Conflict-driven Load Control for the Avoidance of Data-Contention Thrashing

Presentation of a paper by A. Moenkeberg and G. Weikum from the International Conference on Data Engineering (ICDE'91)

Ken Salem

David R. Cheriton School of Computer Science
University of Waterloo

November 8, 2006

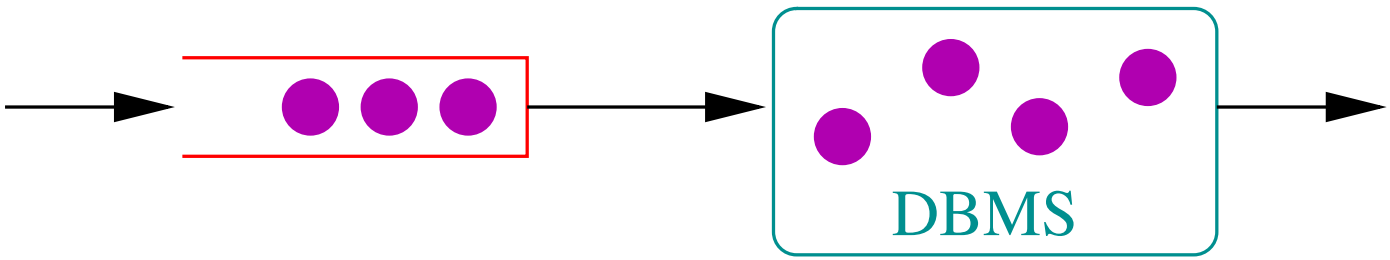
The Problem

The Problem: data contention may cause a DBMS to perform poorly if too many transactions run concurrently.

The Problem

The Problem: data contention may cause a DBMS to perform poorly if too many transactions run concurrently.

The Solution: limit the number of concurrently executing transactions



Control Framework

- What is the actuator?
  - transaction admissions

Control Framework

- What is the actuator?
  - transaction admissions
- What is measured?

Control Framework

- What is the actuator?
  - transaction admissions
- What is measured?
  - Control objective is to maximize throughput, avoiding thrashing. Could directly measure throughput.

Control Framework

- What is the actuator?
  - transaction admissions
- What is measured?
  - Control objective is to maximize throughput, avoiding thrashing. Could directly measure throughput.
  - Problem: what reference value to use? What is the target throughput?

Control Framework

- What is the actuator?
  - transaction admissions
- What is measured?
  - Control objective is to maximize throughput, avoiding thrashing. Could directly measure throughput.
  - Problem: what reference value to use? What is the target throughput?
  - Proposal: measure conflict rate, and keep it below a specified target value.

Control Framework

- What is the actuator?
  - transaction admissions
- What is measured?
  - Control objective is to maximize throughput, avoiding thrashing. Could directly measure throughput.
  - Problem: what reference value to use? What is the target throughput?
  - Proposal: measure conflict rate, and keep it below a specified target value.

Observation

This turns a dynamic optimization problem into a regulation problem.

Conflict Rate

$$\text{conflict rate} = \frac{\text{\# locks held by all transactions}}{\text{\# locks held by non-blocked transactions}}$$

- conflict rate ≥ 1
- conflict rate $= 1$ when no active transactions are blocked
- conflict rate increases as more transactions block

Conflict Rate

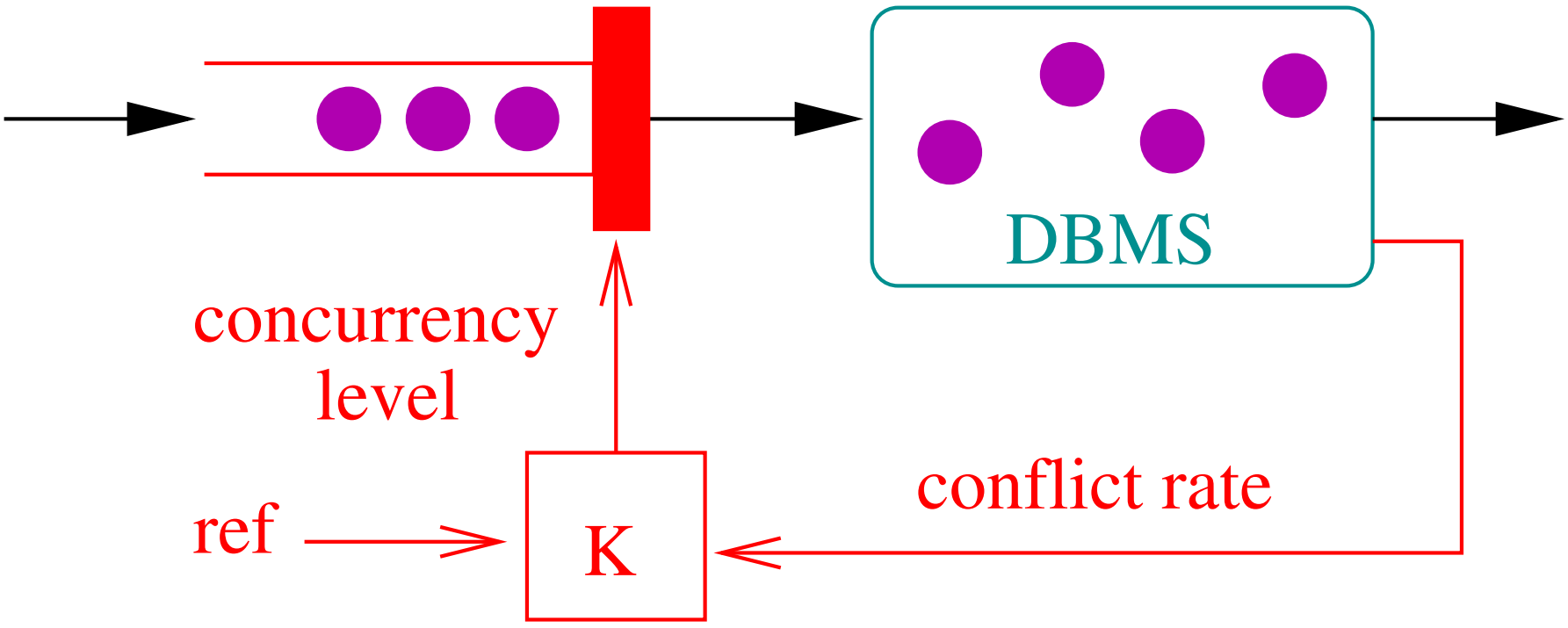
$$\text{conflict rate} = \frac{\text{\# locks held by all transactions}}{\text{\# locks held by non-blocked transactions}}$$

- conflict rate ≥ 1
- conflict rate $= 1$ when no active transactions are blocked
- conflict rate increases as more transactions block

Claim

conflict rate ≥ 1.3 implies data-contention thrashing, regardless of the workload

Regulating Conflict Rate



Proposed Control Rules

- irregular control interval
  - control applied when a lock request blocks and when a transaction request arrives/finishes

Proposed Control Rules

- irregular control interval
  - control applied when a lock request blocks and when a transaction request arrives/finishes
- when a transaction arrives/finishes, consider admitting new transactions

  A0: admit one
  A1: admit all
  A2: admit some (until projected conflict rate reaches limit)

Proposed Control Rules

- irregular control interval
  - control applied when a lock request blocks and when a transaction request arrives/finishes
- when a transaction arrives/finishes, consider admitting new transactions

  A0: admit one
  A1: admit all
  A2: admit some (until projected conflict rate reaches limit)
- when transaction blocks, consider aborting transactions

  C0: abort none
  C1: abort one
  C2: abort some (until conflict rate is below limit)

Comments

- 1.3???

Comments

- 1.3???
- In general, there may be multiple potential bottlenecks in the DBMS:
  - lock contention
  - CPU contention
  - disk contention

Comments

- 1.3???
- In general, there may be multiple potential bottlenecks in the DBMS:
  - lock contention
  - CPU contention
  - disk contention
- Heiss and Wagner (VLDB'91) attempt to solve the dynamic optimization problem directly