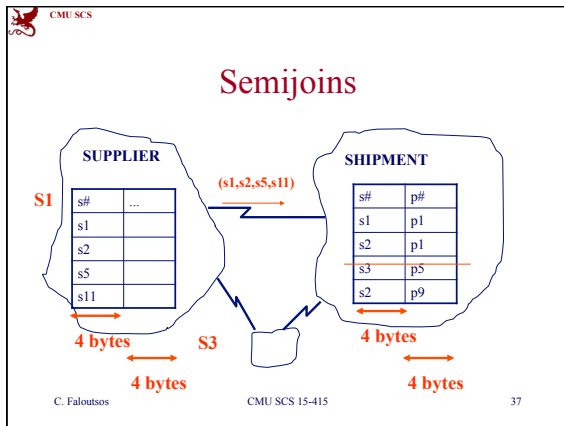
---

---

---

---

---




---

---

---

---

---

---

---

---

**Another brilliant idea: two-way semijoins**

- (not in book, not in final exam)
- reduce both relations with one more exchange: [Kang, '86]
- ship back the list of keys that didn't match
- CAN NOT LOSE! (why?)
- further improvement:
  - or the list of ones that matched – whatever is shorter!

C. Faloutsos CMU SCS 15-415 38

---

---

---

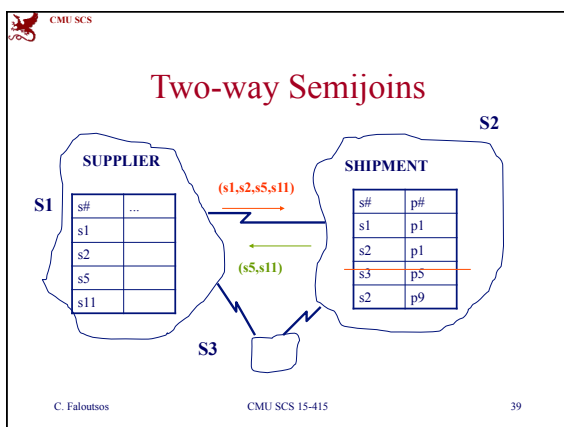
---

---

---

---

---




---

---

---


---

---

---

---

---

 CMU SCS
 

## Overview

- Problem – motivation
- Design issues
- Query optimization – **semijoins**
- transactions (recovery, conc. control)

C. Faloutsos CMU SCS 15-415 40

---

---


---

---

---

---

---

 CMU SCS
 

## Transactions – recovery

- Problem: eg., a transaction moves  
\$100 from NY -> \$50 to LA, \$50 to Chicago
- 3 sub-transactions, on 3 systems, with 3 W.A.L.s
- how to guarantee atomicity (all-or-none)?
- Observation: additional types of failures  
(links, servers, delays, time-outs ....)

C. Faloutsos CMU SCS 15-415 41

---

---


---

---

---

---

---

 CMU SCS
 

## Transactions – recovery

- Problem: eg., a transaction moves  
\$100 from NY -> \$50 to LA, \$50 to Chicago

C. Faloutsos CMU SCS 15-415 42

---

---

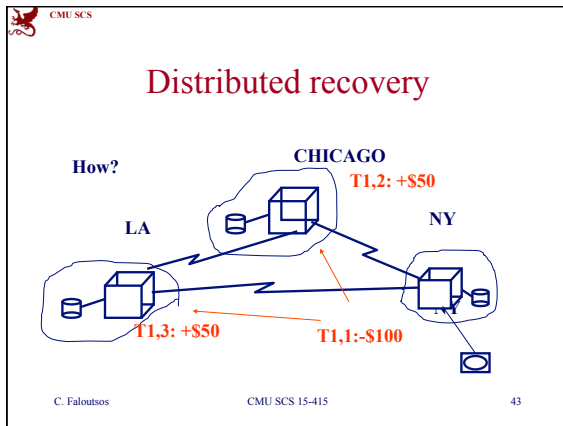
---

---

---

---

---




---

---

---

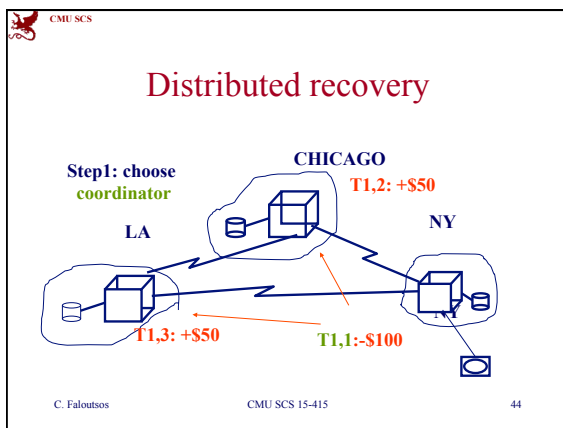
---

---

---

---

---




---

---

---

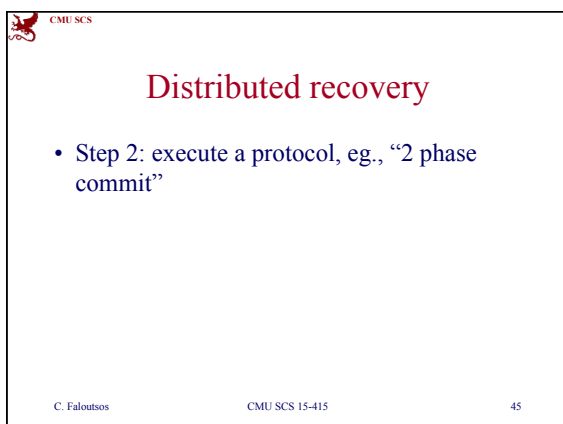
---

---

---

---

---




---

---

---

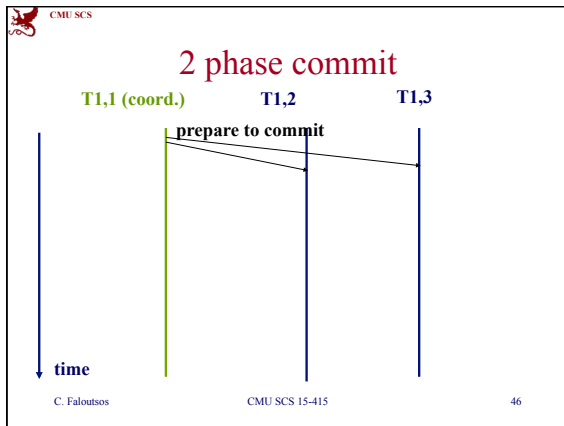
---

---

---

---

---




---

---

---

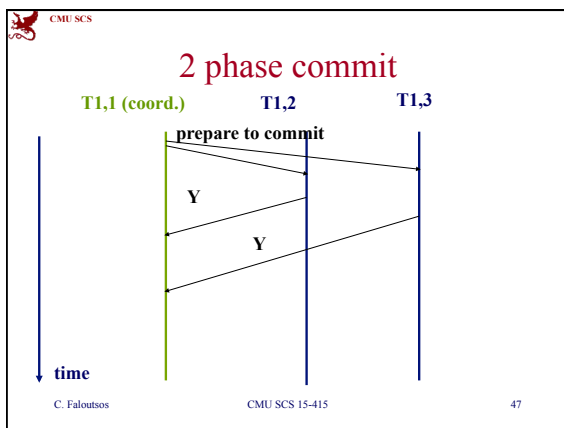
---

---

---

---

---




---

---

---

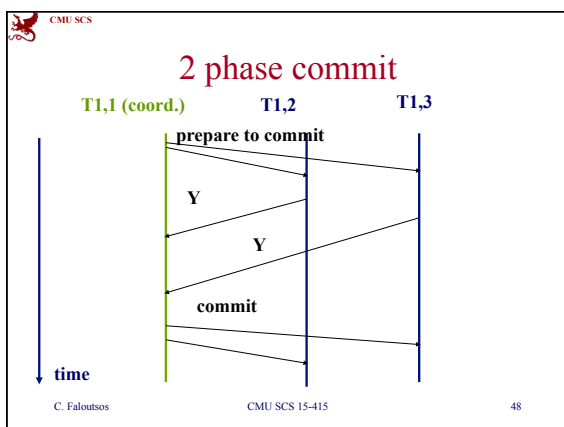
---

---

---

---

---




---

---

---

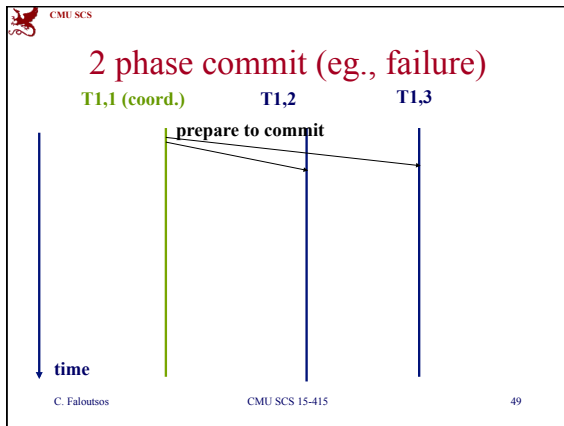
---

---

---

---

---




---

---

---

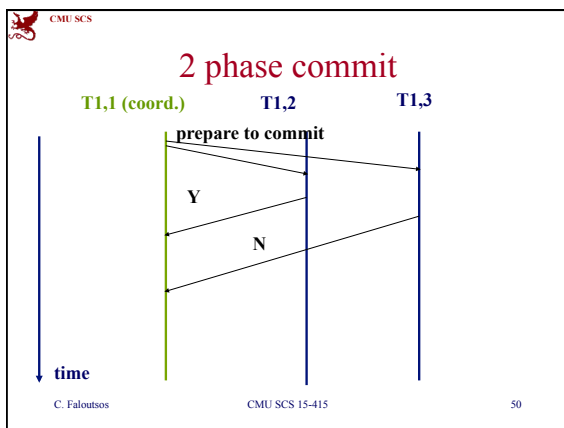
---

---

---

---

---




---

---

---

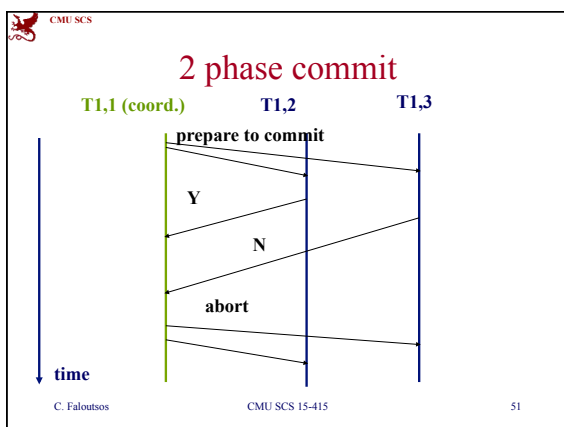
---

---

---

---

---




---

---

---


---

---

---

---

---



CMU SCS

## Distributed recovery

- Many, many additional details (what if the coordinator fails? what if a link fails? etc)
- and many other solutions (eg., 3-phase commit)

C. Faloutsos CMU SCS 15-415 52

---

---


---

---

---

---

---



CMU SCS

## Overview

- Problem – motivation
- Design issues
- Query optimization – semijoins
- transactions (recovery, conc. control)

C. Faloutsos CMU SCS 15-415 53

---

---


---

---

---

---

---



CMU SCS

## Distributed conc. control

- also more complicated:
- distributed deadlocks!

C. Faloutsos CMU SCS 15-415 54

---

---

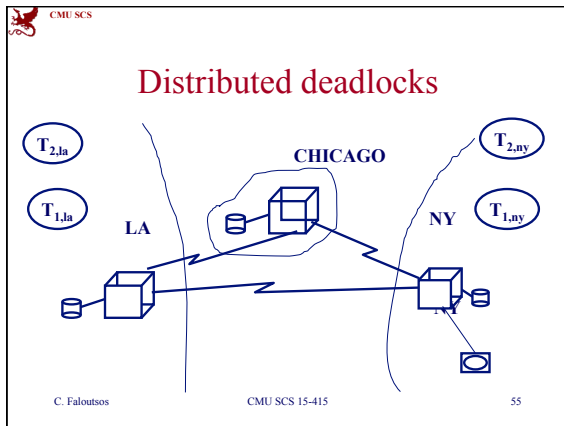
---

---

---

---

---




---

---

---

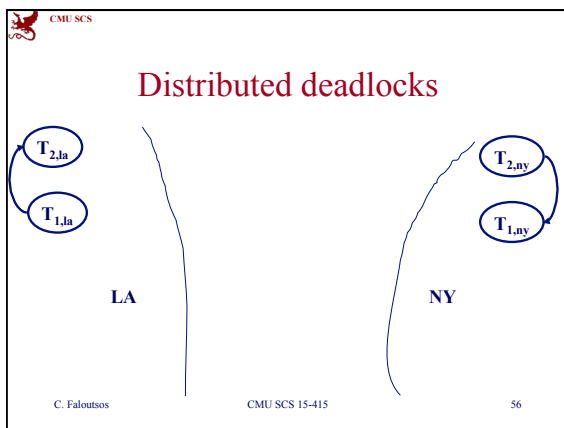
---

---

---

---

---




---

---

---

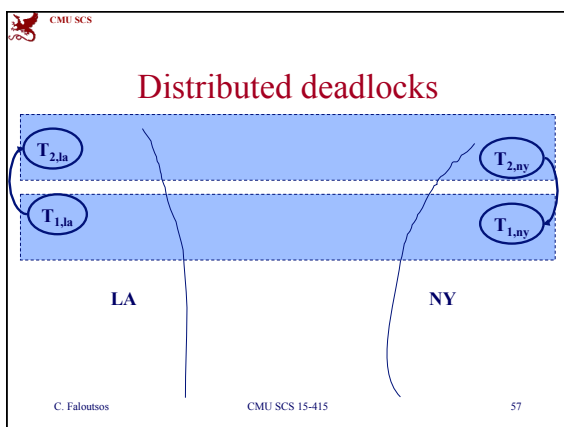
---

---

---

---

---




---

---

---

---

---

---

---

---

CMU SCS

## Distributed deadlocks

LA NY

$T_{2,la}$   $T_{2,ny}$

$T_{1,la}$   $T_{1,ny}$

- cites need to exchange wait-for graphs
- clever algorithms, to reduce # messages

C. Faloutsos CMU SCS 15-415 58

---

---

---

---

---

---

---

---

CMU SCS

## Conclusions

- Distr. DBMSs: not deployed
- BUT: produced clever ideas:
  - semijoins
  - distributed recovery / conc. control
- which can be useful for
  - parallel db / clusters
  - 'active disks'
  - replicated db (e-commerce servers)

C. Faloutsos CMU SCS 15-415 59

---

---

---

---

---

---

---

---