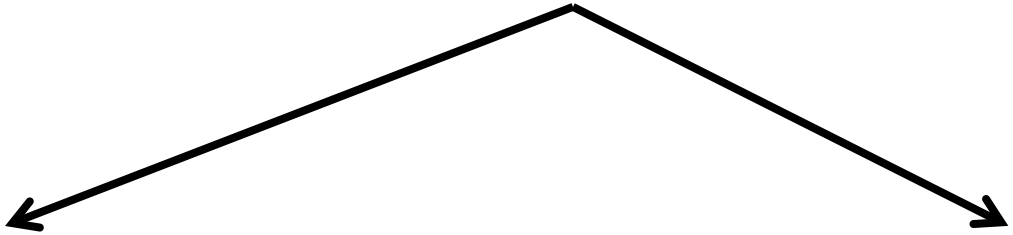
Research Focus

STM Systems



Dependability of STM

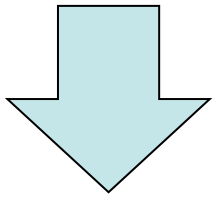
Roberto Palmieri **(PhD Student)**
"Sapienza" University of Rome
Italy

Performance Modelling of STM

Pierangelo Di Sanzo **(PhD Student)**
"Sapienza" University of Rome
Italy

Dependability Issues of STMs

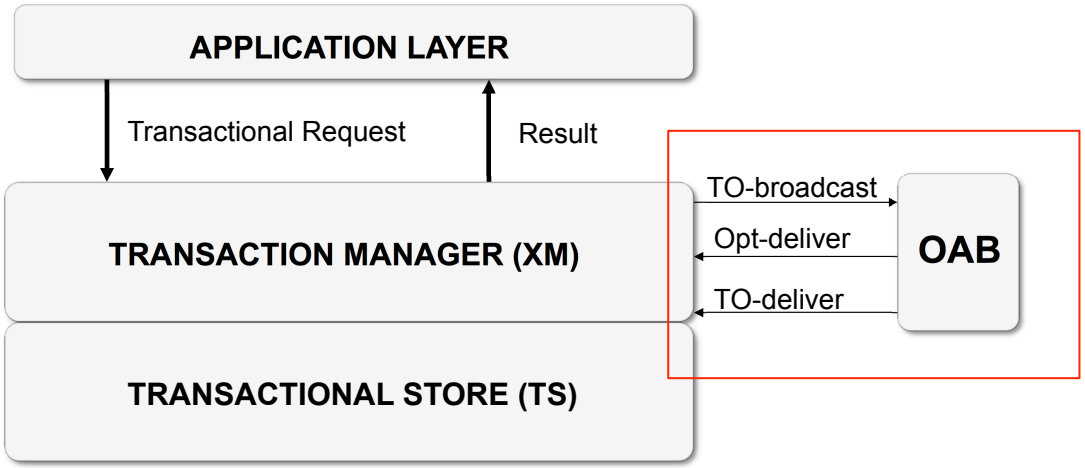
- In STM systems, manipulated data and related statements are not natively logged on stable storage
  - Data-audit loss in case of crashes
- Periodic logging (check-pointing) could be employed, however with time-granularity not adequate to the proper operation grain



REPLICATION

Active Replication + OAB

- Active Replication (AR) is a common replication scheme
- In AR each replica keeps the entire shared data-set and executes the same transactions in the same order
- Optimistic Atomic Broadcast protocol (OAB) is a group communication system involved

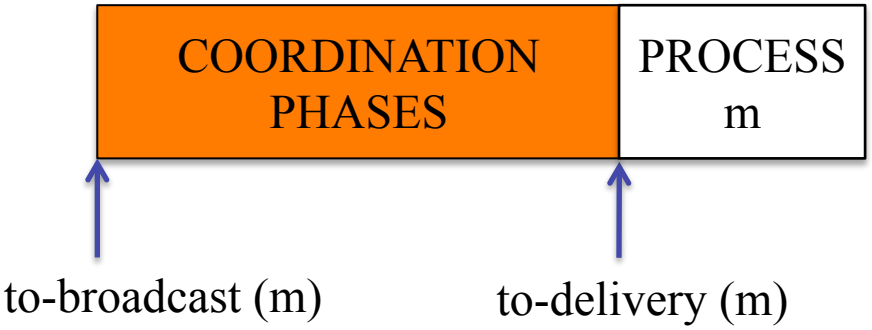


(example of software architecture replica)

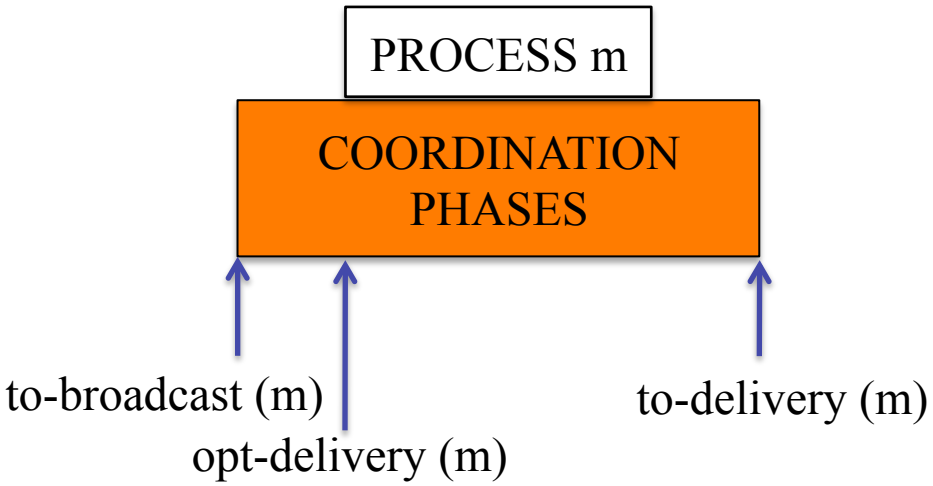
Overlapping Processing

- High delays (1/2*msec*) is in conflict with the growth of available resources in each nodes and with small transaction execution time (typical of STMs)
- Solution could be optimistically overlap local processing with replica coordination's:

WITHOUT OVERLAP (AB)



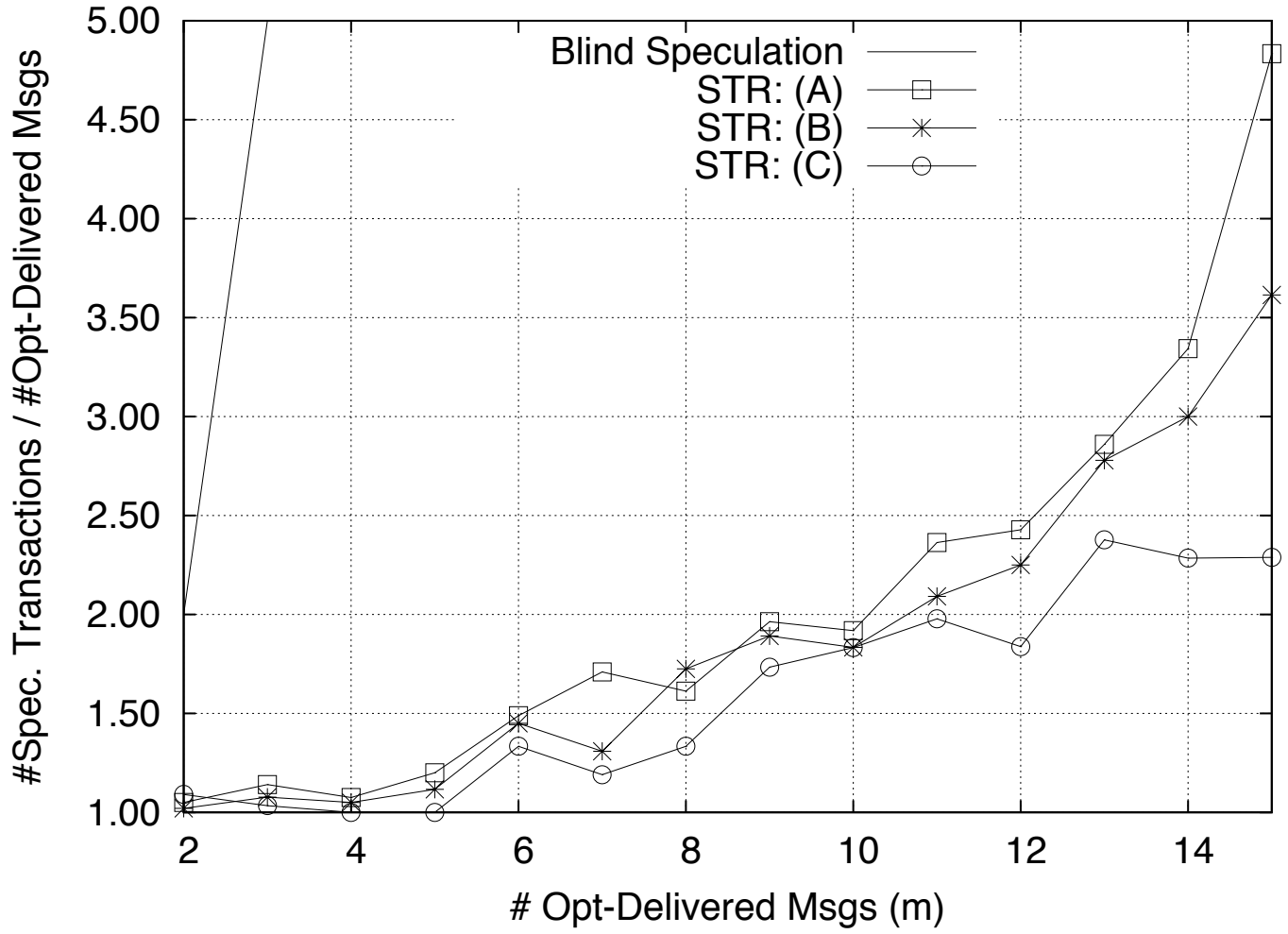
WITH OVERLAP (OAB)



STR FRAMEWORK

- **S**peculative **T**ransactional **R**eplication Framework
- Until the arrive of TO-Delivery, set of optimistically delivered (unordered) transactions could be processed in speculative way
- **Key Idea** -> STR Framework on-line identifies all and only transaction serialization orders that would cause optimistically executed transactions to exhibit distinct outcomes.
- Properties: Consistency, Non-redundancy, **Completeness**
- Graph based Concurrency Control:
  - Speculative Polygraph

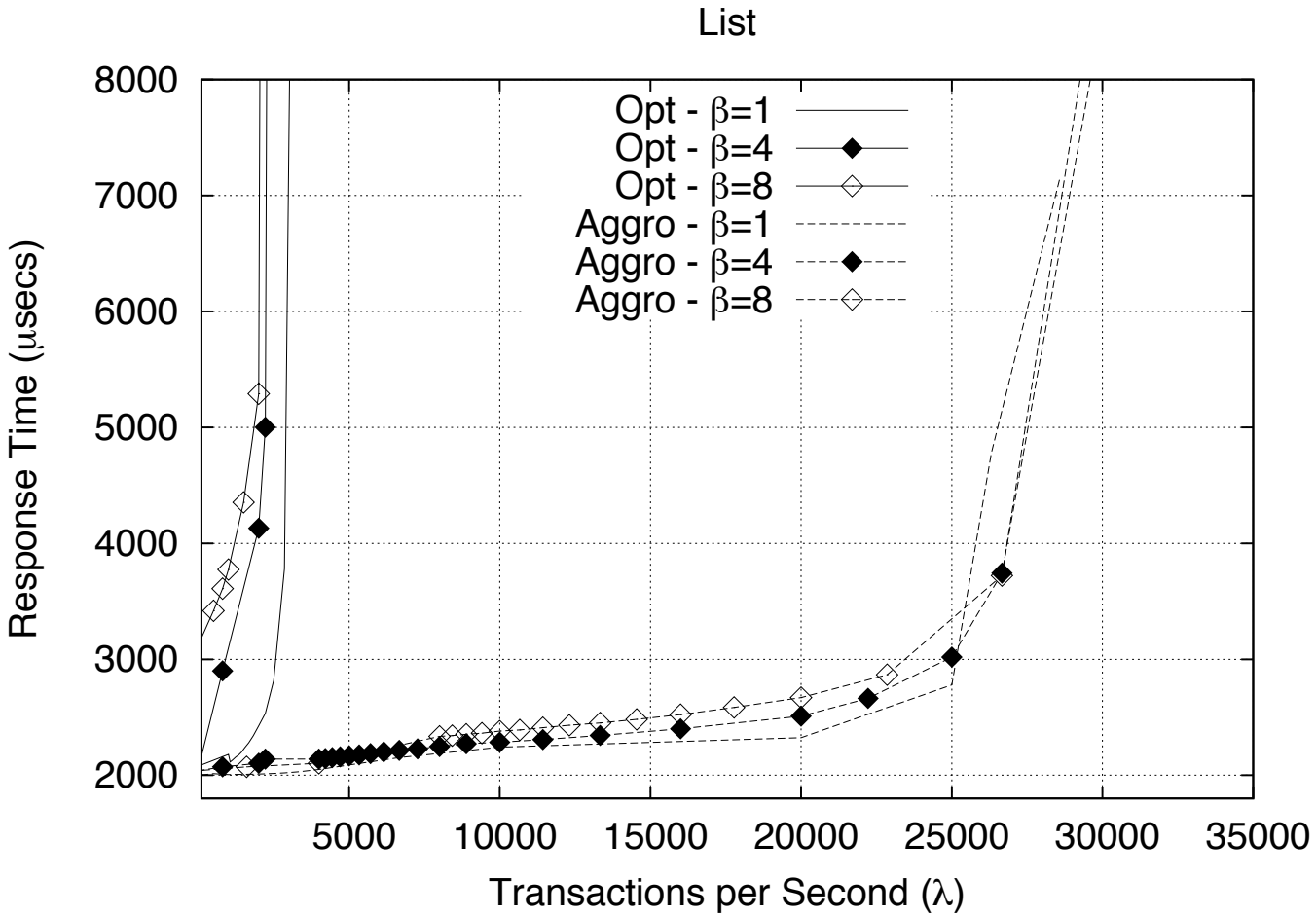
STR FRAMEWORK



AGGRO

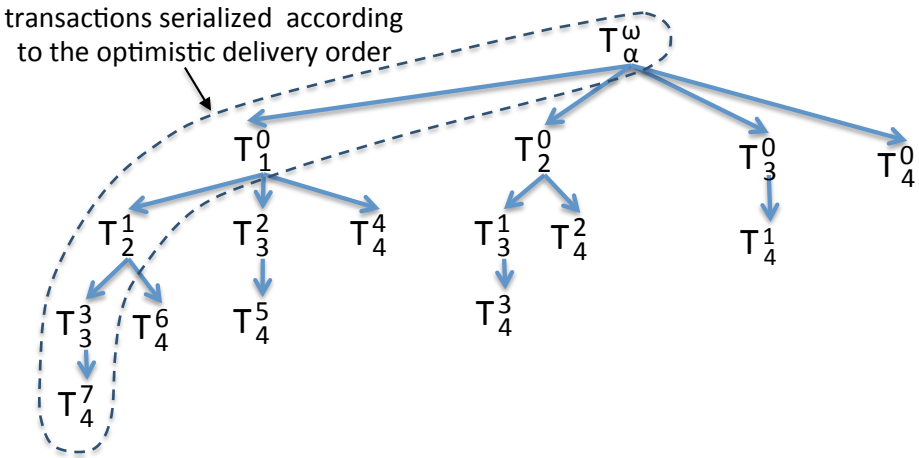
- **AGGR**essively **O**ptimistic replication protocol
- Tailored for network with spontaneous order (Opt-Delivery order matches TO-Delivery order)
- Optimistic processing aimed to follow the optimistic delivery order
- **The key idea** -> Uncommitted data item versions are aggressively made visible to other transactions independently of whether the creating transactions will be eventually committed
- Transactions abort/retry materializing a history compliant with optimistic delivery order

AGGRO


List
Response Time (μsecs)
Transactions per Second (λ)
8000
7000
6000
5000
4000
3000
2000
5000
10000
15000
20000
25000
30000
35000
Opt - β=1
Opt - β=4
Opt - β=8
Aggro - β=1
Aggro - β=4
Aggro - β=8

OSARE

- **O**pportunistic **S**peculation in **A**ctively **RE**plicated Transactional Systems
- The snapshot miss event is used to opportunistic exploring additional serialization orders
- The activation of new transaction instance involves any transaction originally serialized after that (like a wave on a different Speculative serialization order)



Transactional systems performance models: why?

Applications:

- system performance analysis
  - scalability analysis
  - identifying performance bottlenecks
  - ….
- performance comparison among different concurrency control algorithms (CCAs)

what-if analysis

- what would happen if I add one more thread
- what would happen if I change CCA
- ….

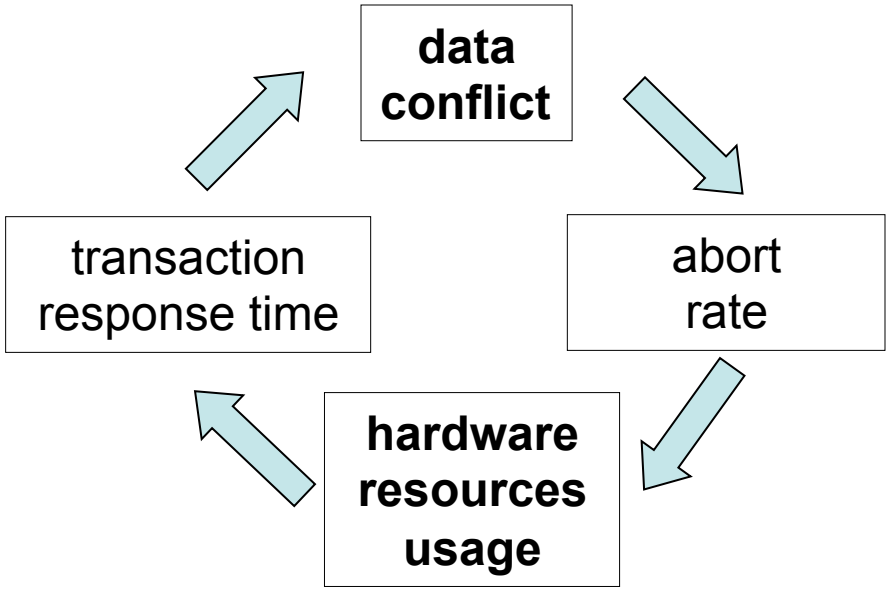
- evaluation of new CCAs
- ….

Building transactional systems performance model…

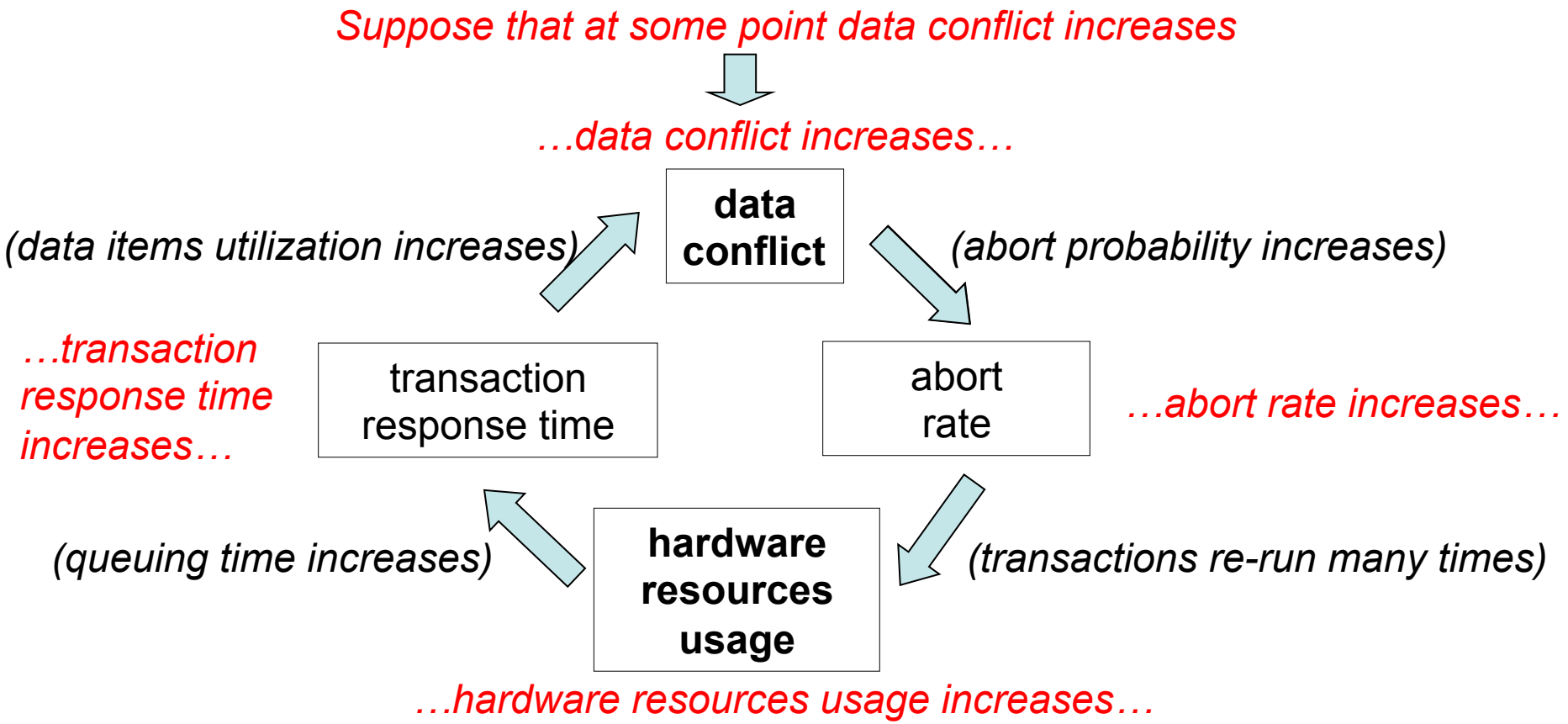
In transactional systems performance is affected by two factors:

- queuing and processing delay in accessing hardware resources (CPU, shared bus, …)
- data conflict in accessing shared data items

mutual dependence



Example (with optimistic concurrency control):



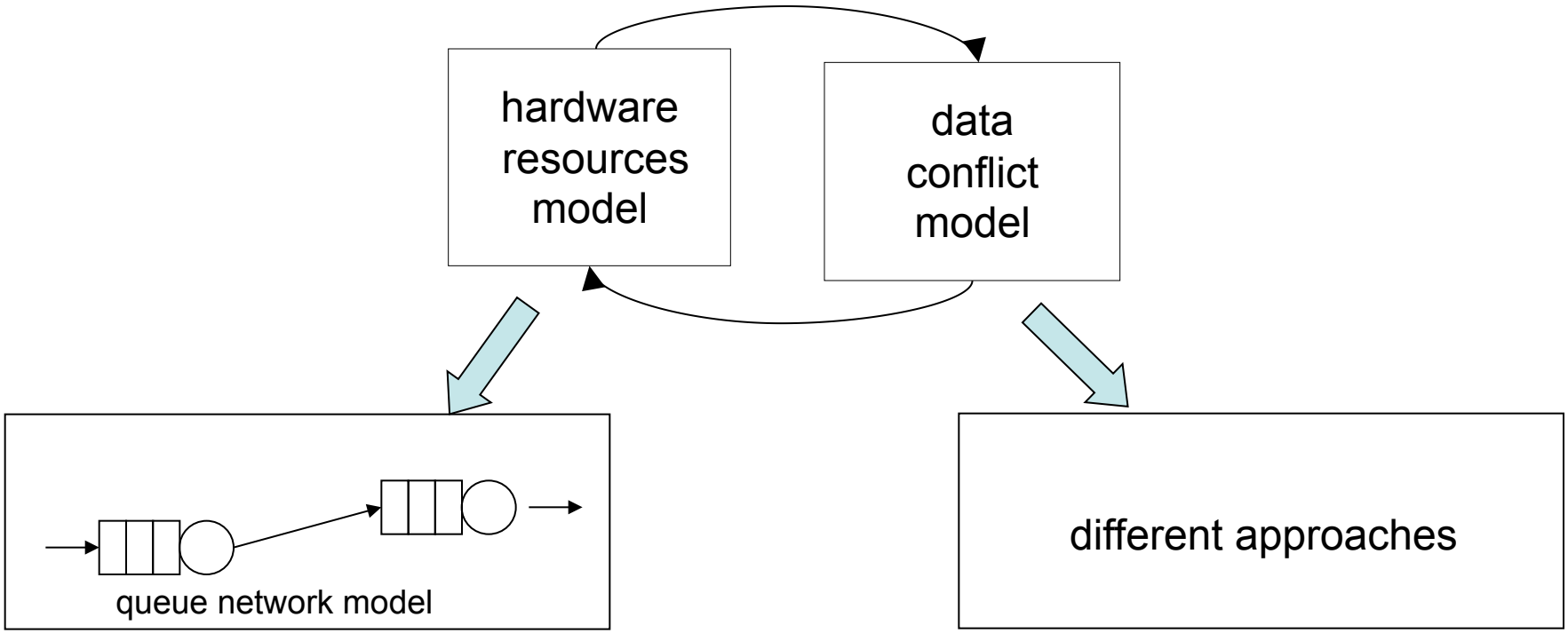
Performance modelling approach

two separated modelling layers:

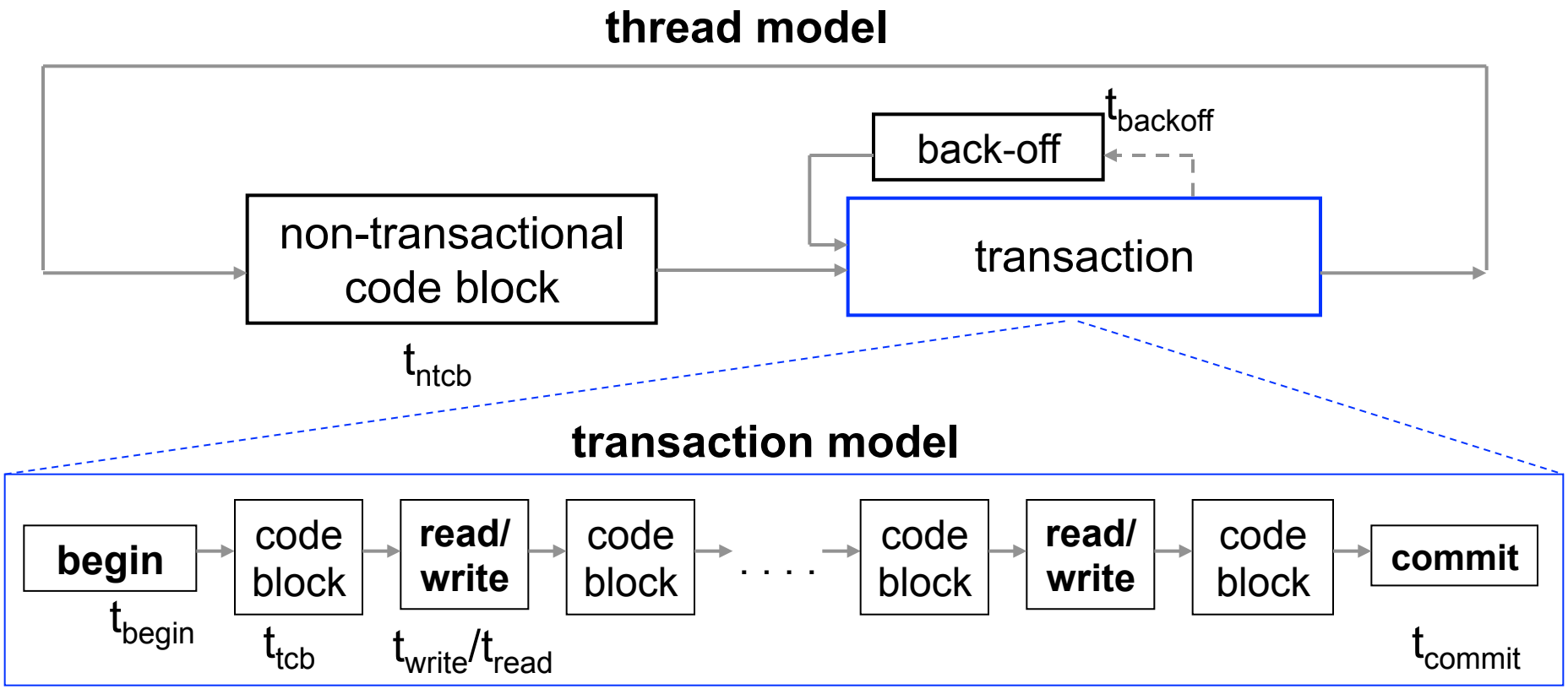
1) hardware resources model

2) data conflict model

Iterative approach to estimate system performance indicators



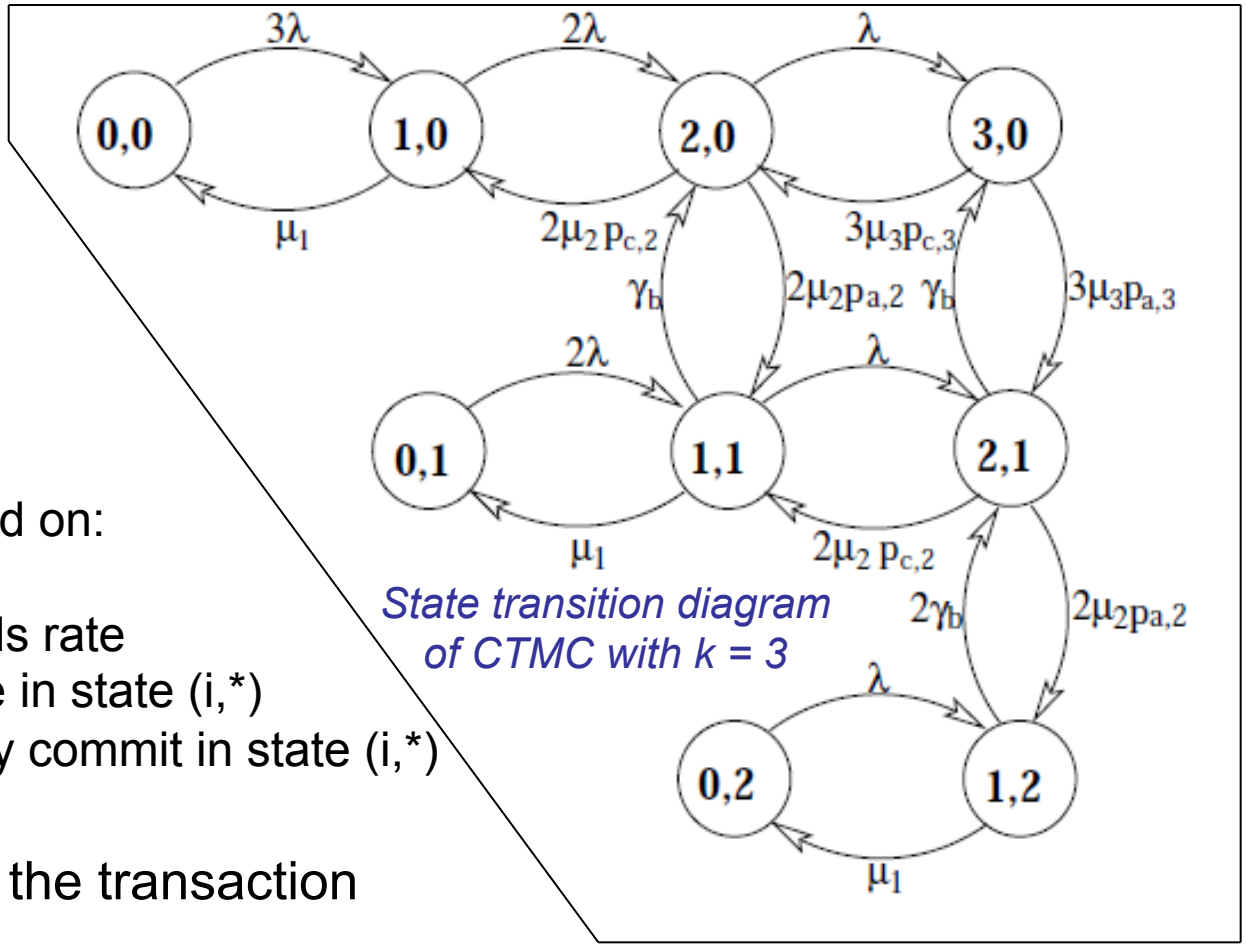
Performance modelling approach: data conflict model



t_: expected completion time (input into the model)

Threads execution is modeled via a Continuous Time Markov Chain (CTMC)

State *(i, j): i* is the number of threads which are running transactions and *j* the number of threads in back-off.



State transition diagram of CTMC with k = 3

CTMC transition rates depend on:

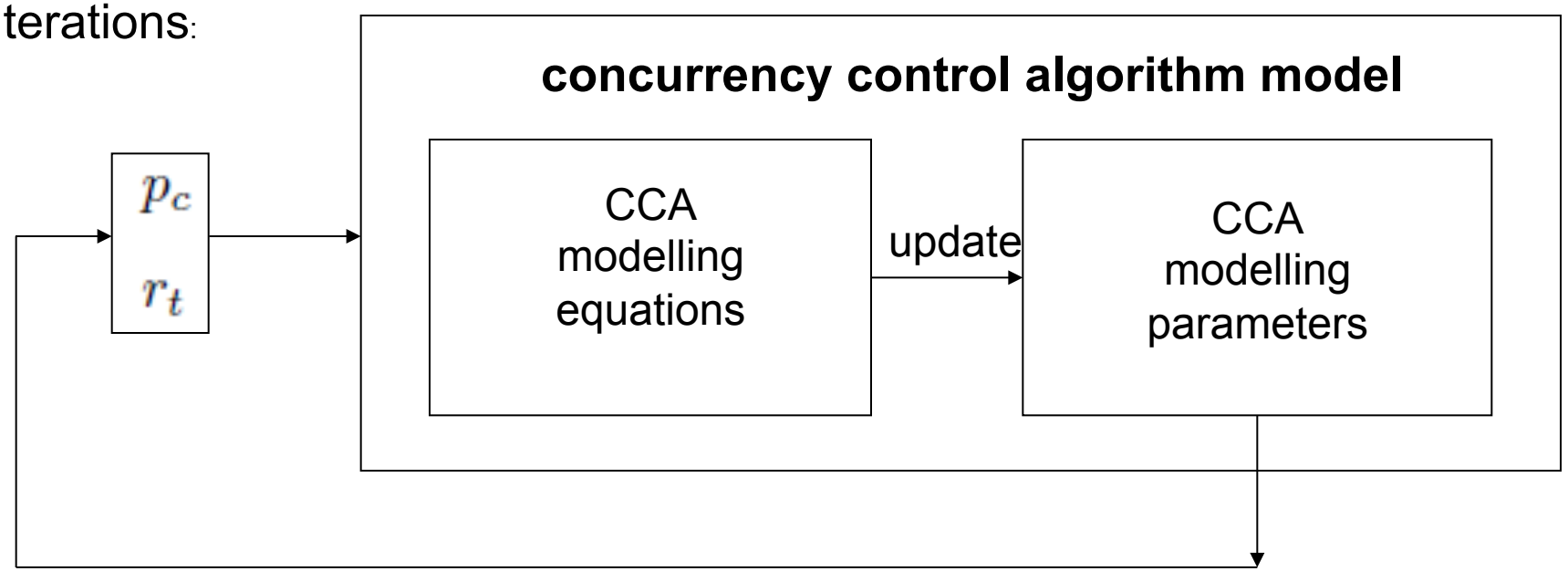
$\lambda = 1/ t_{ntcb}$: transaction arrivals rate
μ_i:(=1/ $r_{t,i}$) txs run service rate in state (i,*)
$p_{c,i}$: probability to successfully commit in state (i,*)

- μ_i and $p_{c,I}$ are input from the transaction modelling layer

For each state (i,*) of CTMC:

initial settings: $p_c = p_{c,i-1}$ $r_t = r_{t,i-1}$

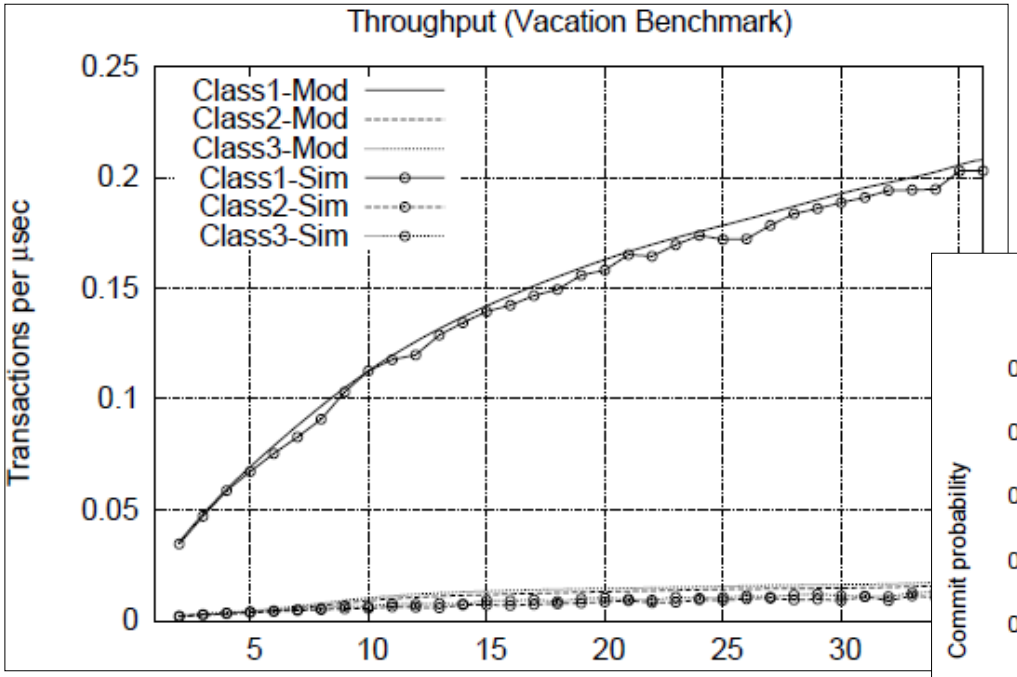
iterations:



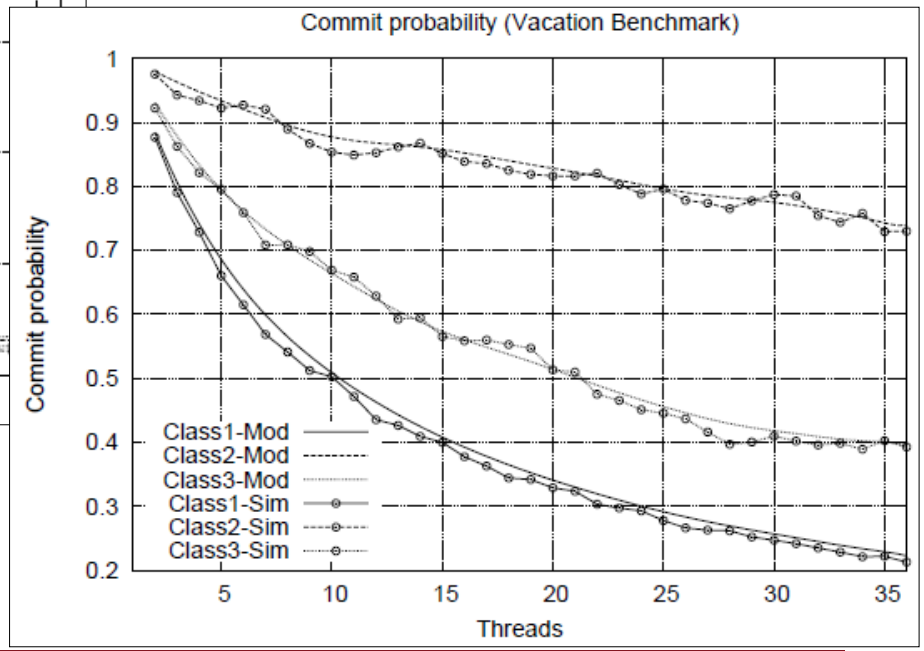
Iterations end when the difference between two consecutive values of p_c is < ϵ

Model Validation: Model vs. discrete event simulation

CCA: lazy locking + read validation (TL2)



testing workload:
- three transactional classes
- uniform data accesses



Thank you