Distributed Commit Processing in Firm Real-Time Database Systems

Ramesh Gupta	**IISc, Bangalore**
Jayant Haritsa	**IISc, Bangalore**
Krithi Ramamritham	**UMass, Amherst**
S. Seshadri	**IIT, Bombay**

Motivation

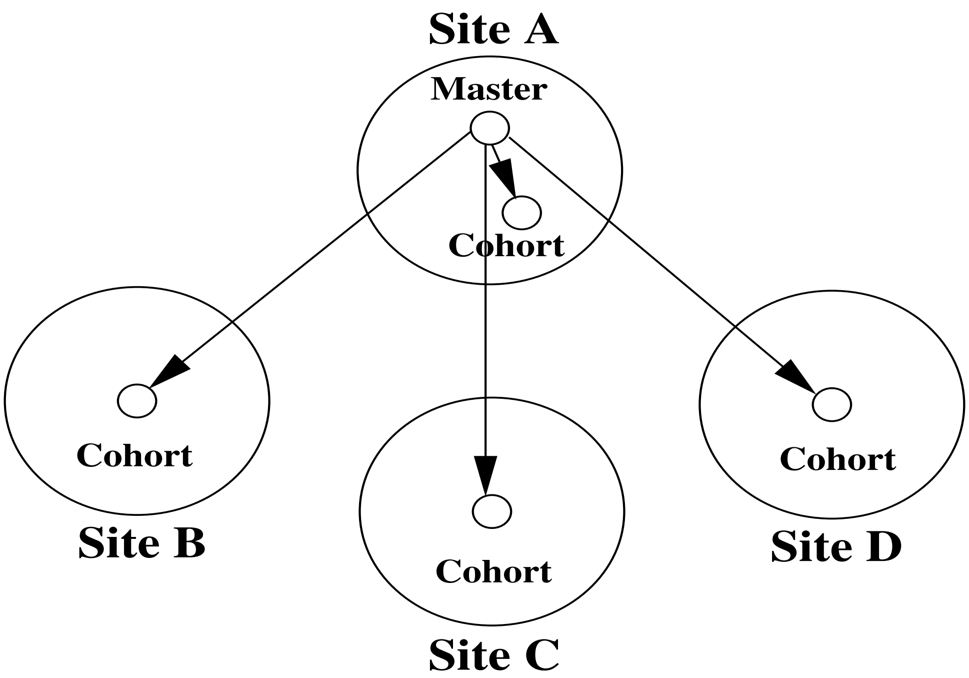
- Many real-time database applications are inherently distributed
  (e.g., communication systems, military systems)

- Introduces new complexities in ensuring transaction atomicity and serializability

- Prior research focussed primarily on distributed real-time serializability

- Little work on distributed real-time atomicity

Distributed Transaction

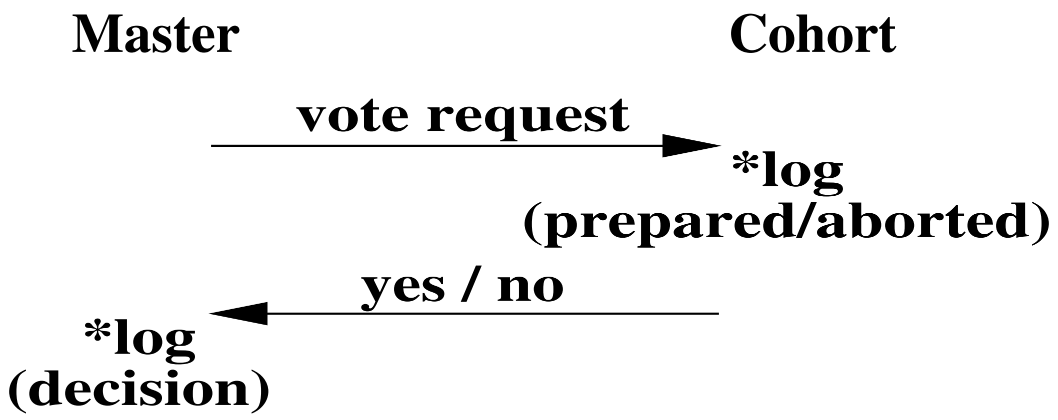


- *Atomicity* : Globally Consistent Decision (Commit or Abort)

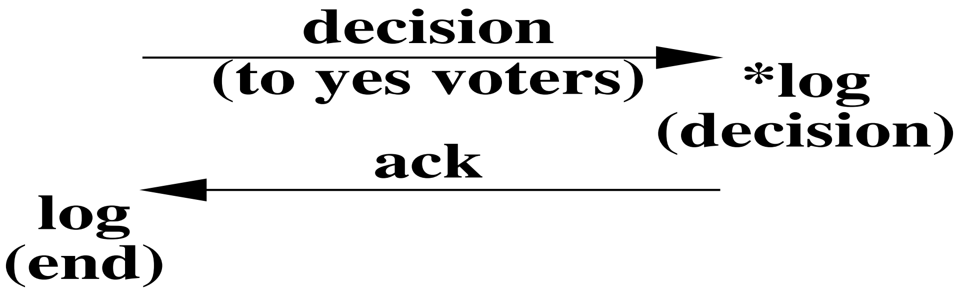
- **Mechanism :** Commit Protocols (2PC, PA, PC, 3PC, . . .)

Two Phase Commit (2PC)

- **Phase I : Reach a decision**



- **Phase II : Implement the decision**



Overheads:

- **Synchronous Execution**
- **Several Messages and Log Records**

(***:** *forced write*)

2PC Optimizations

Presumed Abort (PA):

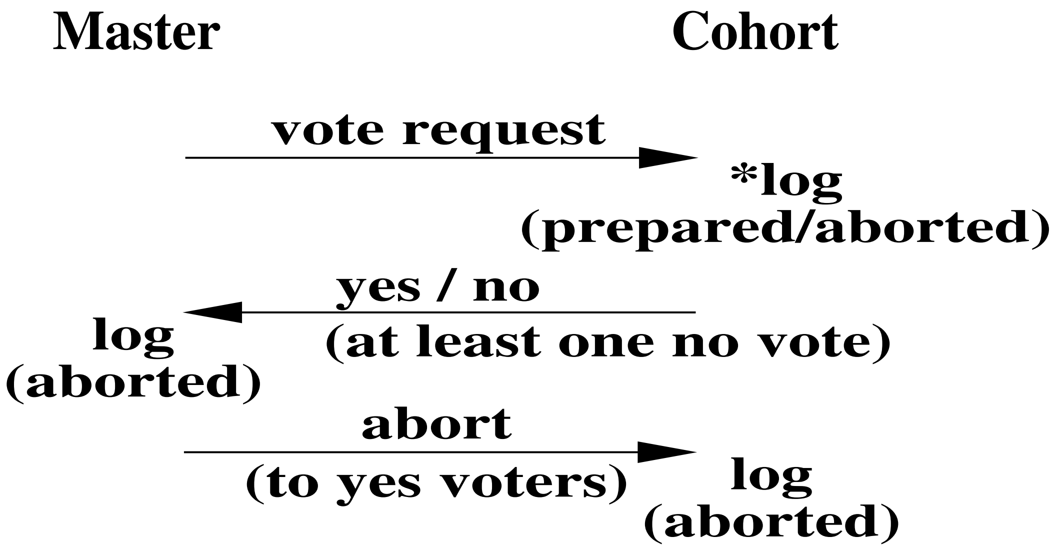
- **After a failure, in case of doubt, cohorts assume that a transaction was *aborted*.**

Presumed Commit (PC):

- **Transactions typically commit, rarely abort**
- **After a failure, in case of doubt, cohorts assume that a transaction *committed*.**

Presumed Abort (PA)

- **After a failure, in case of doubt, cohorts assume that a transaction was *aborted*.**
- **Abort Processing :**

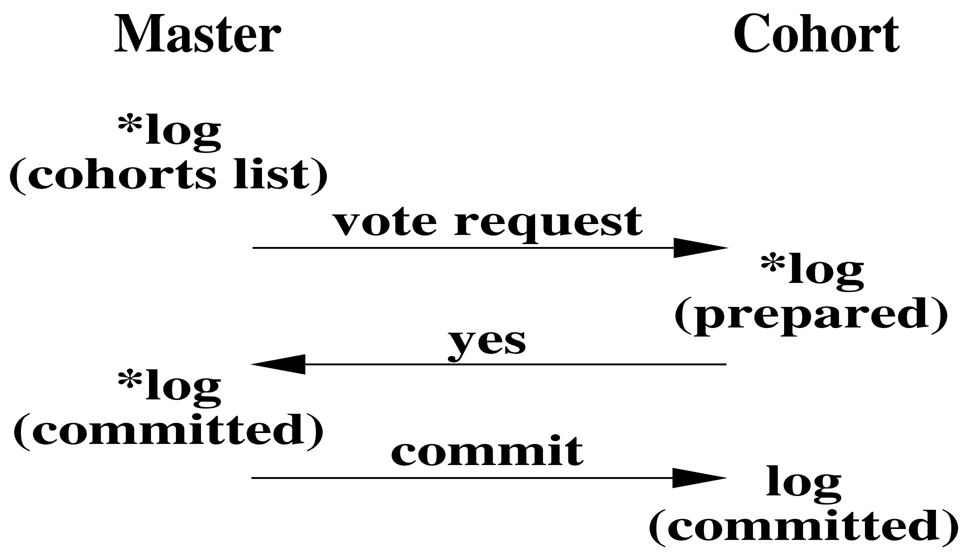


  - **No Acks for Abort**
  - **"aborted" log record not forced**
  - **No "end" log record**

- **Commit Processing same as in 2PC**

Presumed Commit (PC)

- Transactions typically commit, rarely abort
- After a failure, in case of doubt, cohorts assume that a transaction *committed*.
- Commit Processing :



  – No Acks for committing transactions

  – "committed" log record at cohorts not forced

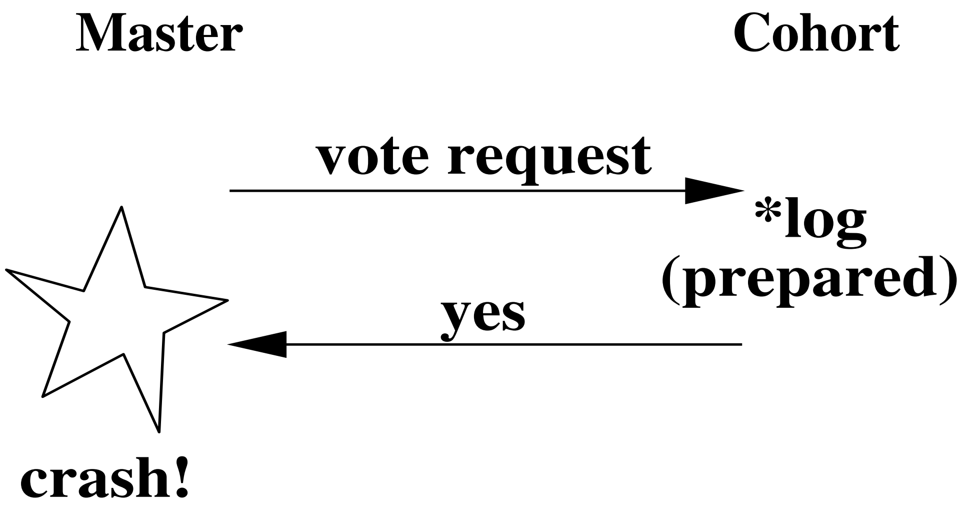
  – No "end" log record

  – Extra log record "cohorts list" is forced

- Abort Processing same as in 2PC
  – with "cohorts list" forced

Failure Blocking in 2PC/PA/PC

- Prone to “blocking” in case of failures



– Master fails after sending vote requests but before sending the decision

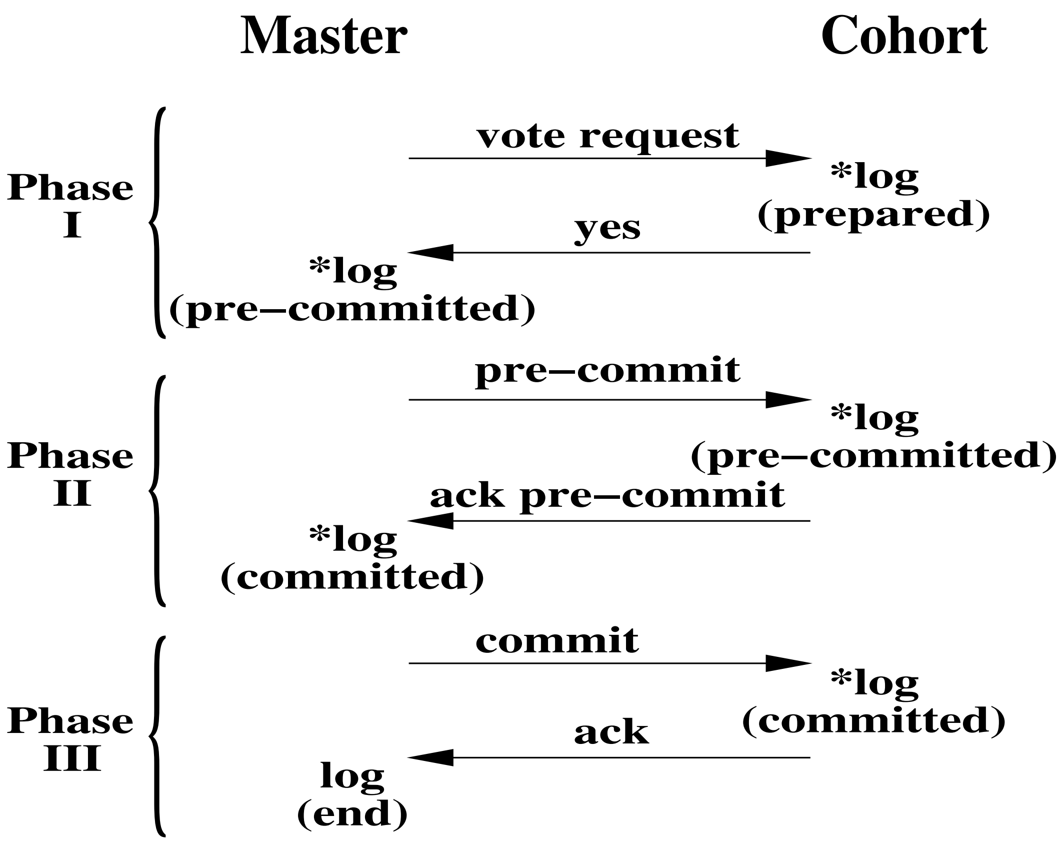
– Prepared cohorts are blocked waiting for the decision from master

– Retain hold on resources

– Unbounded blocking times

3PC : Non-Blocking

- “Write a will and give copies to children”
- Commit Processing :



  – Increased overhead due to extra messages and forced log records

- Abort Processing same as in 2PC

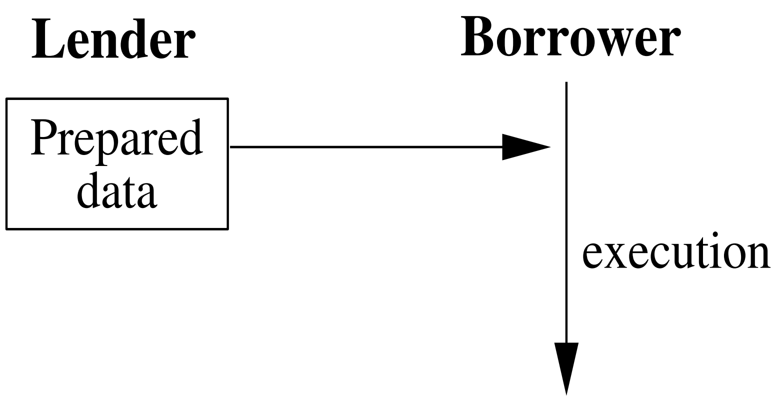
<u>Inadequate for Real-Time DBs</u>

- **Standard protocols are not priority cognizant**
- **Unaware of** *Firm Deadline Semantics*
  - **Even if all votes are** *YES*, **master can commit only if the votes are received <u>before</u> deadline expires**
  - **No communication required for abort processing if deadline expires during transaction execution**
- **"Passive" Aborts (cohorts inform status only on inquiry)**
- **<u>Inherent</u> Priority Inversion**
  - **"prepared" data is inaccessible**
  - **inversion interval unbounded**

Optimistic Commit Protocol (OPT)

- **Reduces Priority Inversion**
  – *controlled* **access to "prepared" data**

- **Low priority "prepared" cohorts**
  lend **data to higher priority transactions**



- **Optimism : Lender will Commit**

OPT Processing

CASE 1:

Lender commits/aborts before Borrower completes execution

– If Lender commits, Borrower continues
– If Lender aborts, Borrower also aborts

CASE 2:

Borrower completes execution before Lender commits/aborts

– Borrower is "put on the shelf" until Lender commits/aborts

- **Borrowers cannot become lenders until borrowing is over**

- **No "Cascading Aborts", abort chain length limited to one**

Features of OPT

- Priority-cognizant processing
- Deadline-cognizant phases
- "Silent Kill" upon deadline expiry
  (Cohorts abort without any communication
  with master)
- Aborts are active
  (Cohorts immediately inform master
  upon non-deadline aborts)
- OPT-PA and OPT-PC
- Non-Blocking OPT (OPT-3PC)

Simulation Model

- **n Sites, m Data Pages uniformly distributed**
- **Shared-memory multiprocessor / multiple disks**
- **High-bandwidth network**
- **Independent Poisson arrival stream of transactions at each site (with rate λ_i at site i)**
- **Distributed transactions (sequential / parallel)**
- *Deadline* = *ArrivalTime* + *SlackFactor* $\times$ *ResourceTime*
- **Firm Deadlines**
- **EDF scheduling of resources**
- **Logging: Only *forced* writes modelled**

Performance Metric : Miss Percent

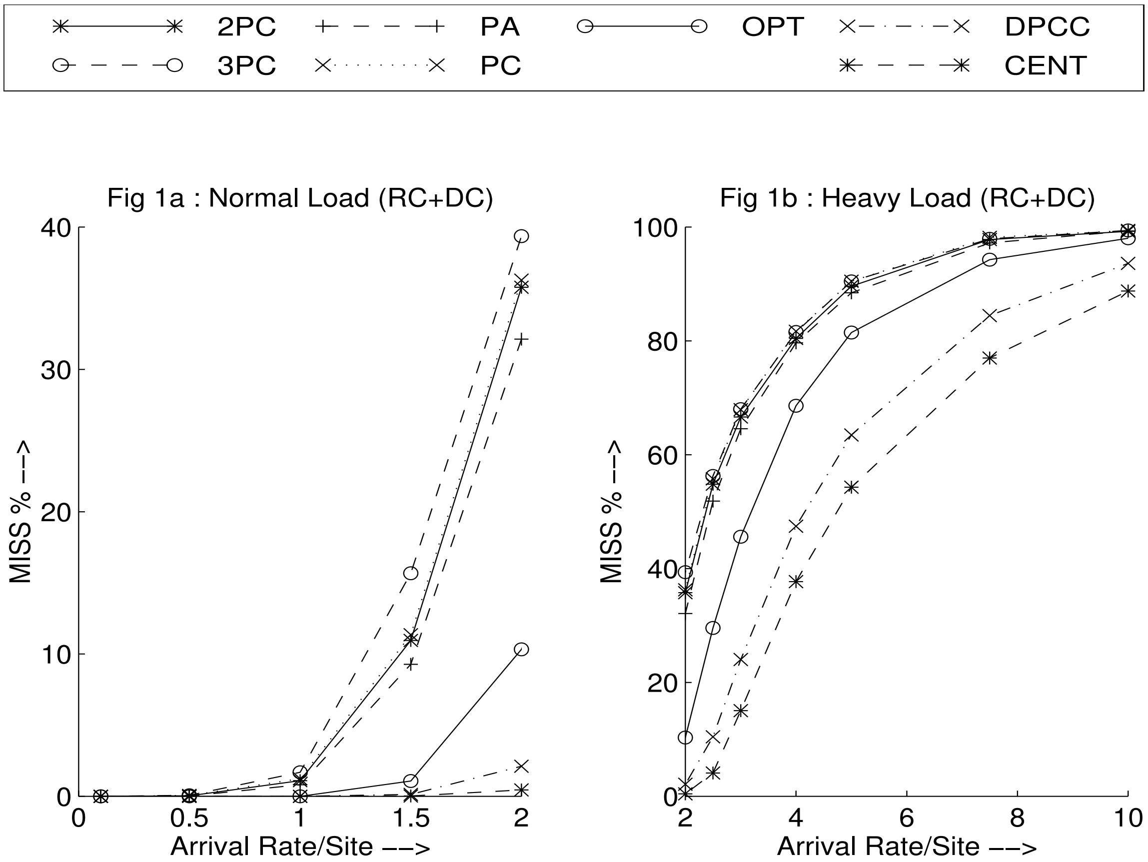
Parameter Values

- 8 **Sites,** 2400 **Data Pages**
- 2 **CPUs and** 4 **Disks per site**
- **Processing time per page :**
  10 *ms* **CPU,** 20 *ms* **Disk**
- **Message communication costs :**
  10 *ms* **at CPU for sending/receiving**
- λ_i **=** λ **for all sites**
- **Transaction executes sequentially at** 3 **sites,**
  accessing 6 **data pages (avg) per site**
  with 0.5 **update probability per page**
- **SlackFactor =** 4
- **Priority assignment is Earliest Deadline**

Baseline Protocols

- **CENT** :
  - Centralized multiprocessor system equivalent to distributed system (equal processors, disks, data)
  - Isolates effect of distribution
- **DPCC** :
  - Artificial system with distributed data processing but centralized commit processing
  - Isolates effect of distributed commit processing

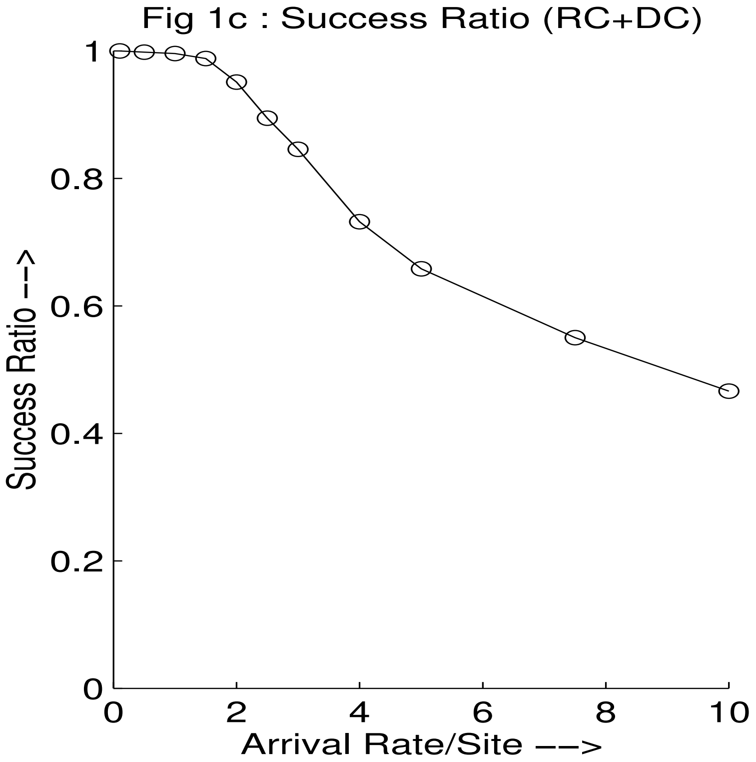
Resource and Data Contention



- **Cost of Commit Processing much more than that of Data Processing**
- **Limited benefits from PA, none from PC**
- **3PC worse than 2PC at normal loads only**
- **OPT significantly reduces commit cost**

Is OPT's optimism well-founded?

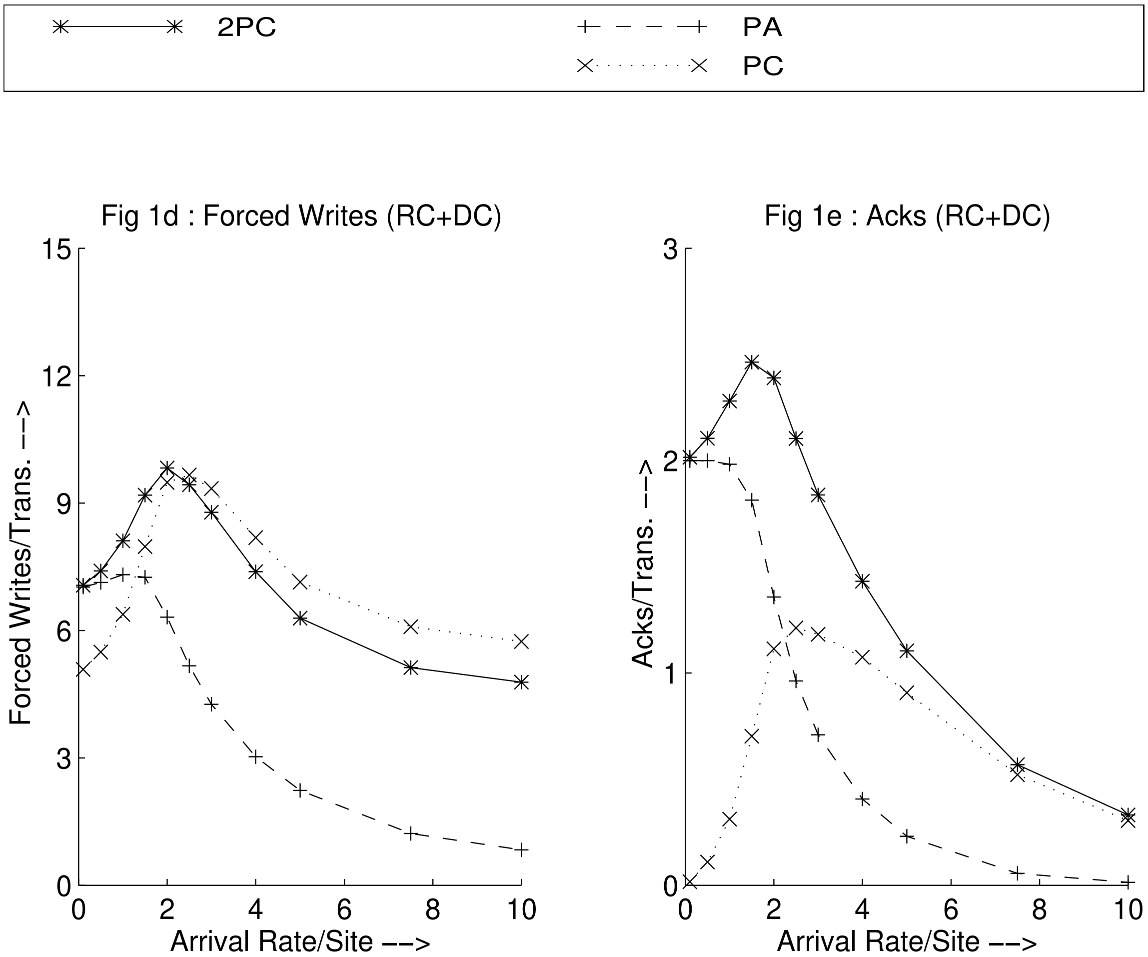
- Fraction of successful borrowings
  $\Rightarrow$ lender commits



- Optimism pays off at normal loads

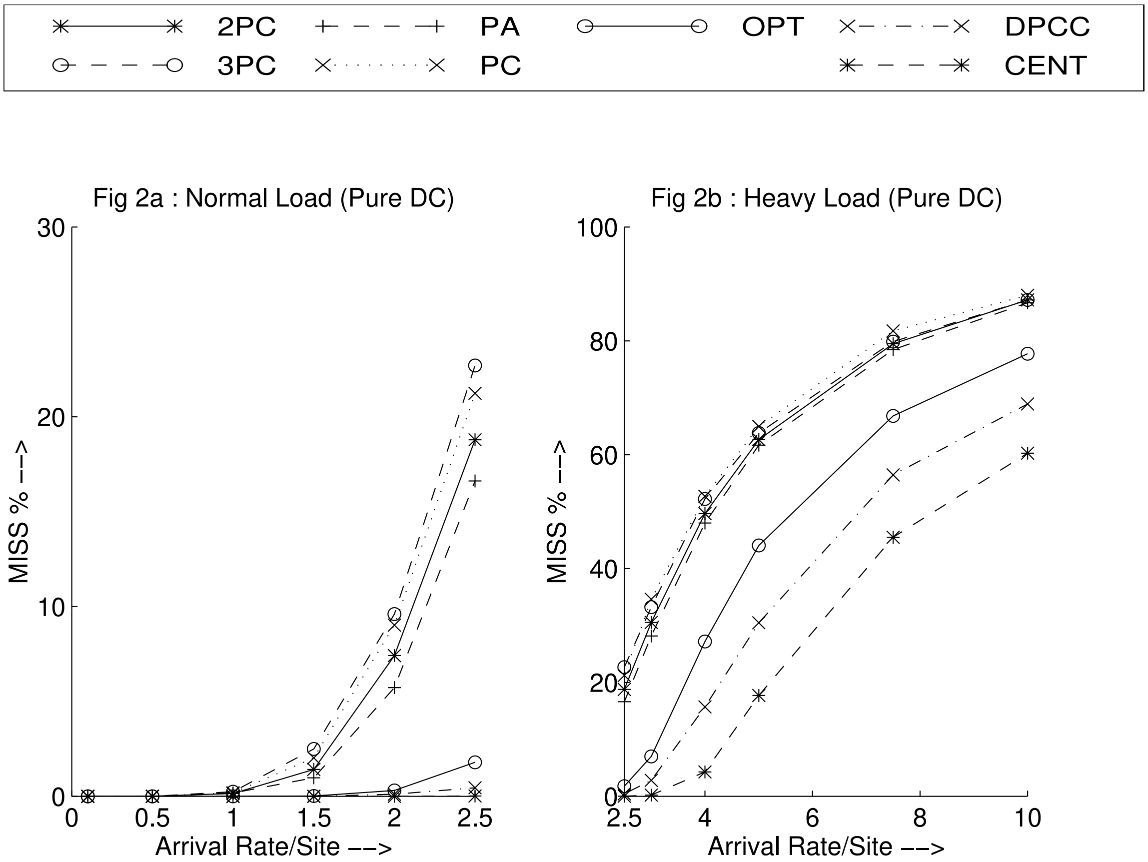
- Judicious borrowing needed at heavy loads
  – due to "unhealthy" lenders

Protocol Overheads



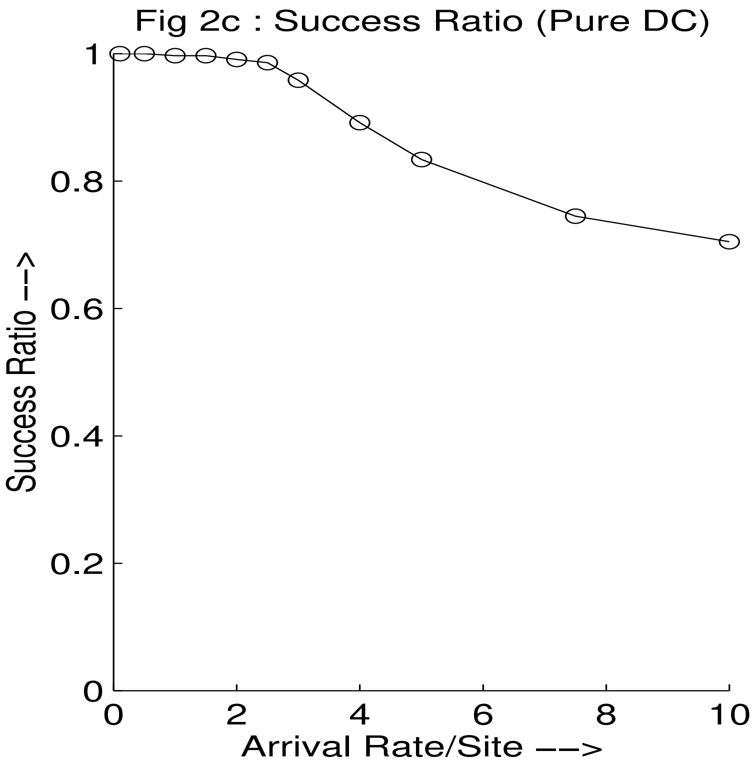
- **PA always better than 2PC**
- **PC better than 2PC only at normal loads**

Pure Data Contention



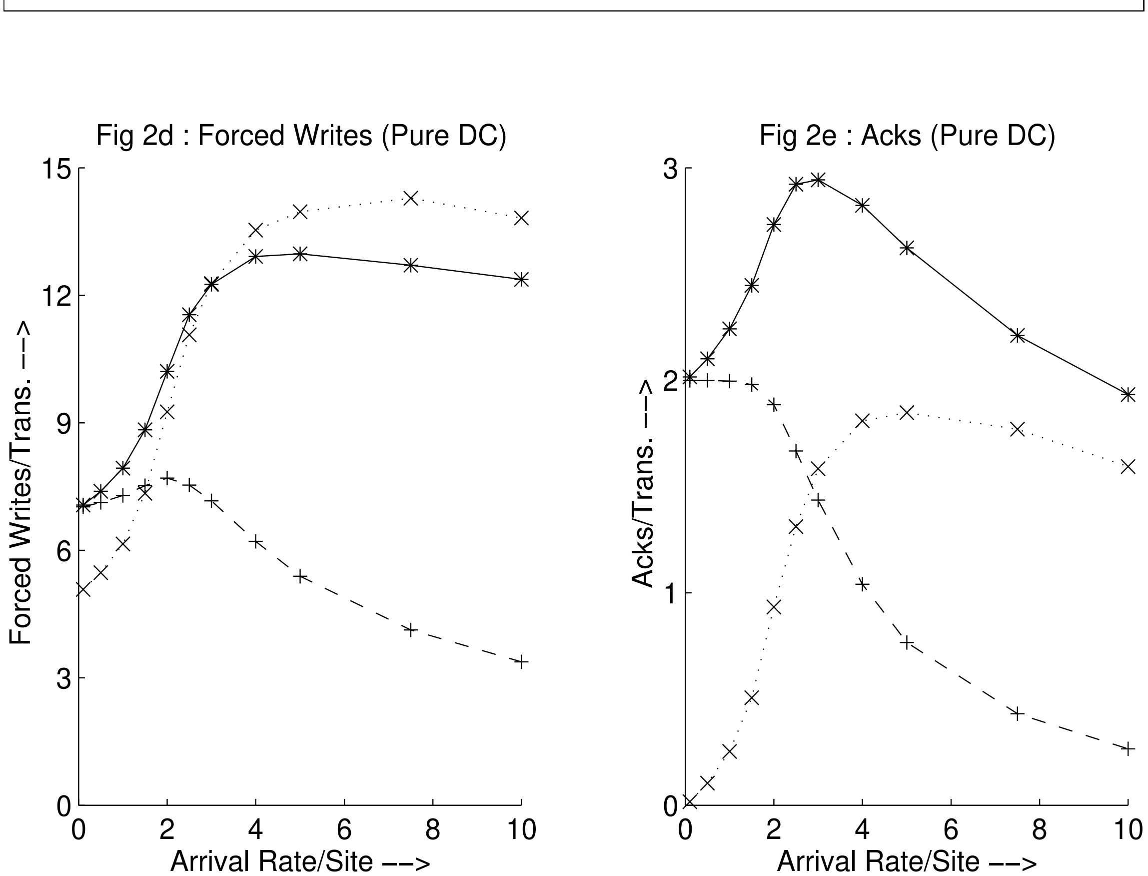
- **No queueing at resources**
- **3PC performs close to 2PC**
- **OPT significantly reduces commit cost**

OPT's Optimism Pays off under Pure DC



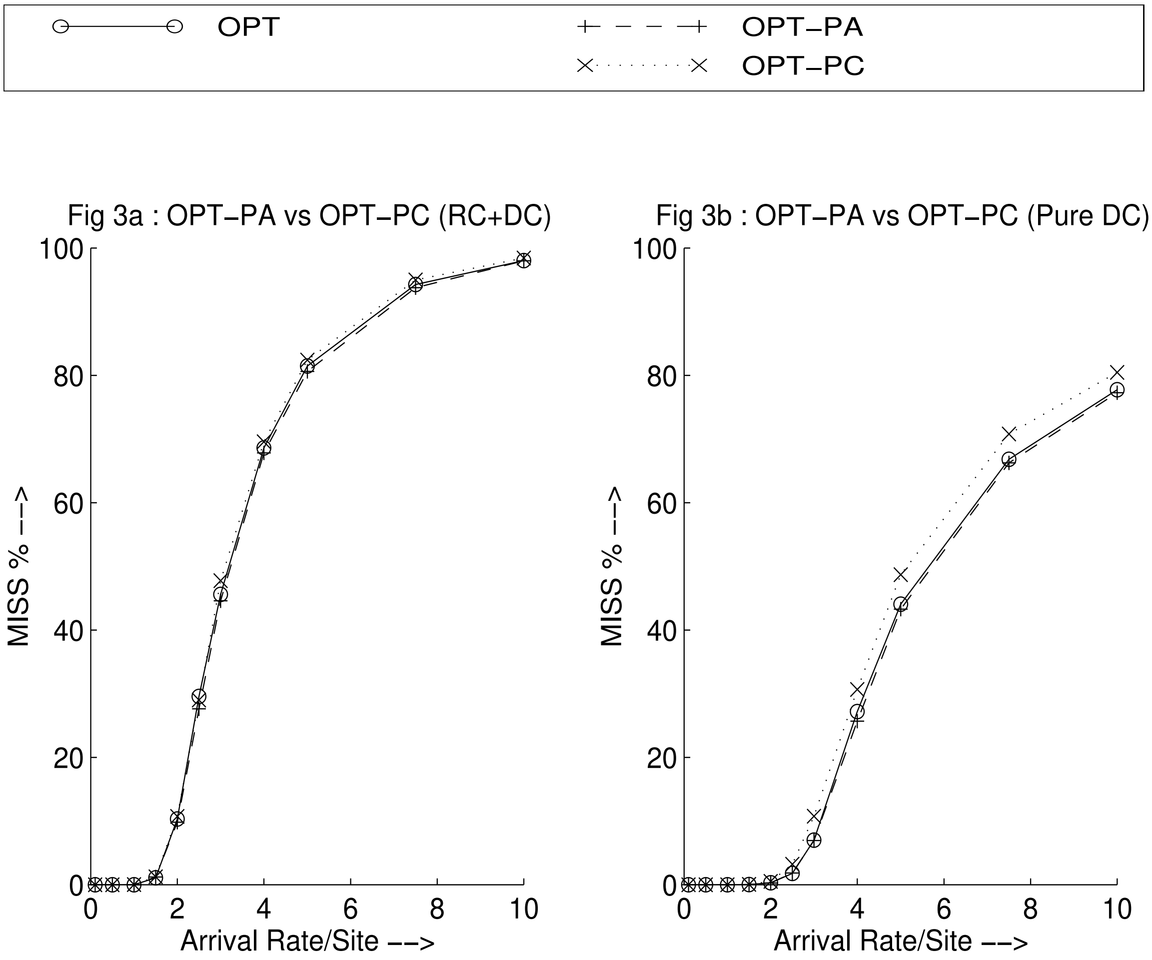
- **High at normal loads**
- **Close to 70 percent even at heavy loads**

Protocol Overheads



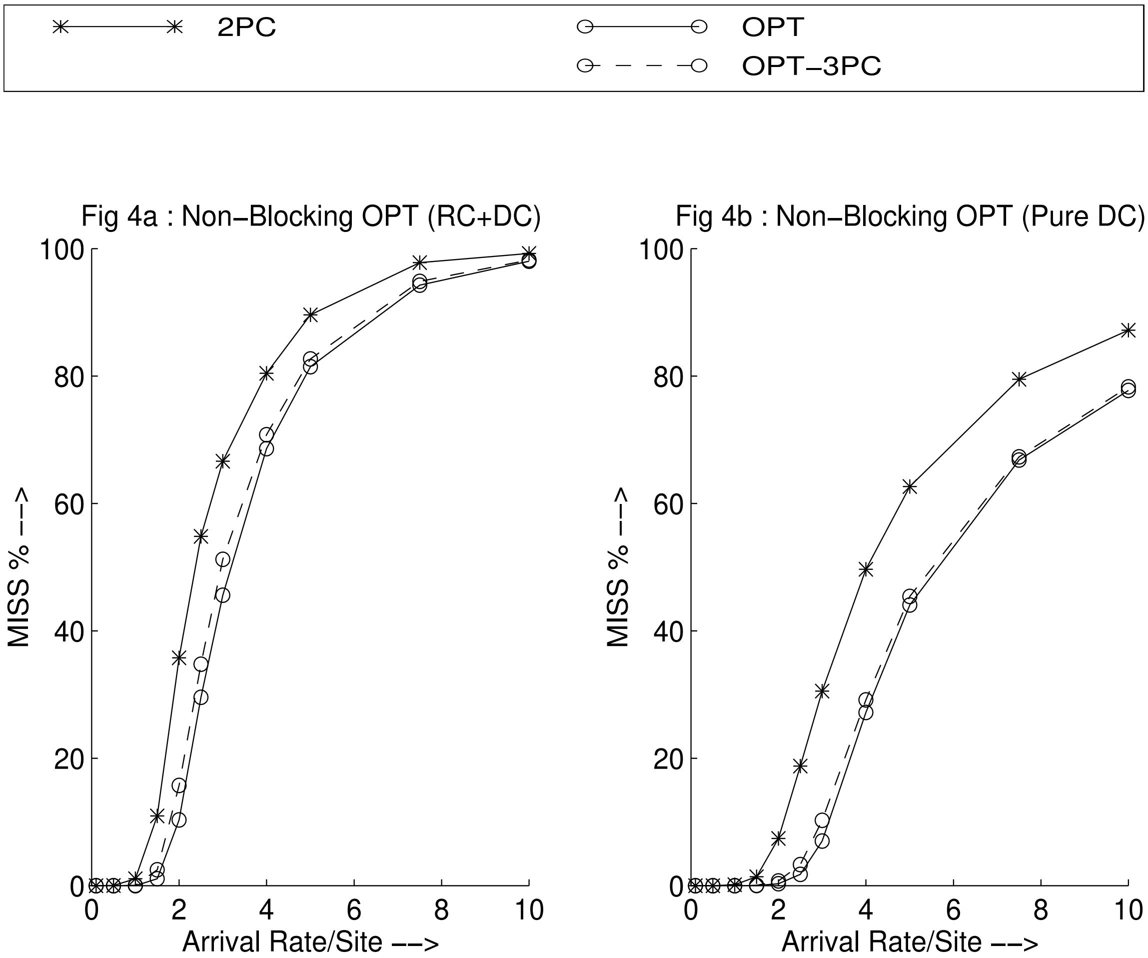
- **Qualitative behavior similar to RC+DC case**

OPT Combinations



- **No tangible benefits from PA or PC**
- **OPT-PC worse than OPT for pure DC**

Non-blocking OPT



- **Non-blocking functionality at modest cost in RC+DC, almost zero cost in pure DC**

- **OPT-3PC performs better than 2PC**

Conclusions

- First quantitative study of distributed real-time atomicity
- Significant overheads of commit processing in distributed RTDBS
- Standard 2PC and 3PC perform poorly in the real-time environment
- PA and PC do not provide tangible benefits in the real-time environment (for update-only transaction workloads)
- The new protocol, OPT, provides significantly better real-time performance
- Non-blocking functionality can be obtained at modest cost in real-time environment

Related Work

- UT-Austin, KAIST
- Unilateral Commit
  - Relaxed notion of Atomicity
  - "Compensation" transactions are required which are application-specific
  - Some actions are not compensatable

Future Work

- *Priority Inheritance* **instead of** *Priority Lending*
- **Heuristics to identify** *healthy* **lenders**
- **Add** *shadow* **transactions to OPT (as in Speculative CC)**
- **Skewed transaction workloads**