ConsAD : A Real-Time Consistency Anomalies Detector

McGill

Kamal Zellag and Bettina Kemme

1. Introduction

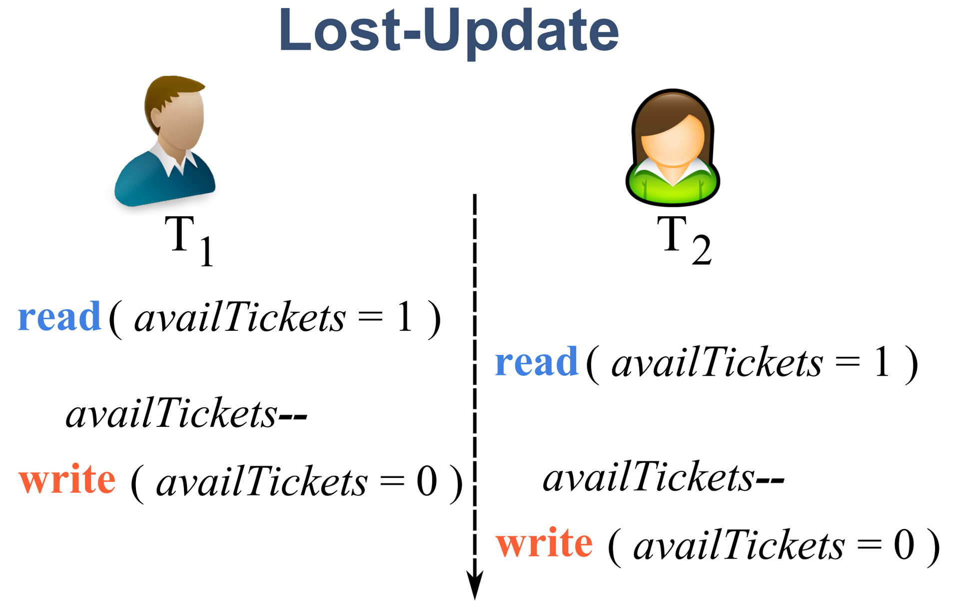
Transactional applications often choose **lower levels of isolation** than serializability to improve performance.
But how do lower levels affect consistency?

ConsAD detects consistency **anomalies** in a ***quantitative*** and ***qualitative*** manner and it:

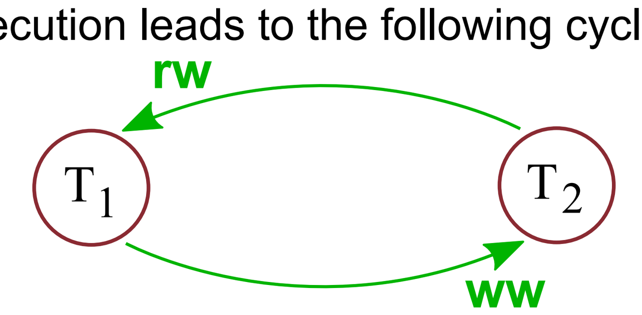
- is developed for multi-tier architectures.
- is transparently plugged into the application server tier.
- observes the application while it is running.
- detects *anomalies* (= ***cycles*** in the dependency graph) shortly after they occur.
- provides information about transactions and data items involved in anomalies.
- determines the culprit business methods and describes how they cause cycles.
- is independent of the database system.
- is independent of the application program under observation.

2. Example of an Anomaly

Lost-Update



- both transactions read the same value of available tickets.
- each transaction performs an update based on the read value.
- final value of available tickets does not reflect the amount of sold tickets.
- the execution leads to the following cycle:



- the isolation level ***read-committed*** allows this anomaly.
- the isolation levels ***snapshot-isolation*** and ***serializability*** avoid this anomaly.

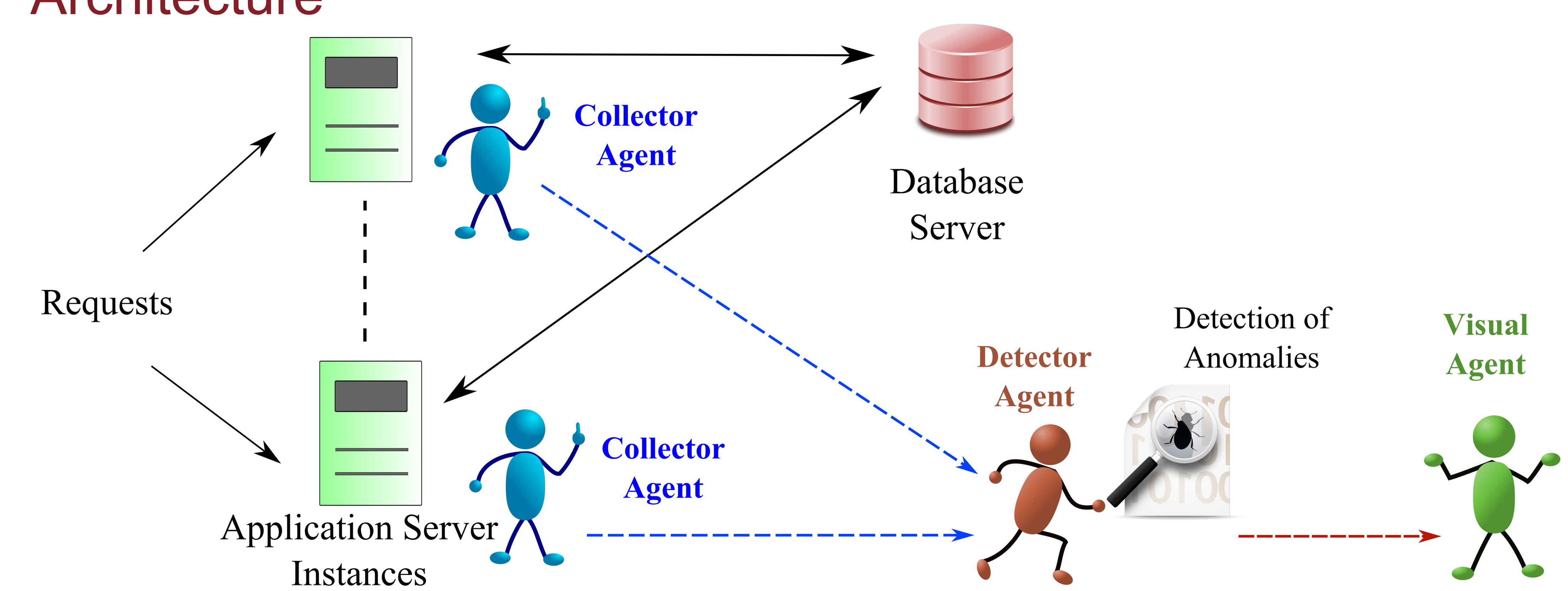
3. Approach

- we observe transactions and the data items they access (at the application server layer).
- based on observations, we build the *serialization graph incrementally* and detect cycles as soon as the last transaction in each cycle has committed and is added to the graph.

Serialization graph

- ***nodes*** are committed transactions.
- three types of ***edges***:
  - $T_i \xrightarrow{ww} T_j$: T_i writes a data item x and T_j writes the next version.
  - $T_i \xrightarrow{wr} T_j$: T_j reads what T_i has written.
  - $T_i \xrightarrow{rw} T_j$: T_i reads a data item x and T_j writes the next version.

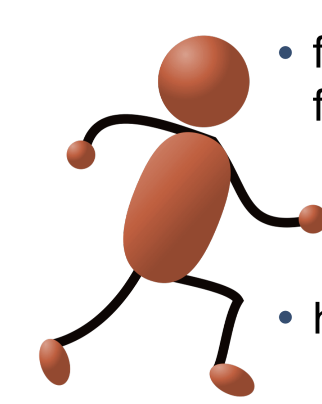
4. Architecture



5. The Collector Agent

- attached to application server instances.
- current implementation based on Java EE.
- adds an extra field *txnInfo* to each database table (transparent).
- when a transaction *T* with an identifier *T.id* updates a data item *x*, *x.txninfo* is set to *T.id*.
- observes transaction start, commit, reads and writes.
- sends transaction information immediately after its commit.
- no modification of application or application server needed.

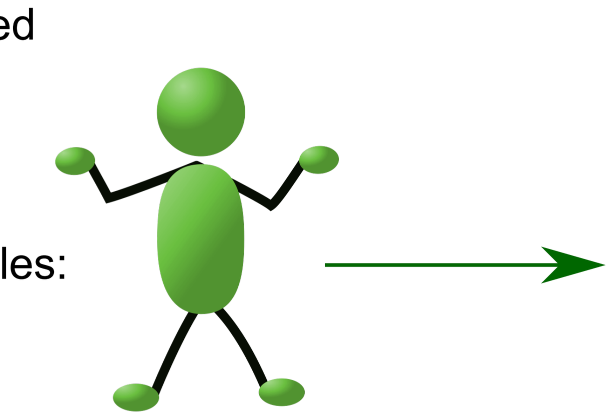
6. The Detector Agent



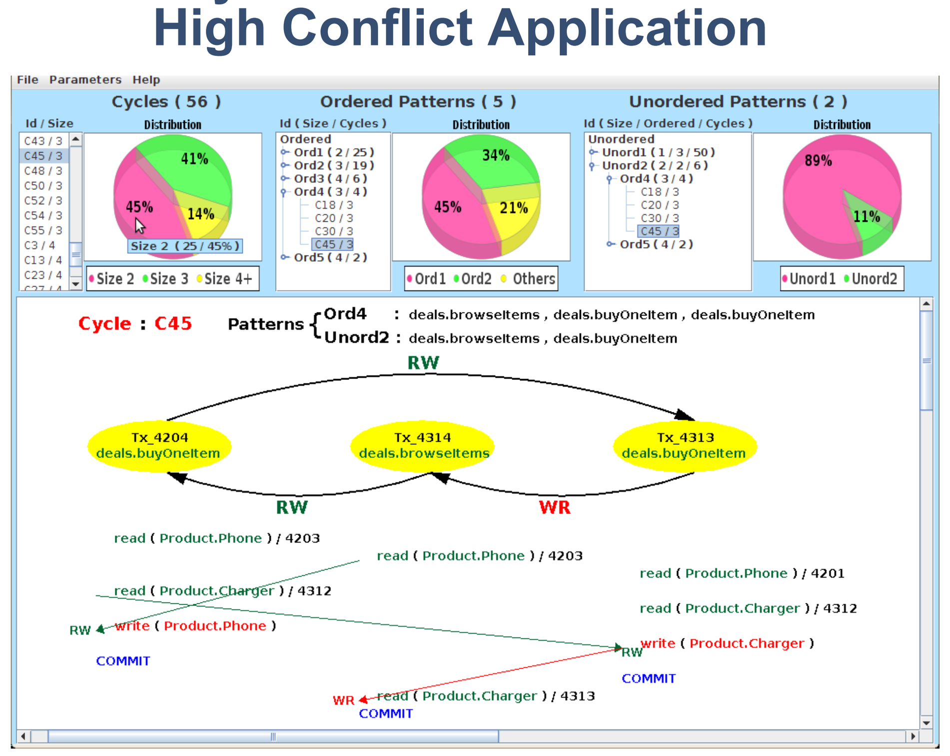
- for each transaction information received from Collector:
  - adds *transaction* and *edges* to the graph.
  - detects new *cycles*.
- highly optimized ***Depth First Search***.
- extracts *business methods* creating cycles.

7. The Visual Agent

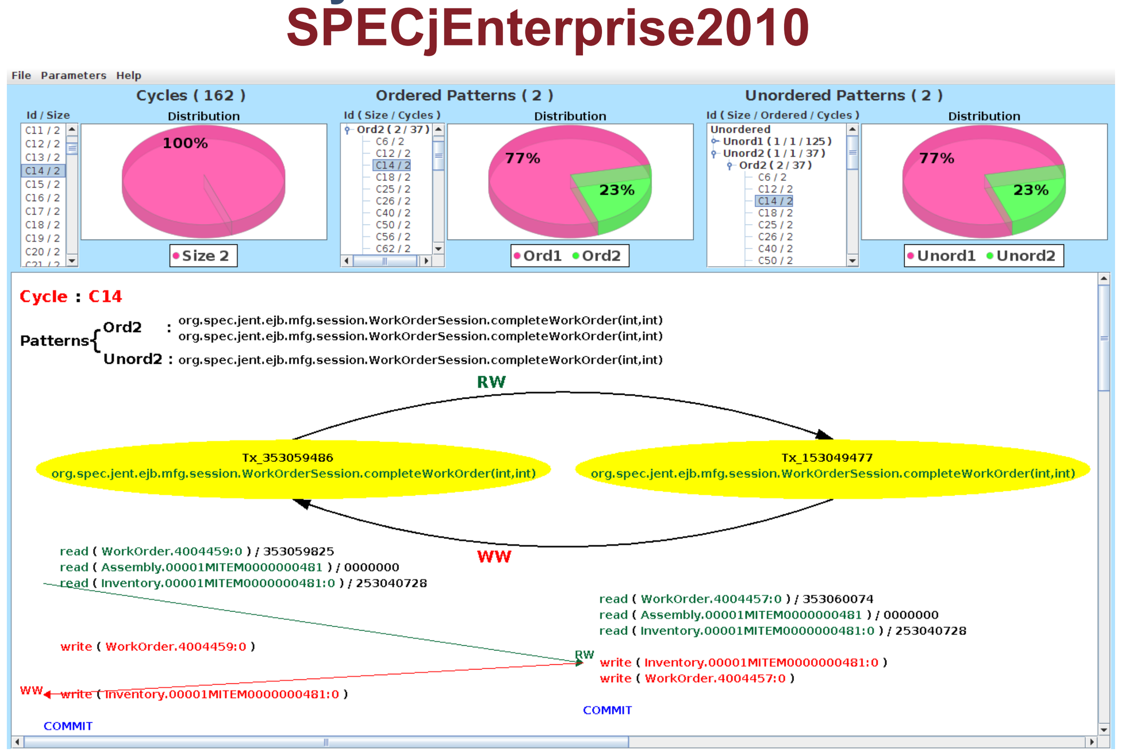
- displays summary information:
  - total *amount of cycles* sorted by their length.
  - information about *patterns* (see next section).
- shows details on individual cycles:
  - transactions identifiers.
  - edges and their types.
  - business methods.
  - data items identifiers.
- uses ***Topological Sort*** for time-based visualization.



Cycles for an *Emulated* High Conflict Application



***Real* Cycles for the Benchmark SPECjEnterprise2010**



8. Patterns Classification

- Typically each transaction is called within the borders of a *business method*.
- The Detector *maps* each cycle of transactions to:
  - a cycle C_m (***ordered pattern***) of their respective methods (in the *same order*).
  - a set C_{set} (***unordered pattern***) of unique methods in C_m (*no order* / *no duplication*).
- The detection of patterns helps designers to locate the origin of anomalies in their application.

9. Future Work

- extension to ***cloud platforms***.
- extension to ***non-transactional*** consistency criteria.

10. Selected References

[1] A. Adya, B. Liskov, and P. E. O'Neil. Generalized isolation level definitions. In ICDE, pages 67-78, 2000.

[2] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil. A critique of ANSI SQL isolation levels. In ACM SIGMOD Conf., 1995.

[3] A. Bernstein, P. Lewis, and S. Lu. Semantic conditions for correctness at different isolation levels. In Proceedings of IEEE ICDE, pages 57-66, 2000.

[4] K. Zellag and B. Kemme. Real-time quantification and classification of consistency anomalies in multi-tier architectures. In ICDE, pages 613-624, 2011.