Gargamel: A Conflict-Aware Contention Resolution Policy for STM

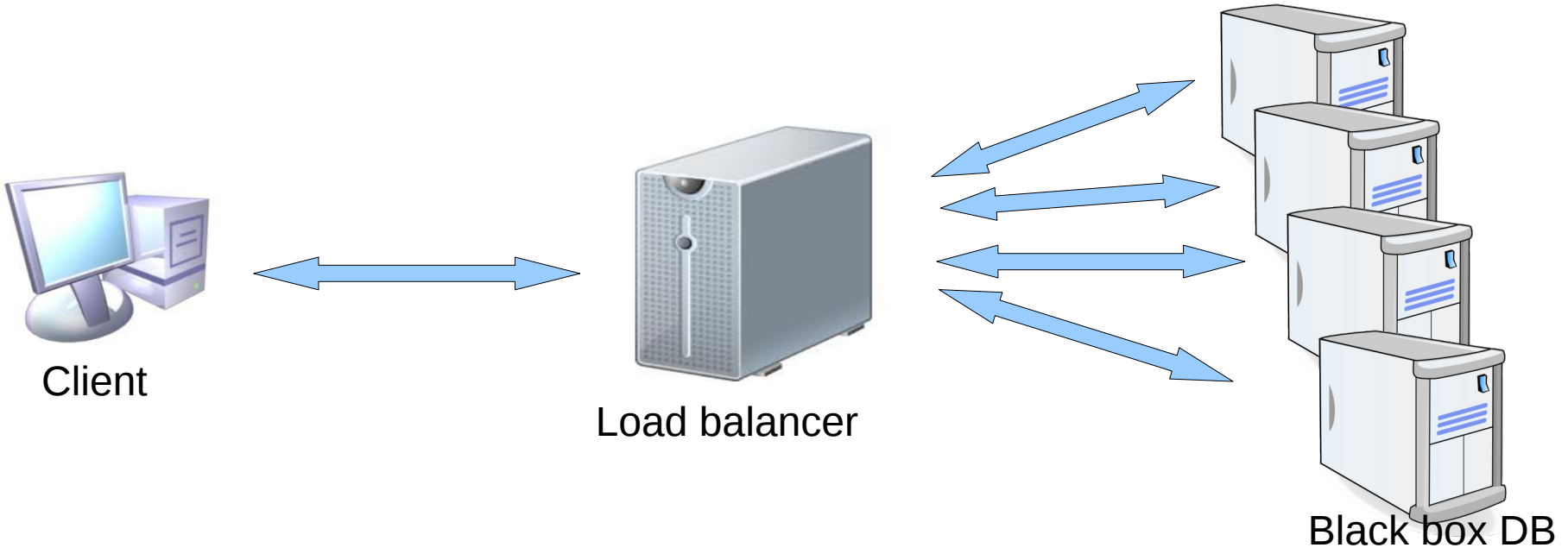
Pierpaolo Cincilla, Marc Shapiro, Sébastien Monnet

- Transactions in a TM execute speculatively:
  - Commit if no conflicts
  - Abort otherwise

Low contention => Good performance

High contention => Repetitive aborts, poor performance

- Single thread DBs perform well
- Scale => parallel, distributed



Concurrency control, contention

=>

Distributed configurations scale poorly

Current approaches:

- Contention Manager } Acts after conflict detection => Do not avoid wasted work
- TM schedulers } Avoid conflicts to appear => Avoid wasted work

Current approaches : Contention Manager

- Polite
- Karma
- Greedy
- Serializer

} Acts after conflict detection => Do not avoid wasted work

- TM schedulers

} Avoid conflicts to appear => Avoid wasted work

Current approaches

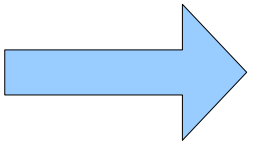
- Memory partition => No global transactions
- Centralise writes
  - Read-dominated workloads

} => SI

- Weaken semantics (key-value store, NoSQL)
  - Problematic for applications

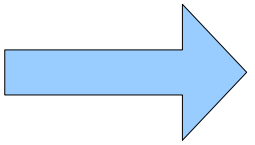
Intuition

Throughput-optimal =

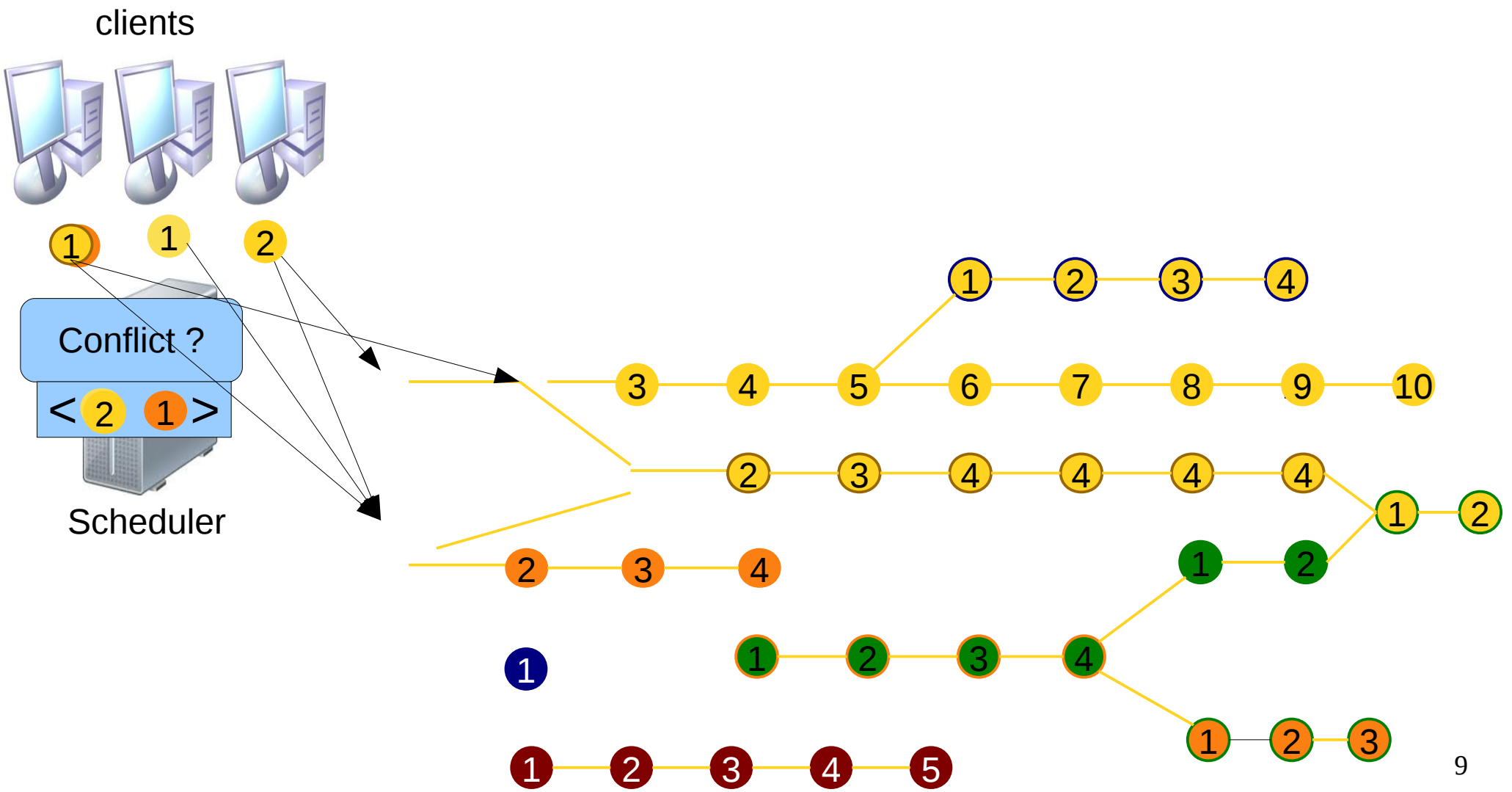
- No concurrent conflicting transactions => no aborts  Conflict estimation
- 1 site = 1 scheduler + n workers threads
  - Conflicting => sequential "chain"
  - Non conflicting => parallel

Intuition

Throughput-optimal =

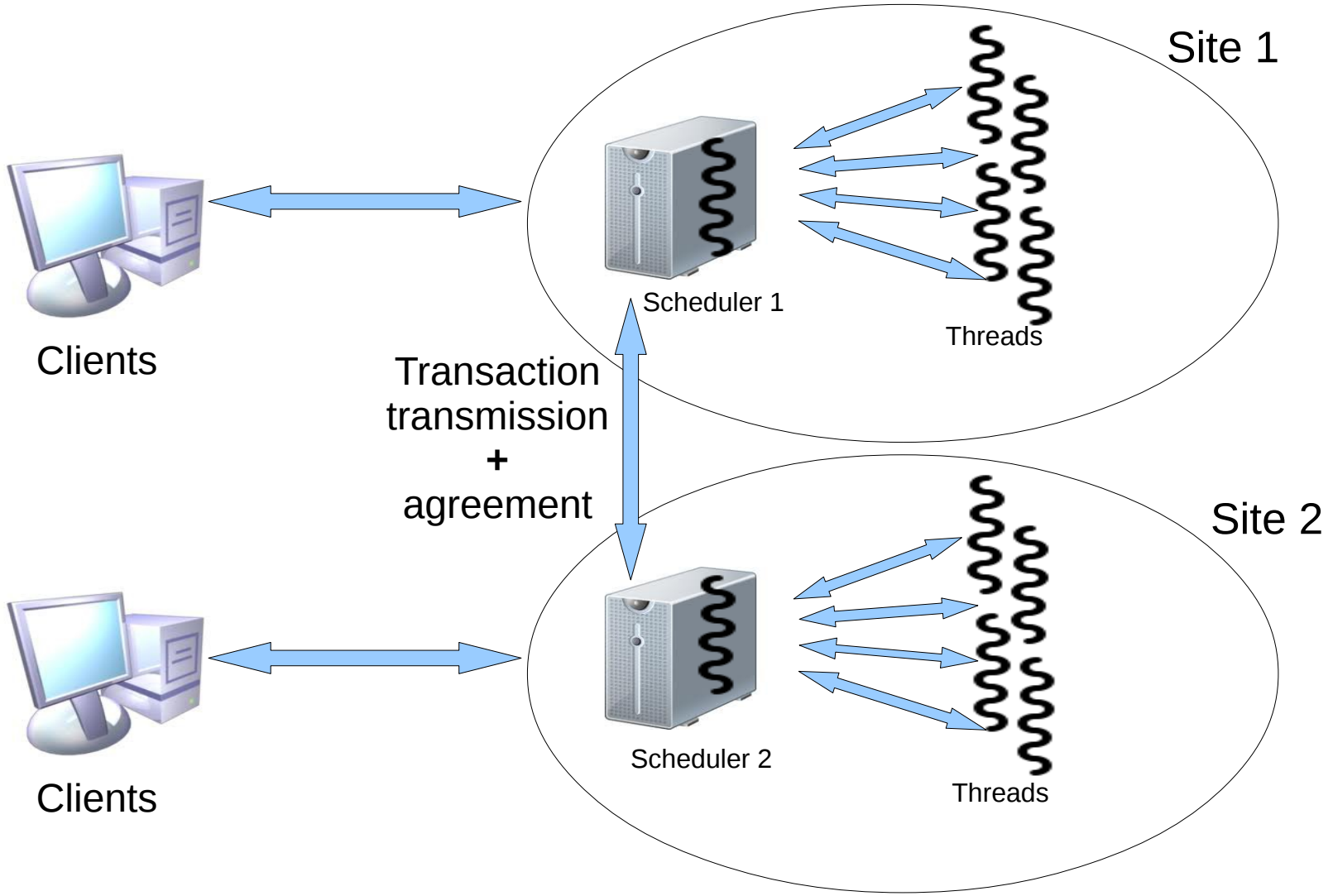
- No concurrent conflicting transactions => no aborts  Conflict estimation
- 1 site = 1 scheduler/load balancer + n workers
  - Conflicting => sequential "chain"
  - Non conflicting => parallel

The Gargamel scheduling algorithm

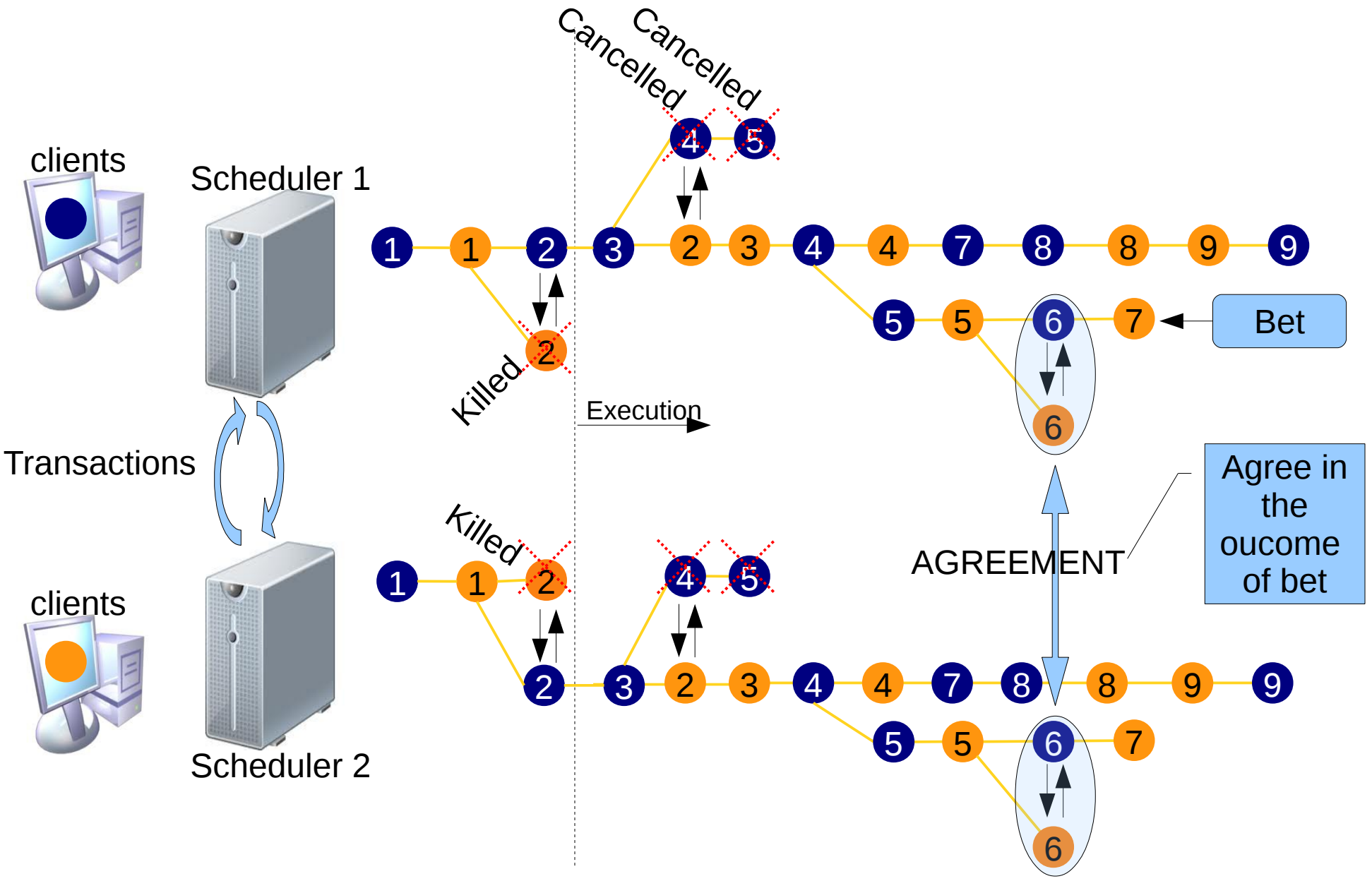


Architecture

Fault tolerance, latency, load => several sites



The Gargamel scheduling algorithm (2)



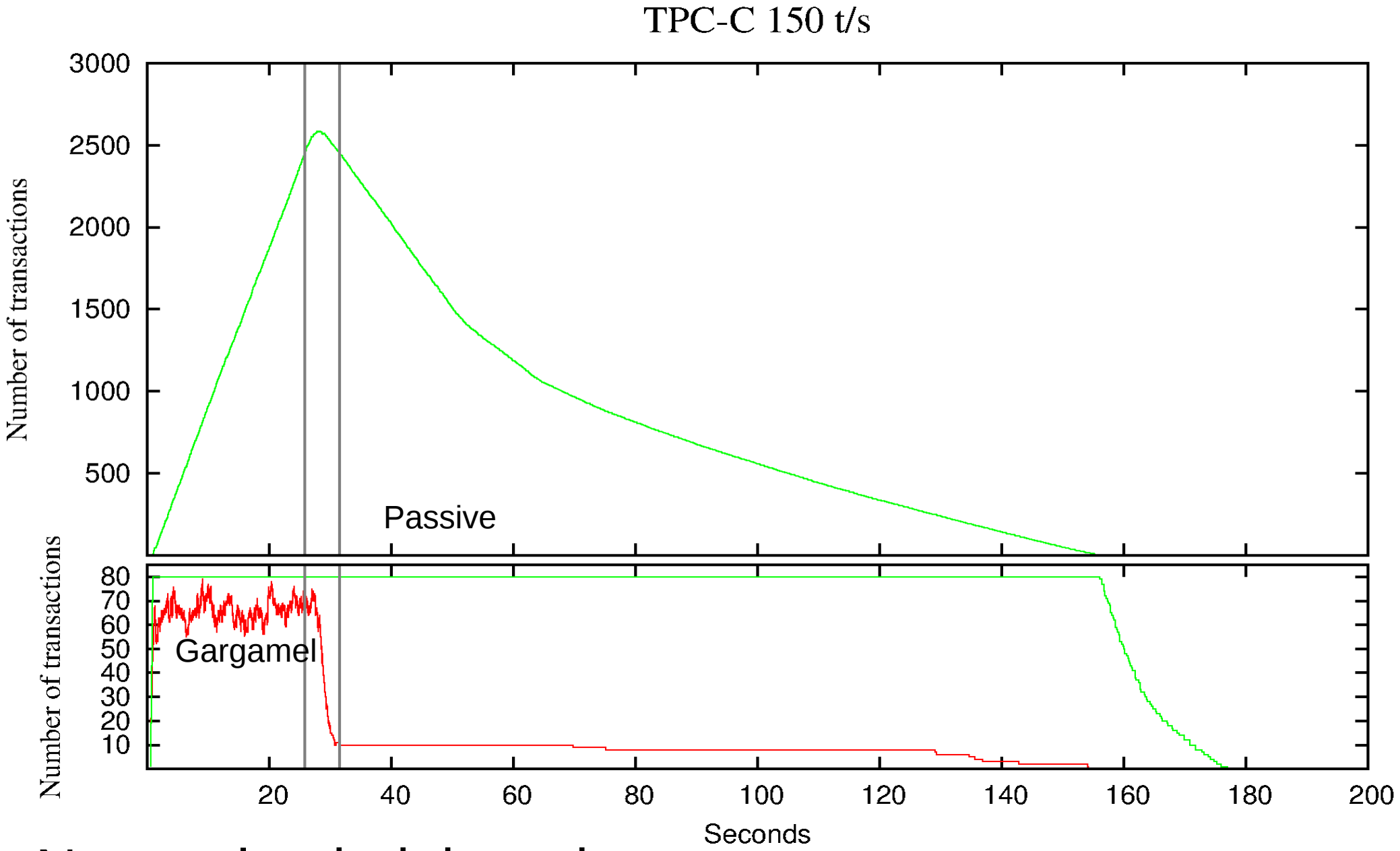
Simulation results

- Simulation engine
- Simulated workload : TPC-C
  - Vary = number of sites, distance between sites (latency within sites =0)
  - Constant : workload

Measure:

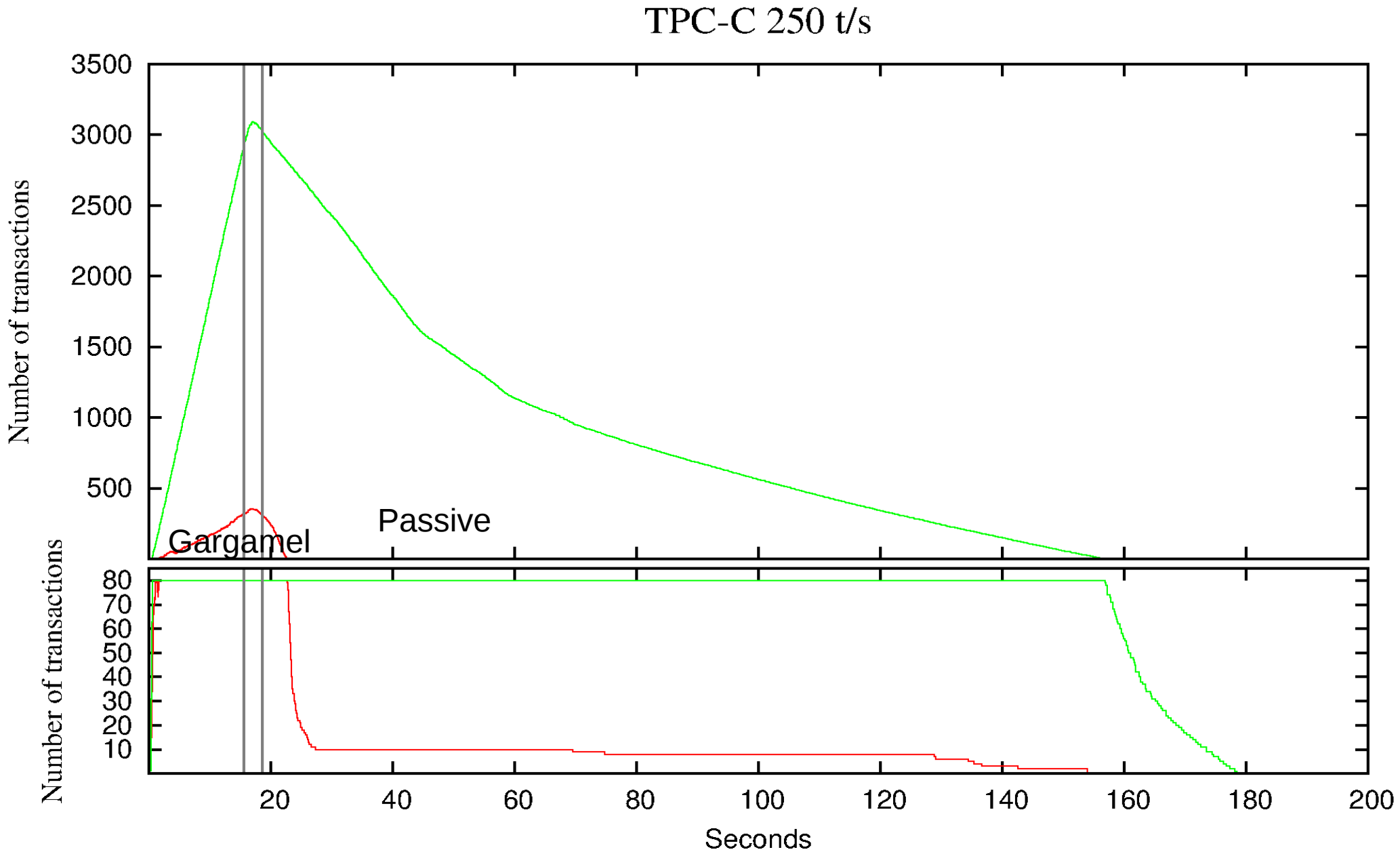
- Resource usage
  - Number of running threads in ressources bounded systems
- Price of concurrency
  - Cancel/kill rate with respect to number of sites, message

Resource usage (TPC-C)



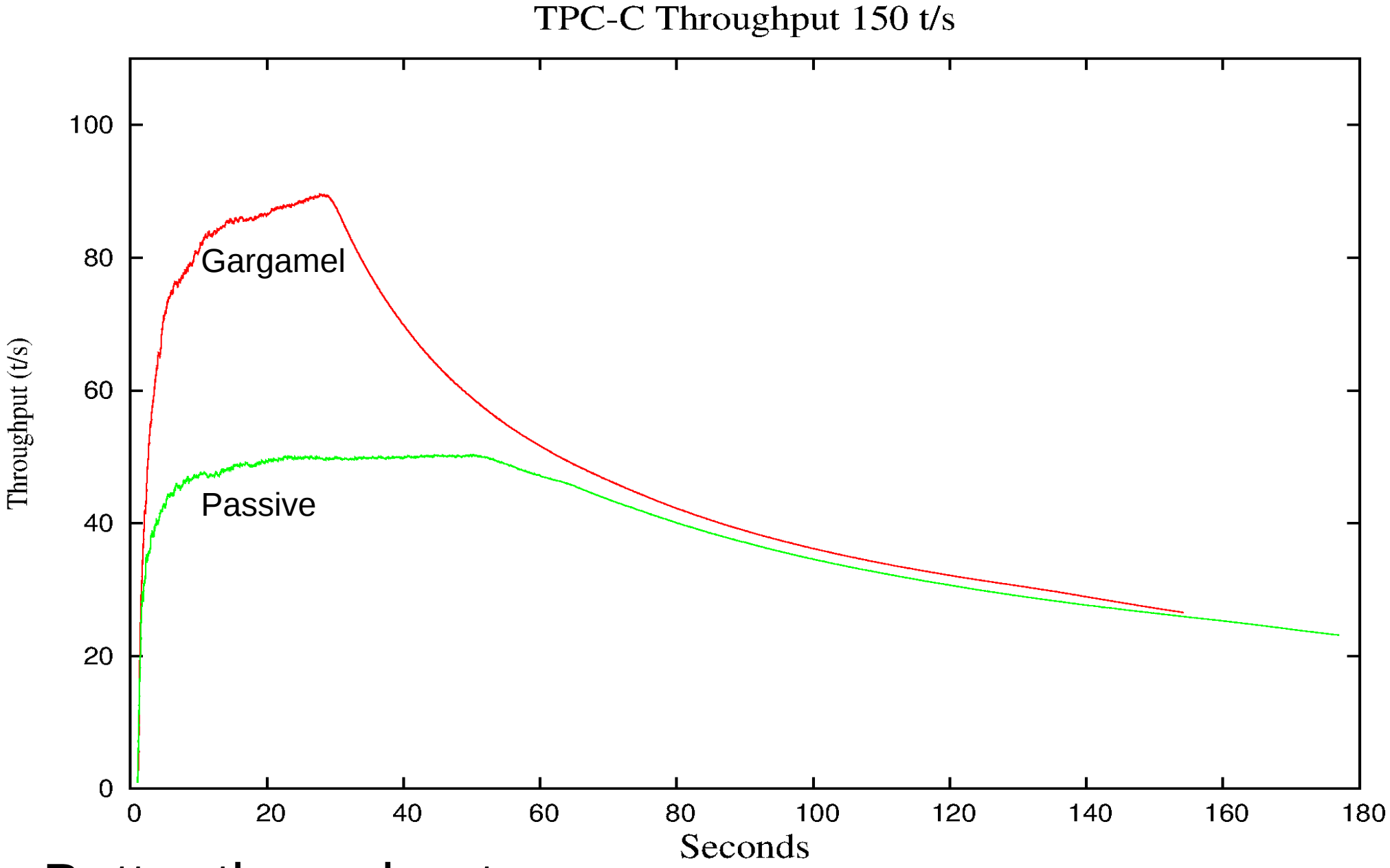
- No overloaded threads

Resource usage (TPC-C)



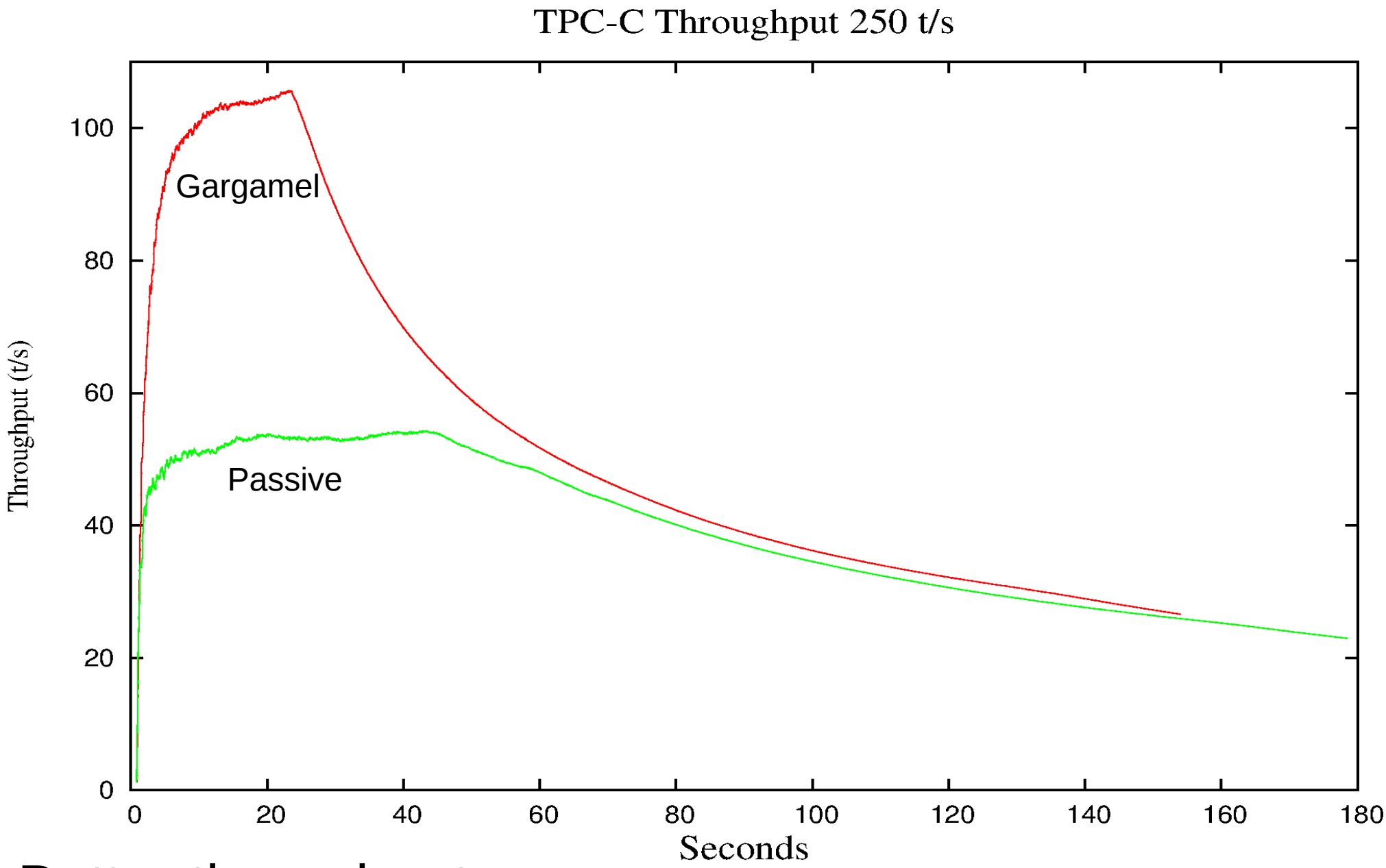
- No overloaded threads

Throughput (TPC-C)



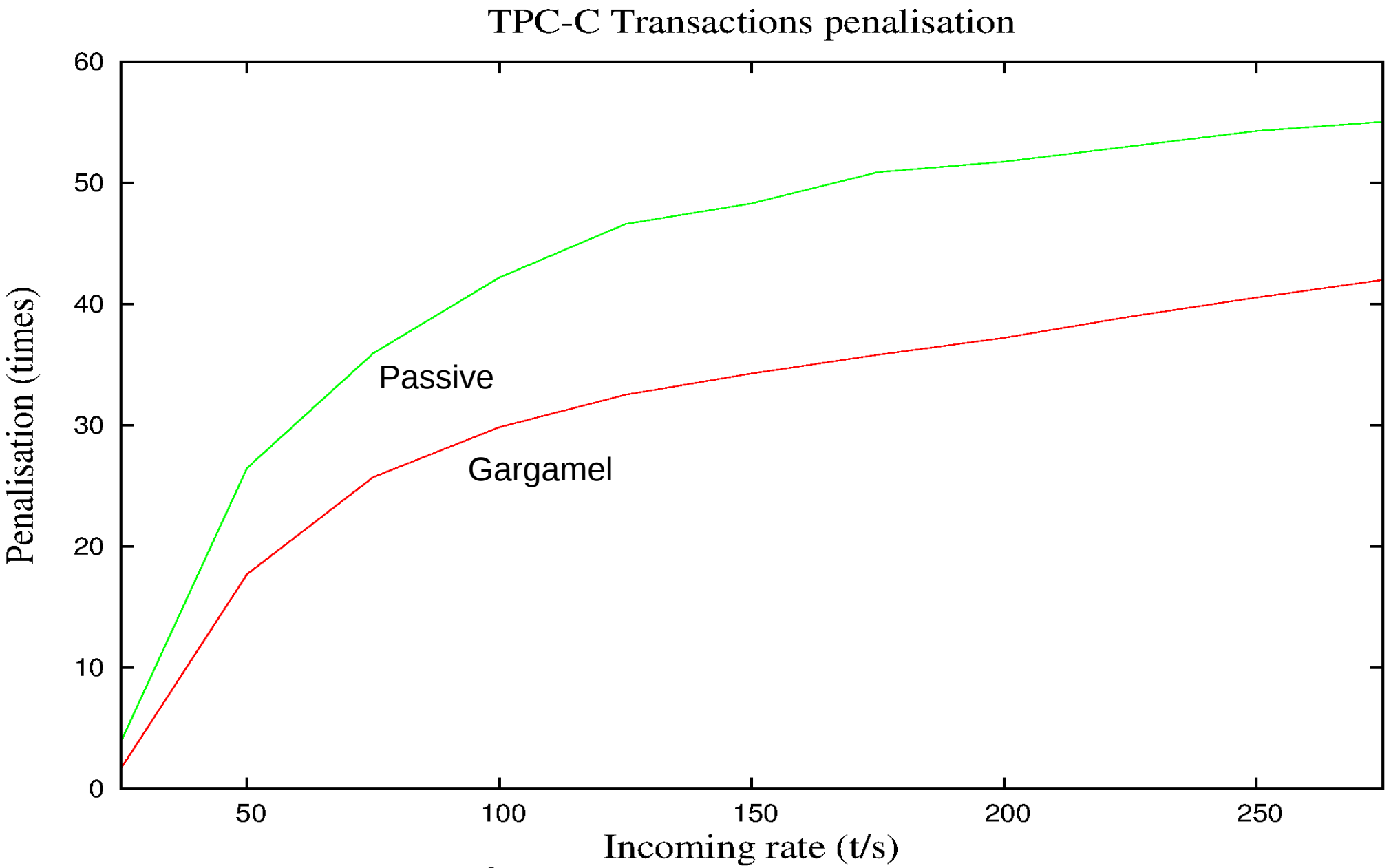
- Better throughput

Throughput (TPC-C)



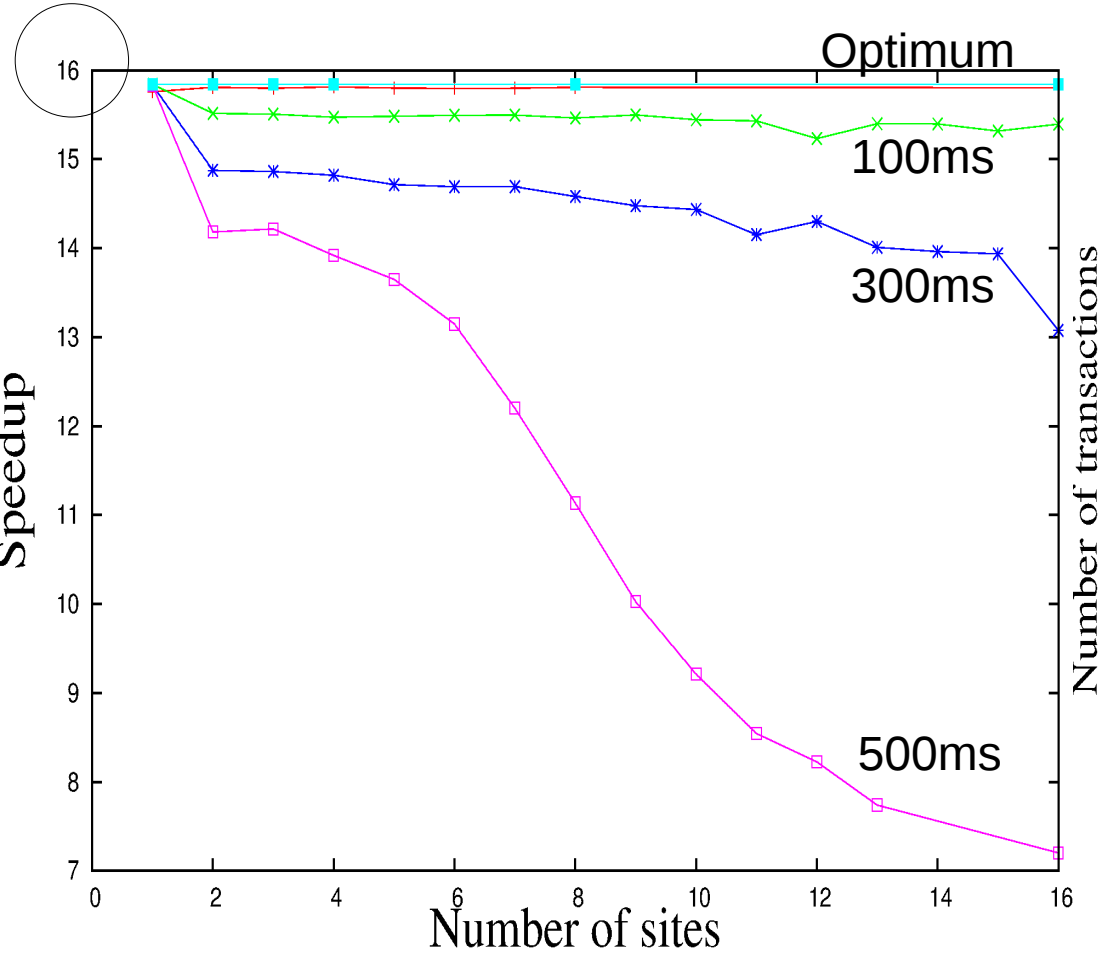
- Better throughput

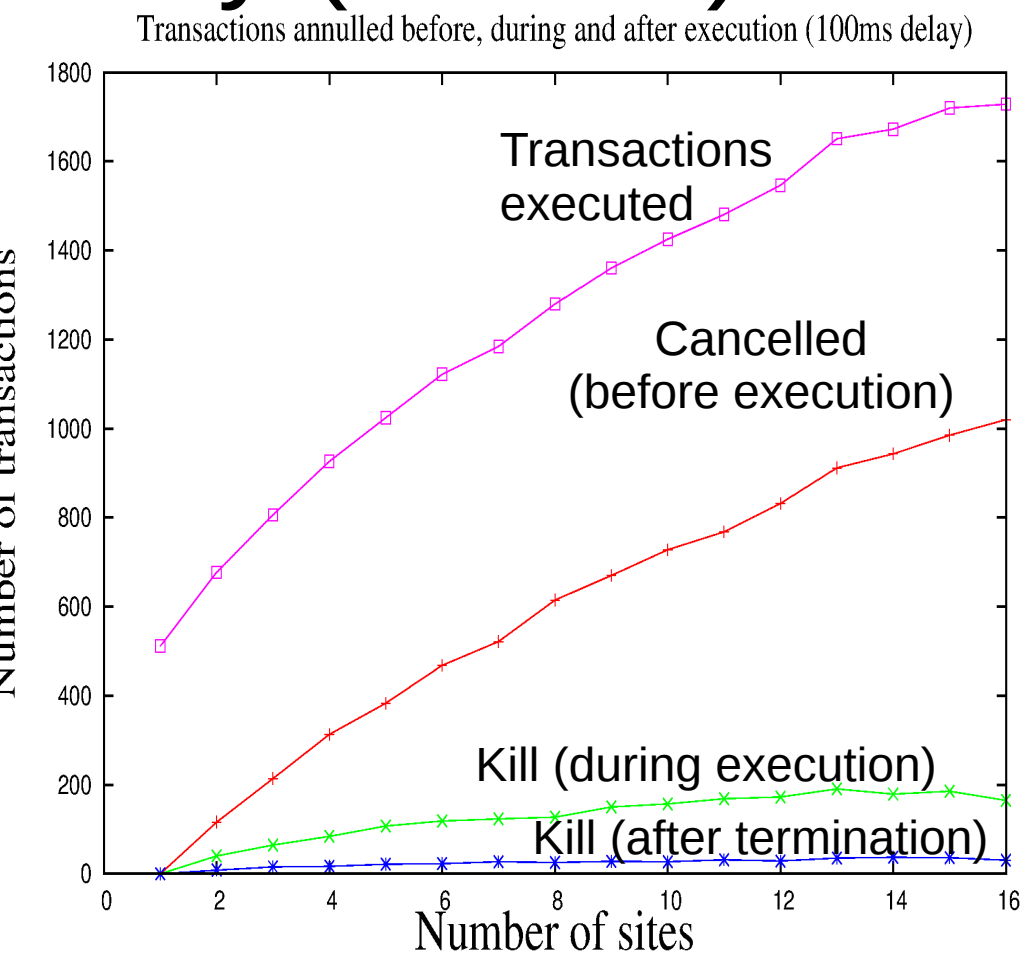
Penalisation (TPC-C)



- Better response time

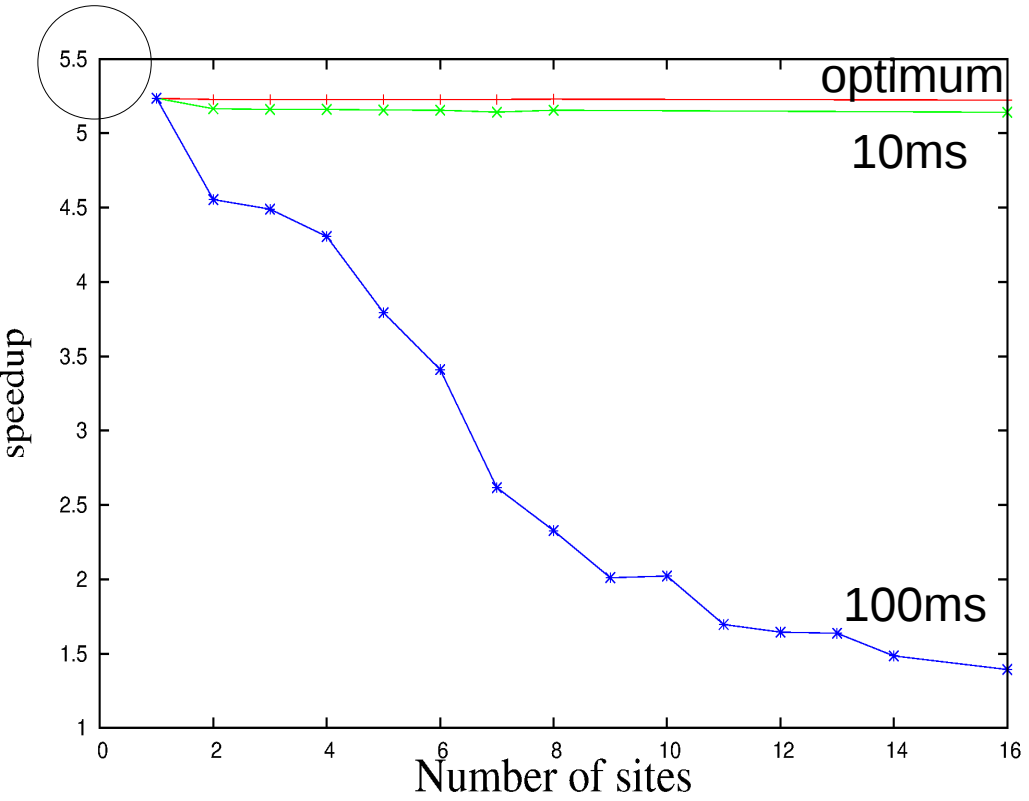
Price of concurrency (TPC-C)

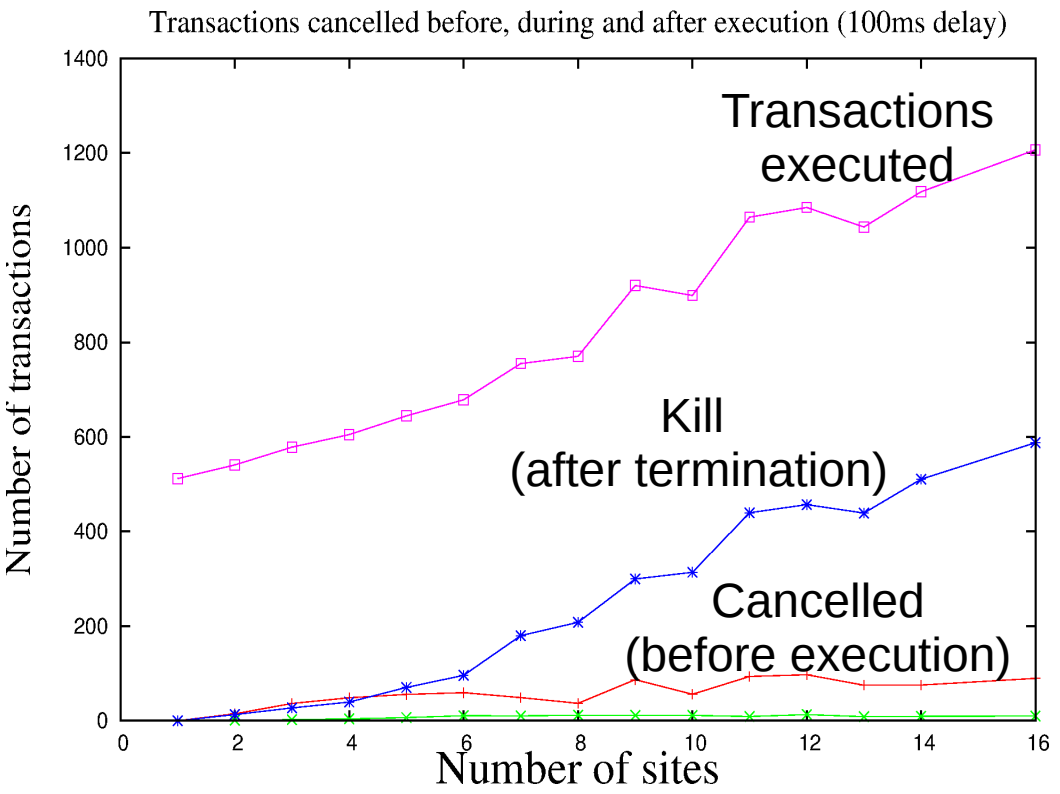




- Good speedup even at high latency (up to 100ms)
- Kills are rare

Price of concurrency (TPC-E)





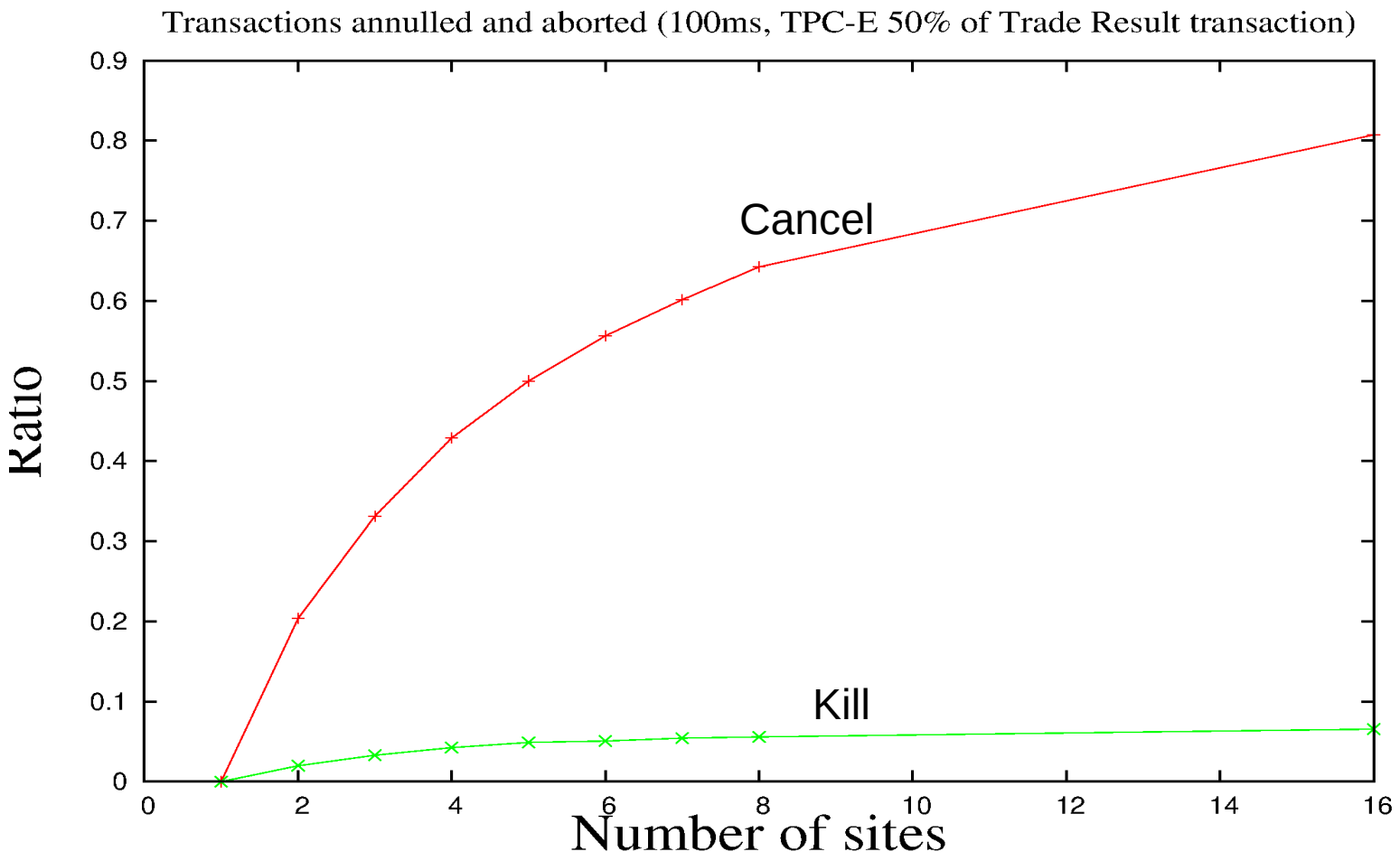
"Trade Result" transaction is the bottleneck

- Conflict parameter not available to oracle
- TODO: add parameter to TPC-E

Impact of message delay

Higher kill/annullation rate in TPC-E

- Different incoming rate of bottleneck transaction (TPC-C : 88%, TPC-E 10%)



Contributions

- Conflict estimation = w/w conflicts = static analysis
- Chains = structured scheduler queues
- Load balancing + concurrency control = schedules
- Preventively parallelize unconflicting chains

Lesson learned

- Measure of optimum = single site, compare multiple site
- No concurrent conflicting transactions => no aborts
- Simulation result = benefits proportional to oracle accuracy and incoming rate
- Schedules as optimal as estimator is

Lesson learned

- Measure of optimum = single site, compare multiple site
- No concurrent conflicting transactions => no aborts
- Simulation result = benefits proportional to oracle accuracy and incoming rate
- Schedules as optimal as estimator is
- Trade-off black box STM / agreement off critical path
- Share-nothing => no resource contention

Lesson learned

- Measure of optimum = single site, compare multiple site
- No concurrent conflicting transactions => no aborts
- Simulation result = benefits proportional to oracle accuracy and incoming rate
- Schedules as optimal as estimator is
- Trade-off black box DB / agreement off critical path
- Share-nothing => no resource contention

Future Work

- Simulate Gargamel VS Tashkent+, and LARD (1m)
- Implementation + evaluation with real workloads (10m)
- Partial replication (similar to Tashkent+, comes almost for free) (1m)
- False negative => handle DB aborts (1m)
- Machine learning oracle (1m)
- Thesis (6m)