# Performance Modeling of Distributed and Replicated Databases

Matthias Nicola[*] Matthias Jarke (Senior Member, IEEE)[*]

mn@matthiasnicola.de jarke@informatik.rwth-aachen.de

Technical University of Aachen, Informatik V (Information Systems)
Ahornstr. 55, 52056 Aachen, Germany

## Abstract

This paper surveys performance models for distributed and replicated database systems. Over the last 20 years a variety of such performance models have been developed and they differ in (1) *which* aspects of a real system are or are not captured in the model (e.g. replication, communication, non-uniform data access, etc.) and (2) *how* these aspects are modeled. We classify the different alternatives and modeling assumptions, and discuss their interdependencies and expressiveness for the representation of distributed databases. This leads to set of building blocks for analytical performance models. To illustrate the work that is surveyed, we select a combination of these proven modeling concepts and give an example how to compose a balanced analytical model of a replicated database. We use this example to show how to derive meaningful performance values and to discuss the applicability and expressiveness of performance models for distributed and replicated databases. Finally, we compare the analytical results to measurements in a distributed database system.

**Index Terms** - performance models, distributed databases, replication, interdatabase communication, modeling assumptions, queueing theory, measurements, benchmarks.

## 1 Introduction

In distributed database systems the data is stored at a number of sites that are geographically distributed over a possibly large region, a country or even the whole world. For many distributed applications like banking, telecommunications, etc. distributed databases represent a more natural and appropriate solution than monolithic, centralized systems. Many of today's commercial database systems such as *Oracle 8* or *IBM DB2 Propagator* provide the required support for data distribution and interdatabase communication. As new communication technologies are emerging, wireless and mobile computing concepts become reality and allow for even higher degrees of „distributedness“ and flexibility in distributed databases. Wireless technology thus expands the scope of distributed computing and enhances distributed applications by enabling ubiquitous database interaction–anywhere and anytime [Imielinski, Badrinath 94].

In this evolving world of distributed databases, data replication plays an increasingly important role. Replication intends to increase data availability in the presence of site or communication failures, and to decrease retrieval costs by local access if possible. The maintenance of replicated data is therefore closely related to intersite communication, and replication management can have significant impact on the overall system performance.

[*] Present Addresses: Matthias Nicola, Informix Software Inc., 485 Alberto Way, Los Gatos, CA 95032, USA.
Matthias Jarke, GMD-FIT, Schloss Birlinghoven, 53754 Sankt Augustin, Germany.

Replication management in distributed database systems concerns the decision when and where to allocate physical copies of logical data fragments (replica placement), and when and how to update them to maintain an acceptable degree of mutual consistency (replica control). The literature offers various algorithms for replica placement [Wolfson et al. 97], [Little, McCue 94], [Acharya, Zdonik 93] as well as replica control [Davidson, Garcia-Molina 85], [Abbott, Garcia-Molina 87], [Ceri et al. 91], [Chen, Pu 92], [Beuter, Dadam 96], [Helal et al. 96].

The vast number of design options in replicated databases requires efficient analytical performance evaluations such that the considerable overhead of simulations or measurements can be focused on a few promising options. A variety of analytical performance models as well as simulation models have been developed in the past to estimate distributed database performance under different network, database and transaction parameters (*dependency analysis*), or to compare two or more algorithms in a given database environment (*comparative analysis*). The performance models vary considerably in the underlying submodels and assumptions which are used to compose an approximation of a real distributed database system. The scale ranges from full replication to no replication, or from infinite transaction processing capacity to sophisticated locking analysis.
The main contribution of this paper is therefore to survey and classify the different modeling alternatives as well as their combinations, interdependencies and expressiveness in performance models of distributed databases. The survey concentrates primarily on analytical models but also classifies a number of simulation studies, because many modeling assumptions and basic modeling concepts (e.g. queueing systems) are independent from the model evaluation methodology.

This paper supplements a set of existing surveys on closely related topics: [Thomasian 98] provides a comprehensive survey on the performance analysis of concurrency control methods, concentrating on centralized database systems. [Agrawal et al. 87] examine different assumptions made in performance models for centralized concurrency control algorithms and study their implications. [Mukkamala 89] surveys *one* aspect of analytical performance models for distributed databases, namely the data distribution submodels, and examines the effects which different modeling alternatives have on the computational model complexity. In [Mukkamala 92] this is extended to analyze the accuracy and complexity of analytical availability estimations.

To illustrate the work that is surveyed and its usability for database designers, we also demonstrate how to develop and deploy an analytical performance model of a replicated database using a selection of proven modeling concepts and assumptions from our classification. Our aim is to compose a performance model that neglects or simplifies less aspects than previous analytical models. This is motivated by three of the main results of the survey:
(1) Considering the great number of alternatives in the design of a replication schema, it becomes apparent that most existing performance models only consider very extreme replication schemata, e.g. no replication or full replication. Furthermore, the important role of intersite communication in replica management is not sufficiently taken into account by many models.
(2) Existing studies typically model a few aspects of a real system quite accurately while the remaining aspects are either neglected or modeled in simplistic manners to keep the model tractable. In particular, usually either the database or the communication part of a model is a reasonable approximation of reality while the other part relies on simplifying and restrictive assumptions. (3) While the evolution and symbiosis of distributed database and modern communication systems is progressing, most performance models lag behind to evaluate the increasing variability in distribution, replication and communication of real world applications in such systems.

Hence, we compose an analytical modeling approach called 2RC (*2*-dimensional *r*eplication model with integrated *c*ommunication) which focuses on the increasingly close interplay between replication and communication. 2RC represents a balanced model of both the database and the communication part. We use this example to show how to derive meaningful performance values (such as response time, throughput, network traffic and scalability), and to illustrate the general applicability and expressiveness of analytical performance models for distributed and replicated databases.

We emphasize that the use of simplifying assumptions is necessary in *every* performance model and often acceptable with respect to the goal of a specific study. By identifying such simplifications in existing studies we do not mean to question the validity of their results rather than to name more expressive alternatives through which results may become more detailed and reliable.

This paper is structured as follows. After a short discussion of general modeling choices in section 2, we survey and classify existing performance models in section 3. This literature review includes a structural dependency analysis of model components, a formal classification of how replication can be modeled, and a novel 2-dimensional model of replication. As an illustrating example, section 4 presents the development of a representative analytical performance model (2RC). In section 5 we discuss a selection of results derived from the model. This demonstrates how analytical models can accurately estimate performance criteria like response time, throughput, network traffic and scalability in order to assist in the evaluation of a variety of common design issues. In section 6 we address the problem of validating performance models for replicated databases. We review existing approaches for measurements in distributed database systems and show how an extension of the DebitCredit benchmark can be used to validate analytical performance estimations through a systematic evaluation of distributed database configurations.

Throughout this paper we refer to Table 1 in which 36 performance studies are evaluated against the criteria discussed in this survey. Note that a cell in the table is left blank in either of two cases: (1) The piece of information is not explicitly given in the study and could not be guessed from the context. (2) The information is not relevant for the study, e.g.: since [Saha et al. 96] compute the number of messages as performance criterion, they do not need to model communication delay.

## 2 Basic Choices

The first decision in a performance modeling project is to decide *what* to evaluate, i.e. to define the envisioned system, configurations or algorithms which are going to be subject of the performance evaluation. Decisions that follow include the selection of performance criteria to consider and the choice between analytical methods or simulations. Then a performance model is constructed for which the alternatives are classified in section 3. In this step virtually all studies make use of the homogeneity assumption - either explicitly or implicitly. Eventually, base values for the model's input parameters must be selected such that a specific system under consideration or a general realistic scenario is represented (depending on the modeling project). In this section we briefly remark on these basic modeling decisions.

### 2.1 Performance Criteria

The most commonly considered performance metrics of distributed databases are *transaction response time* and *transaction throughput*; the former is calculated in more studies than the latter (Table 1). Response time and throughput are perceivable by the database users and hence considered *external* performance criteria. An application oriented external criterion is the number of misrouted phone calls in [Leung 97] where a distributed database for telecommunication networks is evaluated.

A variety of *internal* criteria have been analyzed in specialized studies. [Mukkamala 89] calculates the average number of nodes accessed by a distributed transaction. [Thanos et al. 88] and [Raghuram et al. 92] examine the probability of conflicts between transactions. [Anderson et al. 98], [Ciciani et al. 90] and [Thanos et al. 88] analyze the number and rate of aborted transactions. [Triantafillou, Taylor 95] examine the percentage of restarted transactions with respect to site availability. [Triantafillou 96] evaluates the percentage of stale reads as a performance measure. [Saha et al. 96] and [Alonso et al. 90] calculate the average number of messages per second in the distributed database system. [Triantafillou, Taylor 95] compute the number of messages per read or write operation, [Gray et al. 96] calculate the probability and rate with which transactions wait and deadlocks or reconciliation occur. [Ciciani et al. 90] estimate the processing power in MIPS required at each database node to handle a given workload, and similarly [Barbara, Garcia-Molina 82] investigate the hardware costs of a distributed database systems to fulfill specific response time requirements.

[Mukkamala 92], [Saha et al. 96], [Triantafillou 96] and [Noe, Andressian 87] consider *availability* under certain replica control protocols as a primary performance criterion. These studies define availability as the probability that a read or write operation can access a sufficient number of copies to comply with a specific replica control protocol, e.g. ROWA, quorum consensus, tree quorum protocol, available copies or grid protocol. The approach is then to assume that each database node is available with probability $p$ and to calculate availability for each protocol under consideration. [Shah, Marzullo 89] follow a similar approach to investigate replica availability in partitioned networks. Other authors define availability as the amount of time that data is available for access, or as the ratio of successful transactions over the total number of transactions submitted [Coan et al. 86]. The latter is used in [Shah, Ghosal 90] to examine availability in case of site and link failures. [Mukkamala 92] also investigates the effect of different modeling assumptions on availability evaluations.

## 2.2 Analytical Models vs. Simulations

Performance studies of distributed databases employed (apart from measurements) analytical methods as well as simulations (see Table 1). In a number of cases simulations have been used to validate analytical results, which is indicated in brackets for some studies in Table 1. The major advantage of simulations is that they can evaluate complex system models whose level of detail precludes analytical solutions. However, simulations are costly in terms of programming and computing time. Thus, simulations often fail to cover a comprehensive parameter space and to carry out a sensitivity analysis as thoroughly as desired. [Mukkamala 89] shows that even for simple evaluations of distributed databases the simulation time may be $10^{24}$ times the one for analytical evaluation, so that especially in the early design stages analytical evaluations are to be preferred over simulations. As an example, [Ciciani et al. 90] can not give confidence intervals for their simulation results because the simulation runs took prohibitively long. [Born 96] describes his experience, that the design, implementation and quality assurance of a reliable simulation model for a distributed system will cost at least an order of magnitude more than an analytical model.

Analytical models typically have to employ more restrictive assumptions than simulation models but the performance results can be obtained very efficiently from closed form expressions or numerical iterations, so that extensive parameter variations are feasible.

## 2.3 Parameter Values

Each performance model (whether analytical or simulation) has a set of input parameters for which base values are required in order to derive concrete results. The choice of the base values can significantly influence the quality of the resulting performance values. While some input parameters depend on current technology (e.g. disk service time, CPU speed, etc.) and can be estimated quite accurately, others are application dependent (e.g. number of data items in the database, number of database nodes, number of lock requests per transaction, etc.) and more difficult to determine. Many studies do not use base values gathered from a specific application and a huge variance can be observed among the values found in the literature. For example, the communication delay ranges between 0.5 and 1200 msec, the transaction mix from 100% updates to 99% read operations, and the transaction arrival rate from 0.1 to several thousand transactions per second (TPS). The number of data items (which runs from 4 objects in [Miyanishi et al. 96] to 1.000.000 object in [McDermott, Mukkamala 94]) has to be interpreted with respect to the assumed data granularity, although many studies do not specify whether the „data items“ are records, pages, tables, or files.

As examples, Table 1 includes a survey of values assumed for the number of data items in the distributed database and the number of database sites. Note that there is no strict correspondence between the values chosen and the date of the study.

## 2.4 The Default Homogeneity Assumption

Common to virtually all performance models of distributed databases (analytical and simulations) is what we call the *homogeneity assumption*. Intuitively, the homogeneity assumption says that all database sites and their respective workloads are identical and that all interdatabase activities are symmetrical among all sites, i.e. the distributed system is homogenous. Some of the main modeling assumptions implied by the homogeneity assumption are:

- All database sites have the same structure and the same service capacity.
- All database sites hold the same amount of data.
- In case of replication, all database sites replicate an equal share of their data items and hold an equal amount of replicas.
- All sites receive the same workload, i.e. have an identical transaction arrival rate.
- The data access pattern is identical at each database site.
- Communication between the database sites is symmetrical, i.e. if the sites are numbered $1,...,n$ , then the average number of messages per second from site $i$ to site $j$ equals the average number of messages per second from site $j$ to site $i$ for any pair $(i,j) \in \{1,...,n\} \times \{1,...,n\}$ with $i \neq j$.

Depending on which characteristics of a real system are examined in a performance model, the homogeneity assumption usually extends to further aspects. For instance, if site (or link) failures are considered, each site (communication link) is assumed to fail with equal probability [Noe, Andressian 87], [Saha et al. 96]. We found that the homogeneity assumption is a default assumption, because it is implicitly used unless a study defines certain modeling aspects to be inhomogeneous

| | Basic Choices (Chapter 2) | | | | Model components and assumptions (Chapter 3) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Study** | **Type of study** | **Performance Criteria** | **# Sites** | **# Data Objects** | **Workload** | **Communication** | **Replication** | **Data access** | **Transactions** | **Concurrency Control** |
| [Alonso et al. 90] | analytical (M/G/1) | response time, #messeages | 1 server 150 clients | | Poisson, 1TPS per client, 20 TPS at server | *n* M/M/1 systems | some-obj-to-all-sites | non-uniform (hot-spots) | queries and updates | No |
| [Anderson et al. 98] | simulation | throughput, abort rate, „start-to-commit-time“ & „commit-to-complete-time“ | 20 - 140 | 1000 | Poisson max. 3000 TPS | 155 Mb/sec or 55 Mb/sec 1 incoming and 1 outgoing link per site | full replication | uniform | 90% queries 10% updates 30% write-Ops in updates | yes |
| [Bacelli, Coffmann 83] | analytical (M/M/m) | response time, throughput | 9 - 24 and 1 - 80 | | Poisson, 6 - 14 TPS | no | full replication | uniform | | No, ROWA only |
| [Barbara, Garcia-Molina 82] | simulation | response time, hardware costs | 1-2 | 1000 | 0.2 TPS | Single FIFO queue 1 Mb/sec | full replication | | 50% queries 50% updates | yes |
| [Born 96] | analytical (constant FCFS queues) | response time and throughput for lock requests | 4 | | Poisson 10 - 1000 TPS | constant FCFS server | none | uniform | 30% exclusive 70% shared locks | no conflicts, protocol overhead only |
| [Bouras, Spirakis 96] | analytical (simulation) | mean waiting time, probability of waiting, etc. | | 250 per site | Poisson 2 or 8 TPS per site | exponentially distrib. delay, i.e. M/M/∞ | none | uniform | | yes (Timestamp ordering) |
| [Burger et al. 97] | simulation | throughput | 4 | 1600 (16 files of 100 pages) | Closed model | constant 10 msec delay | partial: 2 copies per file | | 0 - 100% write-ops in txns | yes (Multiversion) |
| [Cheung et al. 92] | analytical | availability | 4-32 | | Poisson: | neglected | full replication | uniform | | |
| | simulation | response time | 4-32 | 10.000 | 0 - 1 TPS | neglected | full replication | uniform | 20% write ops | Yes, 2PL |
| [Ciciani et al. 90] | analytical & simulation (M/M/1) | response Time, abort probability, MIPS req. Per site | 20 | 100.000 | Poisson 3 TPS per site | constant 0.2 sec delay | all-obj-to-some-sites | uniform | 2 txn classes | yes (various protocols) |
| [Gallersdörfer,Nicola 95] | analytical (M/$H_2$/1) | response time, throughput, #messages | 50 | 3000 primary copies | Poisson, 100 TPS | constant, 2 msg types: 100 ms , 400 ms | some-obj-to all-sites | non-uniform (locality model) | 90% queries, 10 updates | no |
| [Gray et al. 96] | analytical | probability & rate for waits, deadlocks & reconciliation | | | | ignored | full replication | uniform | no reads | |
| [Hung, Lam 92] | simulation, (M/M/1) | response time, throughput | 3 | 5000, 1000 | Poisson | 3 msg types with 1200 ms, 600 ms, 300 ms const. delay | full replication | uniform | 2 txn types, long and short | yes, exclusive locks only |
| [Hwang et al. 96] | analytical | response time | 10 | | 10 to 150 TPS per site | constant 0.01 sec delay | no replication and all-obj-to-some-sites | non-uniform (hot-spot access) | 80% read only txns | No |
| [Jenq et al. 88] | analytical (MVA) | CPU utilization, disk IO rate, etc. | 2 | | | neglected | no replication | uniform | queries & updates | yes |
| [Kemme, Alonso 98] | simulation (open model) | response time, (abort rate) | 10 | 10.000 | interarrival times of 30ms - 300ms | constant 2 - 100 ms | full replication | uniform | 2 txn types: short & long | yes |
| [Keum et al. 95] | simulation | response Time | 3-15 | 10.000 | Poisson | msg queue and send queue | full replication | uniform | queries and updates | yes, blocked queue |

*Table 1: Performance Models for Distributed Databases - Survey*

|  | Basic Choices (Chapter 2) |  |  |  | Model components and assumptions (Chapter 3) |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| **Study** | **Type of study** | **Performance Criteria** | **# Sites** | **# Data Objects** | **Workload** | **Communication** | **Replication** | **Data access** | **Transactions** | **Concurrency Control** |
| [Kuang, Mukkamala 91] | comb.of sim. and analysis | response time |  | 200 (1000) | Poisson, 1-5 TPS | exponentially distrib. delay | all-obj-to-some-sites with 2 - 3 (7) replicas | uniform | r% read ops in txns, $0\leq r \leq 100$ | yes |
| [Leung 97] | analytical ($M/H_2/1$) | #misrouted phone calls | 2 or 5 |  | adjusted so that $\rho$=90%, | constant 10 ms | full replication | uniform | 50% to 100% queries | no |
| [Liang, Tripathi 96] | simulation, analytical | system power, response time | LAN | 500 | long lived txns | constant | no replication | uniform | long lived txns | Aborts & Rollbacks |
| [McDermott, Mukkamala 94] | analytical (M/M/1) | throughput | 4 | 1.000.000 | Poisson | constant | full replication | uniform | update or query biased | yes |
| [Miyanishi et al. 96] | analytical | response time | 3 | 4 | Poisson | neglected | „per-object“ reduced to full replication |  | updates only | yes, exclusive locks only |
| [Nelson, Iyer 85] | analytical | throughput, response time | fixed (3 , 5) or varied |  | Poisson max. 50 TPS | no | full replication | uniform | 99%, 90% or 75% read ops. | No, ROWA only |
| [Noe, Andressian 87] | simulation | availability | 10-30 | 40 per site | 50 - 400 requests per day |  | all-obj-to-some-sites, with 2 or 3 replicas |  | 30%-70% reads |  |
| [Raghuram et al. 92] | analytical (MVA) | response time, conflict probability, throughput | 3 or 5 | 1000 | Poisson 0.1, 0.2 or 0.6 TPS | n-1 queues per site (1 queue per outgoing link) | no replication | non-uniform: locality model *„b-l access“* |  | yes, only exclusive locks |
| [Ren et al. 96] | analytical | response time, queue length distrib |  |  | Poisson | M/G/∞ server | no replication | non-uniform | variable | multiversion timestamping |
| [Saha et al. 96] | analytical | #avg. case msgs, availability | 32-1024 |  |  |  | full replication |  |  | no |
| [Shah, Ghosal 90] | analytical (MVA) | availability, response time |  |  | Poisson | constant | partial replication | uniform | queries & updates | yes |
| [Sheth et al. 85] | analytical (M/M/1) | response time, utilization | 12 | 1000 | Poisson, 0 - 0.15 TPS | 1 M/M/1 queue per link, 3 msg types | full replication | uniform | 100% updates | yes |
| [Shyu, Li 90] | Analytical (simulation) | response time (based on constant txn service times) |  | 200, 400 | Poisson, 0 - 8 TPS | exponentially distrib. delay, i.e. M/M/∞ | no replication | uniform | writes and exclusive locks only | 1 M/M/1 lock request queue per object |
| [Son, Koulombis 91] | simulation | response time, throughput, #aborts | 3 - 10 | 1000 | Poisson, | constant, 0.5 - 10 msec | full replication | uniform | 10% to 90% queries |  |
| [Tai, Meyer 96] | analytical M/M/1, M/G/1 | probability of missed deadline | 3 |  | Poisson, 10 - 200 TPS | exponentially distrib. delay | full replication |  |  | yes |
| [Thanos et al. 88] | simulation | response time, % aborted TAs, conflict probability | 3 (5) | 1000 | Poisson, 10 TPS | constant, 0.1 sec | no replication, full replication, and 3 copies over 5 sites | locality model | updates & queries | yes, 2PL |
| [Triantafillou, Taylor 95] | simulation | %restarted TAs transaction latency | 27 (3 LANs of 9 sites) | 100.000 per LAN | Poisson, 90TPS per LAN | constant | all-obj-to-some-sites 5 replicas per obj. | non-uniform | 25% reads 75% writes | yes, strict 2PL |
| [Triantafillou 96] | simulation | % stale reads | 27 | 300.000, 3000 are hot spots | Poisson: 500, 1000, 1500 TPS | no | all-obj-to-some-sites 3 or 5 replicas per obj. | hot spot 0.8-0.01 access | 80% read ops in txns |  |
|  | analytical | availability | 5 - 20 | 100.000 per site | 100TPS per site | no | all-obj-to-some-sites |  | 90% reads |  |
| [Ulusoy 94] | simulation | % satisfied deadlines | 5 | 200 per site | Poisson, 2 -3 TPS | constant, 5 ms | all-obj-to-some-sites | uniform vs. locality model | 25% or 75% updates | yes |
| [Wu 93] | simulation | response time | 16 | 10 | Poisson | constant 0.5 sec | full replication | non-uniform: 0.8-0.2 access | ratio of read to write ops: 5/1 | yes, 2PL |

*Table 1(cont'd): Performance Models for Distributed Databases - Survey*

# 3 Classification of existing Performance Models

In this section we analyze alternatives in performance modeling of distributed databases in various perspectives: (1) the general concepts to model database nodes, (2) the options in considering interdatabase communication, (3) the submodels to account for replication, (4) the assumptions concerning data access patterns, (5) the transaction processing models and finally (6) the interdependencies between all these aspects which are (or are not) captured in existing models. This analysis reveals drawbacks in existing performance evaluations of distributed databases, and the classification of replication models leads to the definition of a new 2-dimensional model of replication in section 3.3. The mapping between the literature and some of the classified modeling concepts is given in the right half of Table 1.

## 3.1 Database Site Models

Simulation and analytical performance studies of distributed databases commonly use queueing systems as the underlying models. For details on queueing theory and Kendall's classifying notation of queueing systems (e.g. M/M/1) see [Kleinrock 75], [Jain 91], [Gross, Harris 85].

Some of the earliest queueing models of distributed databases can be found in [Coffmann et al. 81], [Bacelli, Coffmann 83] and [Nelson, Iyer 85]. These studies model a fully replicated database of *m* local sites by a M/M/m/FCFS queueing system. This means that transactions which arrive according to a Poisson process are served on a first-come-first-serve basis by *m* servers and require an exponentially distributed service time. The read transactions are processed by the *m* servers in parallel, while write transactions occupy all *m* servers during their service time. This models shared read and exclusive write operations. In [Bacelli, Coffmann 83], writes have preemptive priority over read operations. This is modeled as an M/M/m system with preemptive service interruptions, where the interruptions correspond to the service periods of an M/M/1 system which represents the arrival and service of updates. [Nelson, Iyer 85] assume non-preemptive processing of write operations and compare parallel updating with sequential updating of replicas. Major drawbacks of these early models are that intersite communication is neglected, that all sites share a single queue of incoming transactions, and that the extreme case of full replication is assumed.

To remedy some of these flaws, distributed databases can be modeled by *queueing networks* [Kleinrock 75]. Open queueing networks allow a varying number of jobs in the system, i.e. the arrival rate does not depend on the number of jobs already in the system. Open networks have been used in many studies [Mc Dermott, Mukkamala 94], [Mariasoosai, Singhal 90], [Jenq et al. 89], [Singhal 86], [Garcia-Molina 82], [Barbara, Garcia-Molina 82]. Closed queueing networks consider a fixed number of jobs in the system, i.e. a completed job is immediately replaced by a new job. Such networks have been deployed in [Carey, Livny 88,96], [Liang, Tripathi 96]. Open queueing networks are more realistic than closed networks because the number of transactions in a database system is typically not constant. However, in certain cases closed models allow for easier solutions. For example, the performance of concurrency control methods depends on the multiprogramming level and is therefore easier to estimate in a closed rather than an open model which is due to the variability of the number of concurrent transactions in an open system.

[Ciciani et al. 90, 92], [Hung, Lam 92], [McDermott, Mukkamala 94] use networks of M/M/1 queues so that each local database is modeled as an M/M/1 system. However, this still restricts all transactions to have the same exponentially distributed service times. More general, [Banerjee et al. 94], [Hwang et al. 96] model the local databases as M/G/1/FCFS and M/G/1/RR queues respectively, with generally distributed service times. [Gallersdörfer, Nicola 95], [Leung 97] use networks of M/$H_2$/1 systems with 2-phase hyper-exponentially distributed service times to assign different exponentially distributed service times to read-only transactions (queries) and updates. Still, such models do not allow to evaluate real-world systems with more than two transaction types.

For more details on the database sites, each site can itself be modeled as a queueing network: [Garcia-Molina 82], [Sheth et al. 85], [Singhal 86], [Cai 87], [Jenq et al. 88] analyze local databases as feedback networks of an I/O-queue and a CPU-queue.

All queueing models of distributed databases consider unlimited waiting rooms, i.e. there is no restriction on the queue lengths. Additionally, queues representing database nodes are usually defined to serve their jobs on a first-come-first-serve basis, except for [Hwang et al. 96] where the round robin queue discipline is used.

Common to most models is that transactions are assumed to arrive as a Poisson process, i.e. the interarrival time is exponentially distributed. The widely accepted (and thus usually omitted) justification is that Poisson streams have been found to be a good approximation for the arrival of jobs submitted independently by a large number of users [Kleinrock 75],[Gross, Harris 85], [Gray 91].

Most studies also assume that the transaction service times are exponentially distributed, but a justification for this assumption is usually missing. A possible justification is that the time required to process a transaction at a database site is mainly determined by the disk service time, which in turn is closely related to the number of data objects referenced [Son, Haghighi 90]. Transactions that access a small or moderate number of data items are expected to occur more frequently than transactions that reference a large number of data objects. This can be expressed if the number of data objects accessed per transaction is assumed to be geometrically distributed. The service time for a transaction can then be assumed to be exponentially distributed, which is the „continuous version“ of the discrete geometric distribution.

Very few studies assume constant rather than exponentially or generally distributed service times [Shyu, Li 90], [Born 96]. The simulation model in [Shyu, Li 90] does not assume queueing of transactions but assigns the same constant service time to every transaction. Instead, the read and write operations in each transaction have to queue individually for lock requests in M/M/1 systems (see section 3.5.2). [Born 96] analyzes the distributed database at the level of lock requests and models each site as an M/D/1-FCFS server, claiming that lock requests require constant rather than exponentially distributed service times.

Unlike most other studies, [Bouras, Spirakis 96] and [Ren et al. 96] assume that the local transaction processing time is negligible compared to communication delays. On the other extreme, [Cheung et al. 92] and [Miyanishi et al. 96] assume that network delay is negligible compared to the database service times.

## 3.2 Communication Models

Most distributed database performance studies assume that the communication network has an unlimited transmission capacity and that the transmission time is constant (e.g. [Garcia-Molina 82], [Singhal 86], [Mariasoosai, Singhal 90], [Ulusoy, Belford 92]). The queueing theoretical background of this assumption is that the network is implicitly modeled as an M/D/∞ system which is an infinite server (sometimes called delay center) that introduces a constant delay for each message, regardless of message size or network load [Jain 91]. "Infinite" means unlimited transmission capacity, no queueing of messages, and the network is never considered to be the bottleneck. For simplicity, these details are usually omitted:

Some models relax the restriction of constant transmission delay but still presume unlimited network capacity. [Shyu, Li 90] and [Kuang, Mukkamala 91] consider exponentially distributed communication delay by modeling the network as an M/M/∞ server, [Ren et al. 96] use an M/G/∞ system to model arbitrarily distributed network delay. These studies only compute the response time as a performance metric. Such models would predict that replication always deteriorates throughput but never increases it (which we will disprove in section 5): due to the infinite service capacity, situations in which the network starts getting congested cannot be captured.

However, many large wide area applications and wireless and mobile information systems suffer from low bandwidth. There, the communication links may indeed become a bottleneck, especially when a large amount of replicas is to be maintained. Unfortunately, very few attempts have been made to combine a detailed analytical database model with an analytical model of limited network capacity:

In parts of their study, [Alonso et al. 90] model limited communication capacity and exponentially distributed communication delay by a set of M/M/1 queues which stand for concentrators representing the network. [Alonso et al. 90] use this network model to evaluate caching concepts between client workstations and a centralized database server but it could also be used for a distributed database system. A quite detailed model of interdatabase communication is presented in [Sheth et al. 85] but has not received much attention in subsequent analytical models of distributed databases. The authors assume that *k* outgoing transmission channels are attached to each site. Each transmission channel is modeled by an M/M/1 system and processes an equal share of the outgoing messages at a site. Unfortunately, the database part of the model in [Sheth et al. 85] contains simplifying assumptions like full replication, uniform data access, and a workload of 100% updates (i.e. no read-only transactions).

The capability of simulations to evaluate more complex system models have rarely been exploited to capture interdatabase communication details. In [Anderson et al. 98] the simulated network has a star topology with an ATM switch at its center and each database site has an incoming and an outgoing link with the switch. The simulation model in [Keum et al. 95] contains a message queue and a send queue at each database node. All incoming and outgoing messages join the message queue to receive CPU service. The outgoing messages then enter the send queue to receive network service. Again, both studies are restrictive in the database part of their model, e.g. they assume full replication and uniform data access. The simulation in [Carey, Livny 96] models the database nodes as a detailed queueing network where messages require CPU service. However, they assumes a local-area network where the actual message transmission time is negligible.

Studies like [Anderson et al. 98], [Keum et al. 95], [Sheth et al. 85] on the one hand and [Carey, Livny 96], [Gallersdörfer, Nicola 95], [Ciciani et al. 90] on the other show that in many existing studies either the database or the communication part of a model is an accurate approximation of reality while the other part relies on simplifying and restrictive assumptions. In section 4 we demonstrate how to develop and evaluate a balanced model of both parts.

## 3.3 Replication Models

Many performance studies of distributed databases simply assume *no replication*, i.e. each logical data item is represented by exactly one physical copy (e.g. [Dias et al. 87], [Son, Haghighi 90]). Models which assume partial replication rather than full replication either consider the fraction of replicated data (*how many objects are replicated?*) or the degree of replication (*to how many sites are objects replicated?*), but not both. This distinction leads to the following classification and the development of a new 2-dimensional replication model.

(1) All objects to all sites (full replication)

Most performance evaluations assume full replication (e.g. [Coffmann et al. 81], [Garcia-Molina 82], [Singhal 86], [Mariasoosai, Singhal 90], [Kumar, Segev 93], [Son, Zhang 95]), i.e. all data objects are replicated to all sites so that each site holds a complete copy of the distributed database. This is an extreme case of replication and it has been recognized that for many applications neither full nor no replication is the optimal configuration [Ciciani et al. 90], [Gallersdörfer, Nicola 95], [Alonso 97]. Some authors argue that full replication is an acceptable assumption for worst-case considerations [Anderson et al. 98].

(2) All objects to some sites (1-dimensional partial replication)

Several studies model partial replication in the way that each data object is replicated to some of the sites (e.g. [Carey, Livny 88,96], [Mukkamala 87], [Ciciani et al. 90,92]). Formally, the degree of replication can be denoted by a parameter $r \in \in \{1,2,...,n\}$, describing that each logical data item is represented by $r$ physical copies, where $n$ is the number of sites. A value of $r = 1$ expresses no replication, $r = n$ means full replication, and if $r > 1$, every data item is replicated. Consequently, either no or all data items are replicated. It is usually assumed that the replicas are distributed evenly across the sites, but it is still undefined *which* copies are placed on *which* sites, such that different degrees of quality of a replication schema can be modeled.

Data which is updated frequently should not be replicated to avoid update propagation overhead. However, data which is updated rarely but read frequently should be replicated to increase local availability and avoid communication delays. This common situation requires to select appropriate data items for replication, which cannot be modeled with the all-objects-to-some-sites scheme.

(3) Some objects to all sites (1-dimensional partial replication)

Alternatively, the degree of replication $r$ can be defined as $r \in [0;1]$ describing the fraction of logical data items that are fully replicated to all sites. A data item is either fully replicated or not replicated at all. A value of $r = 0$ expresses no replication, $r = 1$ means full replication.

To the best of our knowledge, this model of partial replication has only been considered in [Alonso et al. 90], [Gallersdörfer, Nicola 95] and [Alonso 97]. [Gallersdörfer, Nicola 95] used it for performance evaluation of relaxed coherency in partially replicated databases. [Alonso et al. 90] modeled a client server information system and assumed that 0% to 20% of the server data is cached at each workstation. This is comparable to the some-objects-to-all-sites replication scheme with $r \in [0; 0.2]$.

[Alonso 97] considered the some-objects-to-all-sites model to examine the correctness of a replication protocol based on group communication.

The some-objects-to-all-sites scheme is orthogonal to the all-objects-to-some-sites approach in the sense that the degree of replication is defined along the fraction of replicated data items as opposed to the number of copies.

For $r < 1$, the selection of data items to replicate is undefined. This *undefined replica selection* can be used to model the quality of replication.

Full replication of some data items and no replication of others is a choice between two extremes and entails considerable update propagation overhead for the former and a severely reduced availability of the latter group of items. Since this situation is not typical in real-world applications, the some-objects-to-all-sites scheme is again a questionable modeling approach.

<u>(4) Some objects to some sites (2-dimensional partial replication)</u>

Based on the classification above, we propose a new 2-dimensional replication model called "*Some objects to some sites*". This scheme integrates the two orthogonal 1-dimensional concepts and has not yet been used in existing performance evaluations of replicated databases. In the 2D-model, replication is modeled by a pair $(r_1,r_2) \in [0;1] \times \{2,...,n\}$ such that $r_1 \in [0;1]$ describes the fraction of logical data items which are represented by $r_2$ physical copies each, i.e. they are replicated to $r_2$ of the $n$ sites. A share of $1 - r_1$ logical data items remain unreplicated, i.e. are represented by only one physical copy. No replication is expressed by $r_1 = 0$. Full replication is modeled by ($r_1 = 1$, $r_2 = n$). The 2D-model does not define the selection of data items to replicate nor their placement. This property of *undefined replica selection and placement* can be exploited to model the quality of replication.

For $d$ logical data items, a replication schema $(r_1,r_2)$ increases the number of physical copies from $d$ (no replication) to $(r_1 \cdot d \cdot r_2) + (d \cdot (1 - r_1))$. Viewing the number of copies of replicated objects $(r_1 \cdot d \cdot r_2)$ as the actual extent of replication, we express it (for visualization and calculations) independently from $d$ and normalized to the interval [0;1] as an overall level of replication. This is achieved through dividing by $d \cdot n$,

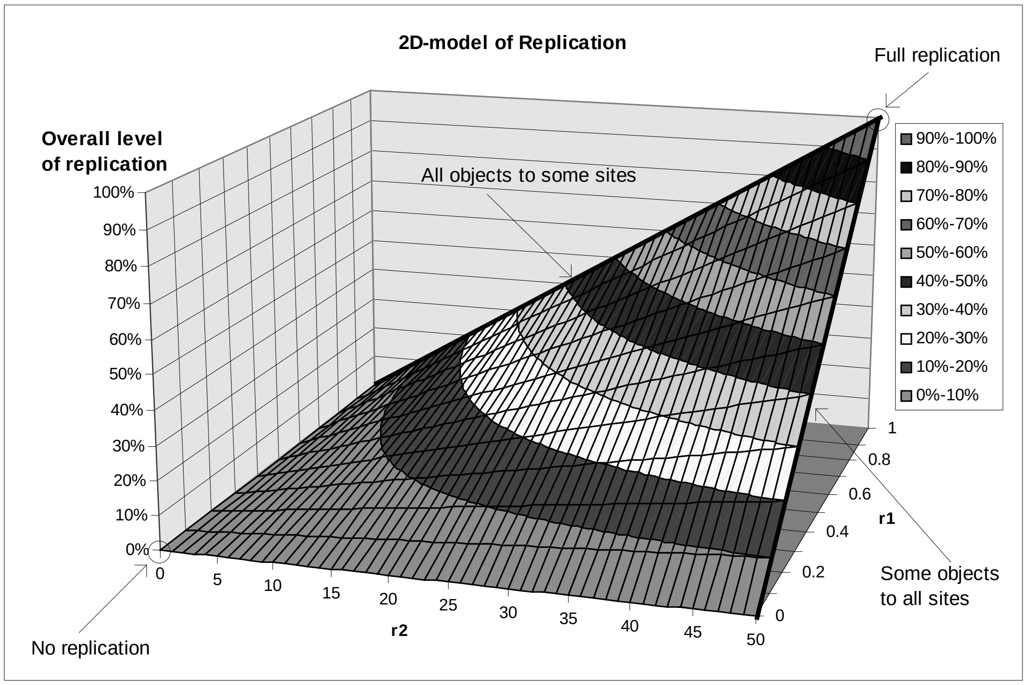

*Figure 1: The 2-dimensional model of replication*

yielding $(r_1 \cdot r_2)/n$. This characteristic of the 2D-approach is depicted in Figure 1 for $n = 50$ sites. Performance models which assume full replication only consider the point $(1,n)$ as the possible replication schema. The all-objects-to-some-sites scheme analyses replication along the bold line from point (1,50) to (1,0) only. The orthogonal some-objects-to-all-sites scheme studies replication along the line from (0,50) to (1,50) only. Using the 2D-scheme any point in Figure 1 can be considered a possible replication schema.

In large wide area distributed databases it is hardly affordable to replicate some data items to all sites (causing high update propagation overhead) and others to none (reducing their availability). Thus, the some-objects-to-all-sites scheme is not realistic. Furthermore, in many applications there is update-intensive data which should be replicated to very few sites while read intensive data should be replicated to many sites. This cannot be modeled with the all-objects-to-some-sites scheme. The 2D-approach can capture such scenarios and models realistic replication schemata more accurately than previous studies. Thus, we believe that integrating the 2D-scheme in an analytical queueing model is a profitable contribution towards a better understanding of how replication affects distributed system performance.

(5) Replication-per-object models

Although the 2D-model is significantly more expressive than previous 1-dimensional schemes, it is still restrictive in the assumption that the degree of replication is the same for all replicated data items. At the expense of a considerably higher model complexity [Mukkamala 89], this can be overcome if the degree of replication is treated as a parameter on a per object (or object class) basis:

(5.1) For each of the *d* logical data objects (or objects classes), which are assumed to be numbered 1,2,...,*d* , the number of copies could be defined individually. The core of this replication model is a function $r$: $\{1,2,...,d\} \to \{1,.....,n\}$ such that the value of $r(i)$ is the number of copies of data item *i*. Although the number of copies are specified individually for each logical data item, the placement of the replicas or their distribution over the *n* sites is remains unspecified.
To the best of our knowledge, [Ulusoy, Belford 92] is the only study that defines this replication model. However, they circumvent the complexity of the model by assuming that for all logical data items *i*, the value $r(i)$ is uniformly distributed between 1 and *n*. Hence, on average each data item is represented by $(n + 1)/2$ copies. Since the calculation of the response time is based upon this average, the complex replication model is actually reduced to the 1-dimensional *all-objects-to-some-sites* model. [Ulusoy, Belford 92] assume that the $d \cdot (n + 1)/2$ physical copies are uniformly distributed over the *n* sites to allow for the homogeneity assumption.

(5.2) As an extension of (5.1), not only the number of copies but also their allocation at particular sites can be specified individually for each logical data item. For *d* logical data items and *n* sites, the function of the replication model is defined as $r$: $\{1,.....,n\} \times \{1,2,...,d\} \to \{0;1\}$ such that $r(i,j) = 1$ if site *i* holds a replica of data item *j*, and $r(i,j) = 0$ if site *i* does not hold a replica of data item *j*. Such a model definition can be found in [Miyanishi et al. 95] and [Carey, Livny 88,96]. However, both studies only *define* the detailed model of replication but do not use it for the analytical calculations and simulation experiments. In [Miyanishi et al. 95] this replication model is combined with a detailed workload model in which parameters $\lambda_{ij}$ denote the arrival rates of lock requests for data item *j* at site *i*, and $r(i,j) = 0$ implies $\lambda_{ij} = 0$. [Miyanishi et al. 95] do not exploit this expressiveness of their model definition but circumvent its complexity by assuming that all $\lambda_{ij}$ have the same value, which in turn implies full replication. Furthermore, they consider updates only and assume that communication is negligible.

In [Carey, Livny 88,96], files are assumed to be the unit of data replication. In the simulation experiments each file has the same number of copies (one, two or three copies, or full replication) so that the complex replication model is reduced to the 1-dimensional *all-objects-to-some-sites* scheme.

True replication-per-object models are of considerable complexity, because they entail that different sites hold different amounts of replicas and will hence be exposed to different workloads [Mukkamala 89]. Thus, the homogeneity assumption typically used in performance studies of distributed databases is violated so that the performance criteria would need to be calculated separately for each site. Hence, such models have actually not been applied in existing studies.

## 3.4 Data Access Models

A database performance model can either assume uniform data access or define a model of non-uniform (or *skewed*) data access. For mathematical tractability, most studies assume uniformly distributed data access, i.e. each data item is accessed with equal probability (e.g.: [Garcia-Molina 82], [Singhal 86, 90], [Ulusoy, Belford 92], [Banerjee et al. 94], [Son, Zhang 95]). Non-uniform data access is more realistic but used in very few studies of distributed database systems (e.g. [Triantafillou 96], [Hwang et al. 96], [Raghuram et al. 92], [Alonso et al. 90]). These models of non-uniform data access are usually adopted from evaluations of centralized databases. They can be classified to be either *hot-spot models* or *locality models*. In hot-spot models certain data groups (*hot spots*) are more likely to be accessed than others. In locality models local data is more likely to be accessed than remote data.

(1) Hot-spot models

The classical hot-spot model of non-uniform data access for centralized database systems is *b-c access* [Tay et al. 85]. Figuratively, the model of non-uniform *b-c access* describes that *b* % of the data requests are made to *c* % of the data items. More precisely, [Tay et al. 85] define that a fraction *c* of the data items in the database are called regular granules and a lock request is with probability *b* for a regular granule. Among regular granules, each granule is accessed with equal probability, and the same is assumed for non-regular granules.

Using the notions of *object-intensity* and *operation-intensity*, the simulations in [Triantafilliou, Taylor 95] and [Triantafillou 96] apply the *b-c access* model to individual transactions such that *b*% of the operations of each transaction are directed to *c* % of the data items which are access intensive. They set *b* to 0.8 and *c* to 0.01, 0.05, and 0.25 . The simulation in [Wu 93] uses *0.8-0.2 access*. [Gray, Reuter] argue that for many applications *0.99-0.01 access* is a realistic model.

[Zhang, Hsu 96] present a generalization of the *b-c access* pattern to allow for more arbitrary distributions of non-uniform data access. Their approach is to assume that the database *D* is divided into *k* classes of granules, $D_1, \dots, D_k$, with

$$D = \bigcup_{j=1}^{k} D_j$$

The probability of a request being made to a particular granule in class $D_j$ is $p_j$. The probability that a request is made to any granule in $D_j$ is $\left|D_j\right| \cdot p_j$ such that

$$\sum_{j=1}^{k} \left|D_j\right| \cdot p_j = 1$$

For $k$ = 2 this generalized model corresponds to the *b-c access* model in [Tay et al. 85].
Such hot-spot models capture the „skewness“ of the data access pattern which in turn has significant impact on the evaluation of lock conflicts (see section 3.5.2).

(2) Locality models
[Raghuram et al. 92] use a data access model called *b-l access* which simply describes that *b* % of the data request can be satisfied locally. Note that this is substantially different to the *b-c access* model in [Tay et al. 85]. If a transaction requests a local data object, each local data object is accessed with equal probability and the same holds for remote data objects respectively. [Raghuram et al. 92] use a value of $b$ = 0.8, while [Hwang et al. 96] assume $b$ = 0.5 to express that 50% of the primary copies are accessed locally. The analytical model in [Alonso et al. 90] assumes that locally cached data is *f* times more likely to be accessed than other data, where *f* is set to 5 and the fraction of cached data ranges from 0% to 20%.

## 3.5 Transaction Processing Models

Replica control protocols assumed in performance evaluations of distributed databases include ROWA, primary copy with synchronous and asynchronous update propagation, as well as optimistic and quorum based algorithms [Helal et al. 96]. Concurrency control protocols (distributed two-phase locking, optimistic methods, etc.) to capture lock conflicts and blocking of transactions are usually only modeled to compare concurrency control algorithms [Garcia-Molina 82], [Singhal 86], [Carey, Livny 88], [Thanos et al. 88], [Ciciani et al. 90,92], [Keum et al. 95], [Liang, Tripathi 96]. Such models are of considerable complexity. They typically use simulations and simplified modeling assumptions concerning replication and communication.

### 3.5.1 Transaction Models

Since read-only transactions (queries) are easier to process for a database system than updates [Garcia-Molina, Wiederhold 82], a distinction in performance models is recommended unless a worst-case analysis is intended. Still, many models consider updates only [Ciciani et al. 90], [Singhal 86], [Mariasoosai, Singhal 90], [Raghuram et al. 92], [Miyanishi et al. 96]. Several studies model two transaction types (i.e. queries and updates, e.g. in [Alonso et al. 90], [Gallersdörfer, Nicola 95], [Anderson et al. 98]). This is sufficient for general performance considerations while the evaluation of real-world applications desires the ability to model more detailed workload patterns.

For simplicity, most studies do not consider distributed transaction processing. Some models assume a fully replicated database and that transactions can always be processed locally [Garcia-Molina 82], [Singhal 86], [McDermott, Mukkamala 94], [Keum et al. 95]. Others assume that due to skillful data fragmentation and allocation transactions can always be executed at a single site which is either the local or a remote site [Bouras, Spirakis 96], [Gallersdörfer, Nicola 95], [Kuang, Mukkamala 91], [Shyu, Li 90]. Distributed transaction processing is addressed in [Thomasian 93], [Simha, Majumdar 97] and [Mukkamala, Bruell 90]. Under varying assumptions regarding the data access pattern, the number of data objects, and the number of database sites, these studies calculate the average number of data objects referenced per transaction and the average number of remote sites accessed per transaction as performance measures. However, these models of distributed transactions are not used to compute response times or transaction throughput. Performance studies that consider distributed transaction processing in the response time and throughput analysis are typically simulation studies [Kemme, Alonso 98], [Carey, Livny 96]. Moreover, [Ciciani at al. 90] and [Yu et al. 93] consider two classes of transactions: Class 1 transactions submitted at site *k* only access data locally available at site *k*. Class 2 transactions access local as well as well remote data items using distributed two-phase locking with primary copy or a distributed optimistic protocol.

### 3.5.2 Lock Conflict Models

Standard locking protocols (i.e. two-phase locking with blocking or abort-and-restart upon lock conflict) are the most common concurrency control methods for database systems. Lock conflicts and the resulting effects on transaction performance have been investigated extensively for centralized databases . Many results can be extended to distributed databases [Dias et al. 87], [Ciciani et al. 90], [Yu at el. 93]. While [Ciciani et al. 90] and [Yu at el. 93] assume dynamic locking (i.e. locks are not acquired before they are needed), the performance of static locking in distributed database systems where all locks are obtained at the beginning of the transaction has been analyzed in [Shyu, Li 90], [Kuang, Mukkamala 91]. Dynamic locking is more realistic, because the a priori identification of all required locks is only possible at a very coarse granularity of locking or in special applications. A comprehensive survey of the general modeling concepts for concurrency control is given in [Thomasian 96,98], [Yu et al. 93]. [Cellary et al. 88] provide an introduction and various examples for performance modeling of distributed concurrency control.

Basically, the probability of lock conflicts depends proportionally on the average transaction arrival rate, the transaction size (i.e. number data objects accessed per transaction) and the lock holding time, and is inversely proportional to the total number of data items in the distributed database. The lock holding time depends on the delay of blocked transactions which in turn depends on the lock conflict probability. Hence, an iterative calculation is used in [Garcia-Molina 82] and [Ciciani et al. 90]. The lock conflict probability also depends on the data access pattern:

[Tay et al. 85] prove that under three approximations (A1), (A2), (A3) the performance of locking under *b-c access* (see section 3.4) in a database of size *D* is the same as that for uniform access in a database of size

$$\frac{D}{1+(b-c)^2 / c \cdot (1-c)}$$

This finding has been named the *database reduction approach* in [Zhang, Hsu 96] or the *effective database size paradigm* (EDSP) in [Thomasian 98]. The three approximations in [Tay et al. 85] are:

(A1) The number of locks held by a single transaction is negligible compared to the total number of locks held.

(A2) The rate of transactions which are aborted and restarted due to deadlocks is negligible compared to the system throughput.

(A3) The number of lock conflicts among three or more transactions is negligible compared to the number of lock conflicts that involve only two transactions.

These assumptions are generally accepted and also used for other purposes in analytical performance evaluations of databases. They can be justified by probabilistic considerations [Tay et al. 85].

Most models of distributed databases that capture lock conflicts consider updates and exclusive locks only [Miyanishi et al. 96], [Raghuram et al. 92], [Ciciani et al. 90], [Garcia-Molina 82]. With another type of EDSP it can be shown that the performance of a database of size $D$ with shared and exclusive lock requests is equivalent to that of a database of size $D/(1 - s^2)$ with exclusive locks only, where $s$ denotes the fraction of lock requests that are in shared mode [Tay et al. 85], [Thomasian 98]. Nevertheless, [Kuang, Mukkamala 94] and [Born 96] consider shared and exclusive locks. [Born 96] assumes that lock conflicts are negligible and investigates only the overhead requesting and releasing locks. [Ciciani et al. 90] distinguish between weak and strong locks: weak locks are requested during the execution of a transaction but can be preempted by strong locks in which case the preempted transaction is aborted. At the beginning of the two-phase commit of a transaction, weak locks are upgraded to strong locks. If any strong lock request is rejected (due to a conflict with another strong lock) the transaction is aborted.

## 3.6 Dependency Structures

Apart from how the various aspects of a real system are modeled, it also matters which *dependencies* between them are considered in the model. Figure 2-6 sketch the dependencies captured in some representative performance models[1]. Rectangular nodes represent input parameters and modeling assumptions, oval nodes stand for intermediate or final results, the latter in bold. An arrow from a node A to a node B indicates that B depends directly on A. Although more advanced than others, the model in [Hwang et al 96] assumes constant communication delay and unlimited network capacity. On the database side, they consider 1D-replication and do not allow different types of queries or updates (like all models we examined). [Alonso et al. 90] captures limited network capacity and load dependent communication delay (highlighted by the bold arrow), but this is not exploited for throughput calculation and combined with a 1D-replication model. In [Hwang et al. 96] and [Alonso et al. 90] the arrival rates depend on non-uniform data access, modeled by a factor for access locality

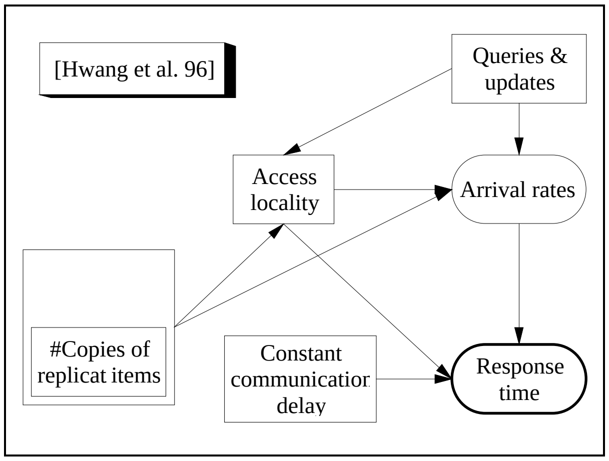

*Figure 2: Dependencies in [Hwang et al 96].*

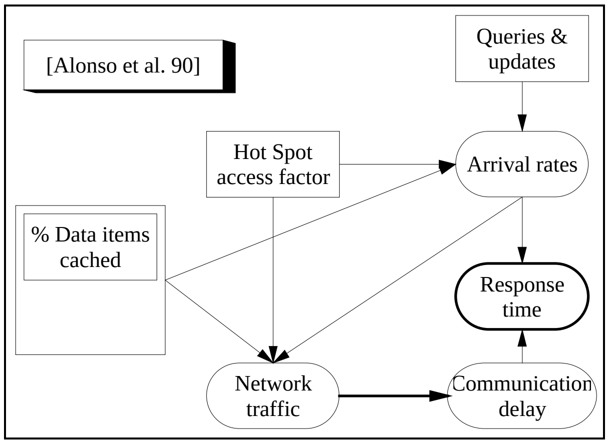

*Figure 3: Dependencies in [Alonso et al. 90].*

[1] Unfortunately, many papers do not provide sufficient details of their models to allow a characterisation like in figures 2-6. Moreover, note that the figures 2-6 are simplified representations of the corresponding models.

and hot spot access respectively, while such dependencies are neglected in [Ciciani et al. 90]. Unlike [Hwang et al. 96] and [Ciciani et al. 90], in the model in [Alonso et al. 90] the communication delay depends on the non-uniform data access and the 1D-replication model. [Ciciani et al. 90] consider lock conflicts and aborts in a detailed manner but are restricted to 1D-replication and neglect load-dependent communication delay. Among the models in figures 2 - 6 the one in [Keum et al. 95] is exceptional because it considers limited network capacity as well as lock conflicts in an integrated manner. The price paid for this is a simple database model assuming full replication and uniformly distributed data access. Moreover, the model evaluation is done exclusively by simulations which do not enjoy the low implementation effort and short run times of analytical models. [Keum et al. 95] and [Gallersdörfer, Nicola 95] consider 2 message types to distinguish between short and long messages. [Gallersdörfer, Nicola 95] is one of the very few analytical studies that model the quality of a replication schema and they consider throughput as a performance criterion[2]. However, they assume constant communication delay and unlimited network capacity which leads to quite simplistic throughput estimations (i.e. the throughput depends on the arrival and service process of the database nodes only).

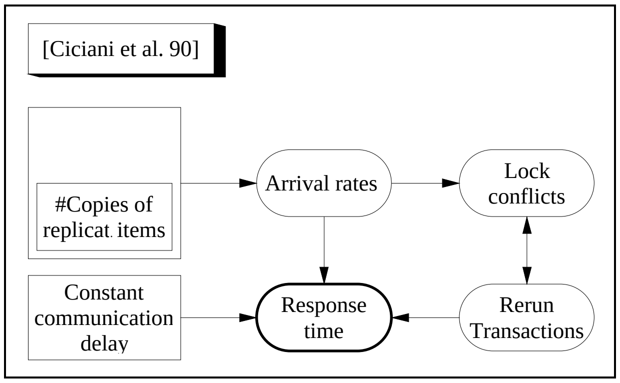

*Figure 4: Dependencies in [Ciciani et al. 90].*

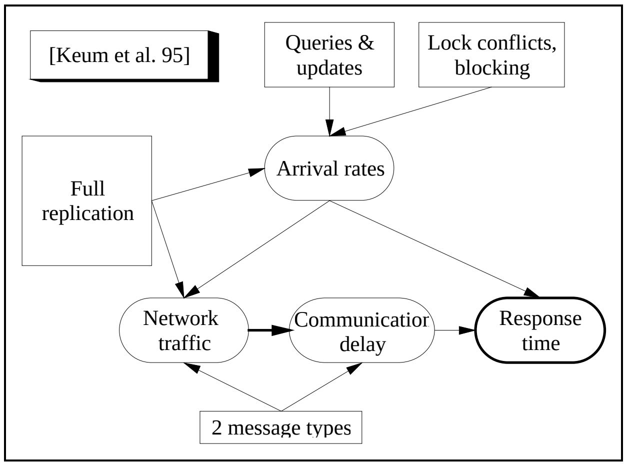

*Figure 5: Dependencies in [Keum et al. 95]*

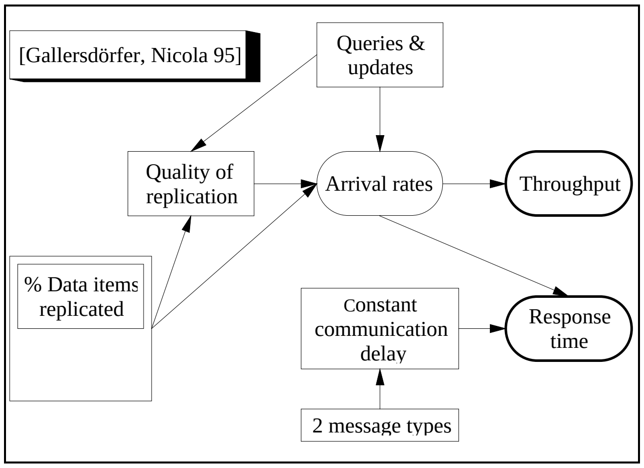

*Figure 6: Dependencies in [Gallersdörfer, Nicola 95]*

## 4 2RC: Example of an Analytical Performance Model

As a condensed summary of the survey, Table 2 shows typical modeling alternatives for the different model components. In each row of the table the complexity of the modeling options increases from left to right which often goes along with a higher expressiveness and accuracy of the model. The shaded cells of the table highlight the most common modeling options found in the majority of the studies. To illustrate a selection of the modeling

[2] The quality of a replication schema depends on its effects on transaction processing and intersite communication. Intuitively, a replication schema is considered „good“ if read intensive data is replicated while update intensive data is not. This yields a performance improvement due to increased local read access while performance degradation due to high update propagation overhead is avoided.

concepts classified in the preceding sections, we now demonstrate the stepwise development of an analytical performance model for distributed and replicated databases. In section 4.1 we define the desired scope of the model and section 4.2 describes the required dependencies. Viewing the classification of modeling concepts as a modular construction kit, we pick building blocks from Table 2 as the main components for our model. These are the bold framed cells in Table 2. The result is a *2*-dimensional *r*eplication model with integrated *c*ommunication (2RC) that neglects or simplifies less aspects than previous analytical models. Using this example, we sketch how to calculate network traffic, communication delay, response time and throughput as primary performance metrics.

| **Model components** | **Modeling Options** | | | |
|---|---|---|---|---|
| *Database sites:* | M/M/m system for *m* sites | Queueing network of *m* M/M/1 systems | Queueing network of *m* M/G/1 systems | One network per site ("network of *m* networks") |
| *Communication Model:* | constant delay, unlimit. capacity (M/D/∞) | non-constant delay, unlimited capacity (M/M/∞) | exponential delay, limited capacity (M/M/1) | general delay, limited capacity, (M/G/1) |
| *Replication Model:* | No replication | Full replication | partial, 1-dimensional | partial, 2-dimensional |
| *Quality of replication:* | Ignored | Replica placement | Replica selection | Replica placement & Replica selection |
| *Data Access:* | Uniform | Locality model | Hot-Spot model | Locality and Hot-Spot model |
| *Transaction workload:* | Queries only | Updates only | Updates & queries | More than 2 transaction types |
| *Concurrency control:* | Ignored | 2PL, exclusive locks only | 2PL, read and write locks[3] | Other protocols |

*Table 2: Alternative modeling options (Most common choices in gray cells, 2RC choices in bold cells)*

## 4.1 Scope of the model

2RC is intended to evaluate the interplay between replication and communication and thus to represent a balanced model of both the database and the communication part of a real system. Hence, 2RC is based on the new 2-dimensional model of replication (introduced in section 3.3) which is needed to represent and evaluate realistic replication schemata more accurately. Not only response time but also throughput is computable as an important performance criterion. This requires to capture load dependent communication delay, network limited transaction throughput, and the interdependencies between replication and communication. Moreover, bottleneck examinations are supported. The model is able to describe detailed transaction and communication patterns, so that real-world applications can be modeled. Furthermore, non-uniform data access, the quality of replication schemata and relaxed coherency are considered.

The model of transaction processing in 2RC follows the primary copy approach [Stonebraker 79], because it has been judged advantageous over other replica control concepts ([Gray et al. 96], [Keum et al. 95]) and is implemented in commercial systems like Sybase and Oracle. In 2RC, updates are assumed to be propagated asynchronously to the secondary copies, i.e. we do not model 2-phase-commit processing of updates. Furthermore, transactions are assumed to be executed at a single site, either the local or a remote site. Since our

[3] Although we do not cover concurrency control here, [Nicola 99] develops 2RCL: an extension of 2RC which integrates a detailed locking model for synchronous and asynchronous distributed updates.

model is not primarily intended to compare concurrency control algorithms, we refrain from modeling lock conflicts in order to concentrate on replication and communication[3]. In the following section we describe the dependency structure of 2RC.

## 4.2 Dependency Structure

Figure 7 sketches the structure of dependencies we consider in 2RC. The 2D-replication scheme (presented in section 3.3) is a core part of our model and has direct impact on the quality of replication, the arrival rates and the network traffic, and thus substantial influence on all further results. τ transaction types (with different arrival rates and service time distributions) and 2 message types per transaction (with individually distributed transmission times depending on the message size) allow to model a wide range of different applications and workload patterns. The two bold arrows highlight the important dependencies through which load dependent communication delay and network limited throughput are captured. The overall throughput depends on both the network and the local database throughput, which allows detailed bottleneck considerations. The quality of replication in 2RC is a 2D-extension of a concept in [Gallersdörfer, Nicola 95]. However, here the quality of replication (along with non-uniform data access and relaxed coherency) does not only affect the transaction arrivals but also the network load and thus the communication delay. 2RC combines a comprehensive replication model with a detailed communication model, and covers more interdependencies between the two than many previous studies. However, it is nevertheless a *model* in the sense that it is an incomplete and simplified representation of reality.

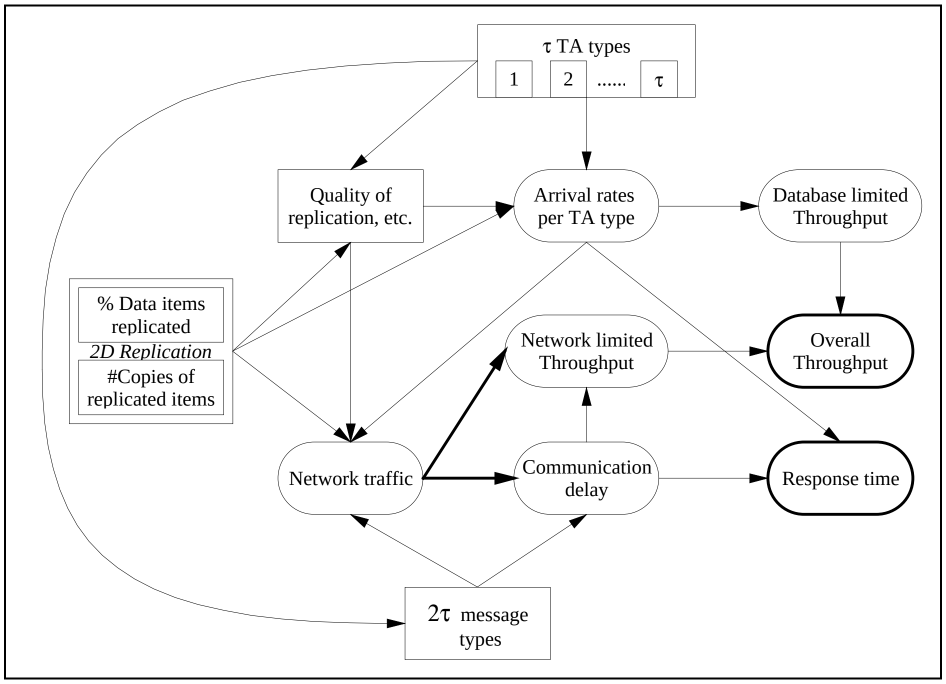

*Figure 7: Dependency structure of 2RC.*

The following sections present the mathematical elaboration of a queueing model of a replicated database according to the 2RC approach. Table 3 shows the model parameters that are going to be used.

| ***Parameter*** | ***Meaning*** |
|---|---|
| $n$ | Number of sites |
| $\tau$ | Number of different transactions types |
| $r_1$ | Fraction of data items replicated |
| $r_2$ | No. of copies of replicated data items |
| $\lambda_i$ | Arrival rate of transaction of type *i* per site (TPS) |
| $a_i$ | Fraction of transactions of type *i* |
| $q_i$ | Function to distinguish between queries and updates |
| $t_i$ | Mean service time for a transaction of type *i* (sec) |
| $loc_i$ | Transactions' locality without replication |
| $\ell_i$ | Probability of local transaction execution |
| $plcmt_i$ | Quality of replica placement |
| $sel_i$ | Quality of replica selection |
| $f\_plcmt_i$ | Fine tuning replica placement |
| $f\_sel_i$ | Fine tuning replica selection |
| $k$ | Coherency index |
| $bps$ | Communication bandwidth (in bits per second) |
| $size_c^{send\_i}$ | Message size for a *send* of a transaction of type *i* (bytes) |
| $size_c^{return\_i}$ | Message size of *return*ed query results (bytes) |
| $t_c^{send\_i}$ | Mean time to *send* a transaction of type (sec) |
| $t_c^{return\_i}$ | Mean time to *return* query results (sec) |

*Table 3: Input parameters.*

## 4.3 Workload and locality

We model a replicated database by an open queueing network in which each of the *n* identical sites is represented by a M/$H_\tau$/1 system. Transaction arrivals to the distributed system from outside are modeled by *n* identical Poisson processes with parameter $\lambda$ (one arrival stream per site). Poisson streams are known to be a good approximation for jobs submitted independently by a large number of users ([Kleinrock 75],[Jain 75]), and are used in nearly all performance models of distributed database (cf. Table 1). The $\tau$ different types of transactions are numbered 1,2,.....,$\tau$. A transaction is with probability $a_i$ of type *i*, $\sum_{i=1}^{\tau} a_i = 1$, so the Poisson arrival process at a site consists of $\tau$ separate streams having rate $\lambda_i = a_i \cdot \lambda$, $1 \le i \le \tau$. A characteristic function *q* distinguishes between queries and updates:

$$q: \{1,2,...,\tau\} \rightarrow \{0,1\} \quad \textit{with} \quad q(i) = q_i = \begin{cases} 1 & \text{, if transactions of type } i \text{ are queries} \\ 0 & \text{, if transactions of type } i \text{ are updates} \end{cases}$$

The number of data objects accessed per transaction are assumed to be geometrically distributed, so that the service time for a transaction of type *i* at a local database is modeled as exponentially distributed with mean $t_i$ (seconds). Hence, the service time for the combined arrival process of all $\tau$ transaction types follows a $\tau$-phase hyperexponential distribution.

We model non-uniform data access with two complementary approaches: access preferences for replicated data (in the next section) and a locality model (cf. (2) in section 3.4). The locality model captures access preferences for local data, because without replication but due to skillful data fragmentation and allocation, transactions exhibit a behaviour of locality in the sense that they tend to access data items locally available at their site of submission. This is modeled by the probability $loc_i \in [0;1]$ $(1 \le i \le \tau)$ that a transaction of type $i$ can be executed at the local site, while it has to be forwarded to a remote site with probability $1 - loc_i$. Introducing partial replication $(r_1, r_2)$ then increases the probability that a query can be answered locally by $(r_1 \cdot r_2)/n$. Due to the primary copy approach, the write availability does not increase.

## 4.4 The quality of replication

The *selection* of data items to replicate and the decision where to *place* them has significant impact on the overall system performance [Wolfson et al. 97]. Thus, we consider this quality of a replication design in the performance evaluation, exploiting the *undefined replica selection and placement* property of the 2-dimensional replication model. We find that *replica selection* has a major impact on updates, while *replica placement* is more significant for query processing. Hence, we model the impact of the quality of replication on the overall system performance by capturing the influence, which *replica selection* has on update processing and *replica placement* has on query processing:

**Updates:** Our model assumes, that updates have to be executed on the primary copy and update propagation is done in a decoupled, asynchronous fashion. Hence, for update transactions only the placement of the primary copy rather than the placement of the secondary copies matters. However, the replica selection is of much higher significance: Selecting many update intensive data items for replication causes high update propagation overhead. Thus, a "selection function" $sel_i$ for $1 \le i \le \tau$ and $q(i) = 0$, expresses to which extent updates of type $i$ tend to access data items that were selected for replication. The first argument of $sel_i$ is $r_1$ because the higher the fraction of replicated data items the more likely it is that not only read intensive but also update intensive data items are selected for replication. The second argument of $sel_i$ is the input parameter $f\_sel_i \in [-\infty, 1]$, and $sel_i$ is defined as

$$sel_i(r_1, f\_sel_i) = \frac{1}{(r_1)^{f\_sel_i}} \quad \in [0\,;\,1/r_1]$$

This definition is meaningful, because $sel_i$ can appropriately influence the way in which the update propagation overhead increases with the fraction $r_1$ of replicated data. The parameter $f\_sel_i$ is then used to "*fine-tune*" the speed with which the update overhead increases over $r_1$. A value of $f\_sel_i \to -\infty$ results in $sel_i \to 0$ which means that it was possible to select data items for replication which are never changed by updates of type $i$. $f\_sel_i = 0$ ($sel_i = 1$) signals no preferences towards replicated or unreplicated data, and $f\_sel_i = 1$ ($sel_i = 1/r_1$) declares that replica selection is so unfortunate that updates of type $i$ always access replicated data. The effects of the quality of replica selection on the update propagation overhead as well as intermediate

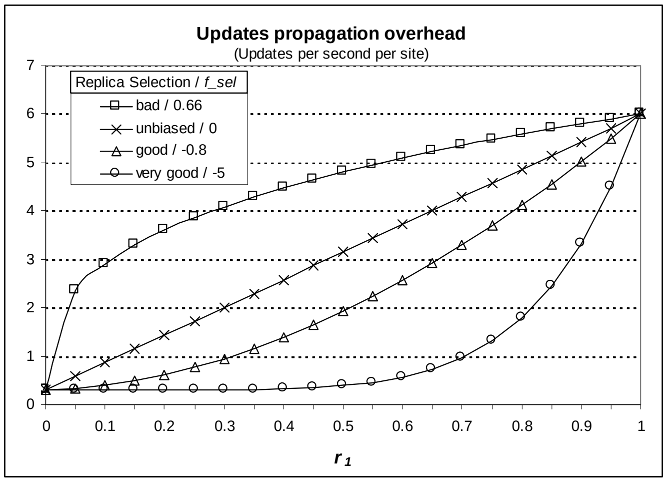

*Figure 8: Effects of the replica selection*

parameter values are illustrated[4] in Figure 8. The informal notion of the update propagation overhead is based on the transaction rate per site ($\lambda_i^{total}$) which is derived in detail in section 4.5. If unbiased replica selection is assumed ($f\_sel_i = 0$), update propagation increases linearly with the fraction of replicated data. This is what typically happens in existing performance models which do not consider the quality of replication. A “bad” replica selection tends to replicate update intensive data items such that the propagation overhead increases rapidly even for a low fraction of replicated data. The better the replica selection, the larger the fraction of replicated data which does not include update intensive objects and keeps the update propagation overhead low.

**Queries:** Although replica selection has some impact on queries, replica placement is much more crucial to query processing: Even if all read intensive data items are replicated, performance gains are quite low as long as these replicas are not available locally to the queries. Thus, a "placement function" $plcmt_i$ for $1 \le i \le \tau$ and $q(i) = 1$ expresses to which extent replica placement is increasing the probability that a query of type $i$ can be executed locally. The first argument of $plcmt_i$ is $(r_1 \cdot r_2)/n$ because the higher the degree of replication the more likely it is that replication increases the local read probability. The second argument of $plcmt_i$ is the input parameter $f\_plcmt_i \in [0;1]$, and $plcmt_i$ is defined as

$$plcmt_i\,(r_1,r_2,n,f\_plcmt_i) = \frac{1}{\left(\frac{r_1 \cdot r_2}{n}\right)^{f\_plcmt_i}} \in [1;\, n/(r_1 \cdot r_2)]$$

A value of $f\_plcmt_i = 0$ ($plcmt_i = 1$) means that replica placement does not necessarily increase the chance for queries of type $i$ to be answered locally, and $f\_plcmt_i = 1$ ($plcmt_i = n/(r_1 \cdot r_2)$) declares that queries of type $i$ can always run at their home sites. Intermediate values have intermediate effects on the local read probability: Assuming that the replicas are distributed evenly across the sites, each site receives an equal share of forwarded transactions and propagated updates. Thus, the overall probability $\ell_i$ that a transaction of type $i$ ($1 \le i \le \tau$) can be executed at its local site amounts to

$$\ell_i = loc_i + q_i \cdot (1 - loc_i) \cdot \frac{r_1 \cdot r_2}{n} \cdot plcmt_i$$

because without replication a transaction is processed locally with probability $loc_i$ (first term). The second term is added for queries only (i.e. if $q_i = 1$) because due to the primary copy approach replication does not increase the locality of updates. The second term captures the fact, that a fraction of the queries which cannot be executed locally without replication (lets call them “potential remote queries”), might nevertheless be processed locally depending degree of replication $(r_1 \cdot r_2)/n$ and the quality of the replica placement ($plcmt_i$). If an unbiased replica placement is assumed ($f\_plcmt_i = 0$), the probability that potential remote queries are executed locally increases linearly with the degree of replication (see Figure 9)[5]. This is what typically happens in existing models which do not consider the quality of replication. In our model, intermediate values (greater than 0) of $f\_plcmt_i$ can express, how much a positive replica placement increases the probability of local access even for low degrees of

[4] Figure 8 assumes 50 sites, $r_2 = 20$ copies, and an arrival rate of 0.3 user submitted updates per second per site.
[5] Figure 9 assumes 50 sites and $loc_i = 2/50$, i.e. local data is accessed twice as often as any of the remote sites.

replication, whereas extending an already high degree of replication causes a much lower increase of locality. This behaviour of the model captures the well known phenomenon, that with a stepwise enlargement of a cache memory the increase of the cache hit-ratio decreases from step to step [Gray, Reuter 93][6]. Since replication can be viewed as a form of caching [Rahm 93], the model of locality and quality of replication is an appropriate approximation of the effects which replication has on the local access probability.

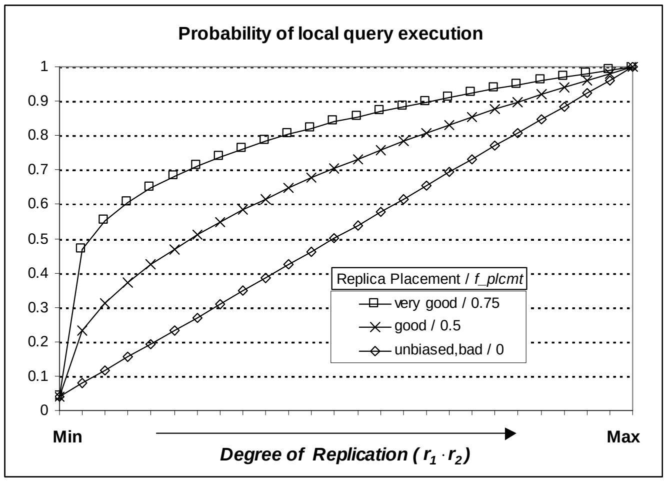

*Figure 9: Effects of replica placement on* $\ell_i$

Table 4 summarizes the model of the quality of replication, the range of possible values and the associated meanings concerning replica selection and replica placement.

| | **Range** | **Best selection** | **unbiased selection** | **Worst selection** |
|---|---|---|---|---|
| $f\_sel_i$ | $[-\infty; 1]$ | $\rightarrow -\infty$ | 0 | 1 |
| $sel_i$ | $[0;1/r_1]$ | 0 | 1 | $1/r_1$ |
| | **Range** | **Best placement** | **Unbiased/bad placement** | |
| $f\_plcmt_i$ | $[0;1]$ | 1 | 0 | |
| $plcmt_i$ | $[1; n/(r_1 \cdot r_2)]$ | $n/(r_1 \cdot r_2)$ | 1 | |

*Table 4: Parameters modeling the quality of replication*

The interval of values for *f_plcmt* starts at 0 instead of -∞ because the model assumes that replica placement can never decrease the local read probability. However, values of *f_plcmt* < 0 could be used to model an alarming scenario in which replicas are placed at sites where they are not used, causing pure overhead. For *f_plcmt* = 0 the „unbiased“ replica placement is still quite good when the overall degree of replication is high. Mathematically, this is because the optimum value for *plcmt* (i.e. $n/(r_1 \cdot r_2)$ ) converges towards 1 as the degree of replication increases. Intuitively, the higher the degree of replication the more difficult it is to design a „bad“ replica placement. Similarly, the larger the fraction of replicated data items the more difficult it is to design a „bad“ replica selection: if $f\_sel_i$ was set to 1 in order to define a bad replica selection, $sel_i$ necessarily becomes 1 as the fraction of replicated data items (i.e. $r_1$) converges towards 1. Intuitively, if *all* data items are selected for replication, there is no way one could *only* pick the update intensive data items. Hence, if *all* data items are selected for replication, the replica selection is necessarily unbiased. Note, that $loc_i$, $sel_i$ and $plcmt_i$ model non-uniform data access to replicated data.

## 4.5 Transaction processing and arrival rates

The model of transaction processing follows the primary copy approach [Stonebraker 79]. Updates are assumed to be propagated asynchronously to the secondary copies, i.e. we do not model 2-phase-commit processing of updates. Furthermore, transactions are assumed to be executed at a single site, either the local or a remote site.

[6] This is analogous to a process' page fault rate which decreases rapidly at first but then more and more slowly as it is allocated more and more main memory frames [Silberschatz, Galvin 94].

The performance of replicated databases can be improved if the requirement of mutual consistency among the replicas of a logical data item is relaxed. Various concepts of relaxed coherency can be denoted by coherency conditions which allow to calculate a *coherency index* $k \in [0;1]$ as a measure of the degree of allowed divergence [Gallersdörfer, Nicola 95]. Small values of $k$ express a high relaxation of coherency, $k = 0$ models suspended update propagation, and for $k = 1$ updates are propagated immediately. For example, instead of immediate update propagation, updates on a data item $x$ could be propagated to the secondary copies of $x$ periodically every $m$ time units. If $x$ is updated $\lambda_u^{(x)}$ times per second not every update but only the latest state of $x$ has to be propagated. Hence, the actual probability of propagation is

$$k = 1/((\lambda_u^{(x)} \cdot m) + 1)$$

and used as a coherency index. If the relaxation of coherency should be version oriented rather than time oriented, the secondary copies of $x$ can be updated after every $i$th update on $x$. The resulting coherency index is $k = 1/i$. Further details can be found in [Gallersdörfer, Nicola 95].

Taking locality, update propagation, the quality of replication, and relaxed coherency into account, the total arrival rate $\lambda_i^{total}$ of transactions of type $i$ ($1 \le i \le \tau$) at a single site amounts to

$$\lambda_i^{total} = \ell_i \cdot \lambda_i \quad + \quad (n-1)\cdot(1-\ell_i)\cdot\lambda_i \cdot \frac{1}{n-1} \quad + \quad (1-q_i)\cdot(n-1)\cdot(r_1 \cdot sel_i)\cdot\frac{r_2-1}{n-1}\cdot k \cdot \lambda_i$$

because a share of $\ell_i$ of the incoming $\lambda_i$ transactions can be executed locally (first term) whereas the remaining $(1-\ell_i)\cdot\lambda_i$ transactions are forwarded to sites where appropriate data is available. The other $n$-1 sites also forward $1-\ell_i$ of their $\lambda_i$ transactions which are received by each of the remaining databases with equal probability $1/(n-1)$. This explains the second term. If transactions of type $i$ are updates (i.e. $1 - q(i) = 1$) the arrival rate $\lambda_i^{total}$ is increased by update propagation from the $n$-1 remote sites. The probability that an update at one of the $n$-1 remote sites hits a primary copy which is replicated, is $r_1 \cdot sel_i$. The probability, that one of the corresponding secondary copies resides at the local database is $(r_2-1)/(n-1)$ because the $r_2 - 1$ secondary copies are distributed evenly over $n$-1 sites. Finally, update propagation may be reduced by relaxed coherency, i.e. if $k < 1$. The above formula simplifies to

$$\lambda_i^{total} = \lambda_i + (1 - q_i) \cdot (r_1 \cdot sel_i) \cdot (r_2 - 1) \cdot k \cdot \lambda_i$$

Note that the $r_1 \cdot sel_i$ (i.e. the probability that an update hits a replicated data item which is replicated) is $r_1$ for unbiased replica selection ($sel_i = 1$). It is 0 if an optimal replica selection managed not to replicate data items which are subject to modifications ($sel_i = 0$), and it is 1 if updates always hit replicated data items ($sel_i = 1/r_1$). Hence, in the model partial replication ($r_1 < 1$) with the worst replica selection has the same effect on $\lambda_i^{total}$ as replicating all data items ($r_1 = 1$) with unbiased replica selection. For further use we define

$$\lambda^{total} := \sum_{i=1}^{\tau} \lambda_i^{total}$$

## 4.6 Intersite communication

Two messages are required to execute a transaction at a remote site: a *send* and a *return*, e.g. a query is sent to a site and the result is returned. We assume that for each transaction type *i* the communication delay for a *send* (*return*) is not constant but exponentially distributed with mean $t_c^{send\_i}$ ($t_c^{return\_i}$) seconds ($1 \le i \le \tau$), because we expect short or medium messages to occur more frequently than very long messages. These mean values mainly depend on the bandwidth and the message size. Therefore, the parameter *bps* represents the network's bandwidth in bits per second, and $size_c^{send\_i}$ ($size_c^{return\_i}$) denotes the mean message size in bytes for a *send* (*return*) of a transaction of type *i*. The means $t_c^{send\_i}$ and $t_c^{return\_i}$ characterizing the exponentially distributed service times are hence defined as

$$t_c^{send\_i} = \frac{8 \cdot size_c^{send\_i}}{bps} \quad and \quad t_c^{return\_i} = \frac{8 \cdot size_c^{return\_i}}{bps}$$

The average number of messages per second in the distributed system amounts to

$$\overline{N} = \sum_{i=1}^{\tau} (1+q_i) \cdot n \cdot (1-\ell_i) \cdot \lambda_i \quad + \quad (1-q_i) \cdot n \cdot (r_1 \cdot sel_i) \cdot (r_2 - 1) \cdot k \cdot \lambda_i$$

$$= \sum_{i=1}^{\tau} n \cdot (1-\ell_i) \cdot \lambda_i \quad + \quad (1-q_i) \cdot n \cdot (r_1 \cdot sel_i) \cdot (r_2 - 1) \cdot k \cdot \lambda_i \quad + \sum_{i=1}^{\tau} q_i \cdot n \cdot (1-\ell_i) \cdot \lambda_i \qquad (*)$$

The first sum covers messages of type *send* (transactions forwarded to remote sites due to a lack of appropriate local data and update propagation), the second sum are returned query results. These results follow straight from the transaction arrival rates. Remote updates are assumed not to be acknowledged and thus do not cause return messages.

Unlike most existing models (cf. Table 1), we capture limited network capacity: Each local database is considered to be connected to the network via a local communication server modeled as an M/$\mathrm{H}_{2\tau}$/1 system. The arrival rate at any such server is $\overline{N}/n$ messages per second, because each site sends and receives the same amount of messages due to the sites' identical layout and symmetrical behaviour (*homogeneity*). The service time follows an $\mathrm{H}_{2\tau}$ distribution, because τ transaction types imply 2τ different message types: τ message types have an exponentially distributed service time with mean $t_c^{send\_i}$, and τ message types with mean $t_c^{return\_i}$ ($1 \le i \le \tau$.).

The expression (*) implies, that a share of

$$share^{send\_i} = \frac{n \cdot (1-\ell_i) \cdot \lambda_i \quad + \quad (1-q_i) \cdot n \cdot (r_1 \cdot sel_i) \cdot (r_2 - 1) \cdot k \cdot \lambda_i}{\overline{N}}$$

of the $\overline{N}$ messages has mean service time $t_c^{send\_i}$ (for $1 \le i \le \tau$) and a share of

$$share^{return\_i} = \frac{q_i \cdot n \cdot (1-\ell_i) \cdot \lambda_i}{\overline{N}}$$

of the messages has mean service time $t_c^{return\_i}$ (for $1 \le i \le \tau$). Hence, the first and second moment of the $\mathrm{H}_{2\tau}$ service time distribution can be derived following [Kleinrock 75], and result in

$$E(B_c) = \sum_{i=1}^{\tau} \left( share^{send\_i} \cdot t_c^{send\_i} \right) + \left( share^{return\_i} \cdot t_c^{return\_i} \right)$$

and $$E(B_c^2) = \sum_{i=1}^{\tau} \left( 2 \cdot share^{send\_i} \cdot \left(t_c^{send\_i}\right)^2 \right) + \left( 2 \cdot share^{return\_i} \cdot \left(t_c^{return\_i}\right)^2 \right)$$

respectively. The average waiting time $\overline{W_c}$ at a local communication server can be obtained using the *Pollaczek-Khinchin formula* for general M/G/1 systems [Kleinrock 75].

$$\overline{W_c} = \frac{\lambda \cdot E(B_c^2)}{2(1-\rho)} \quad = \quad \frac{\frac{\overline{N}}{n} \cdot \frac{E(B_c^2)}{2}}{1 - \frac{\overline{N}}{n} \cdot E(B_c)}$$

$$= \frac{\sum_{i=1}^{\tau} \left( (1-\ell_i) \cdot \lambda_i + (1-q_i) \cdot (r_1 \cdot sel_i) \cdot (r_2 - 1) \cdot k \cdot \lambda_i \right) \cdot \left(t_c^{send\_i}\right)^2 + q_i \cdot (1-\ell_i) \cdot \lambda_i (t_c^{return\_i})^2}{1 - \sum_{i=1}^{\tau} \left( (1-\ell_i) \cdot \lambda_i + (1-q_i) \cdot (r_1 \cdot sel_i) \cdot (r_2 - 1) \cdot k \cdot \lambda_i \right) \cdot t_c^{send\_i} + q_i \cdot (1-\ell_i) \cdot \lambda_i \cdot t_c^{return\_i}}$$

Note that the utilization ρ can be expressed as $\rho = \lambda_c \cdot E(B_c)$ and the arrival rate is here $\lambda_c = \overline{N}/n$.

## 4.7 Performance criteria

We consider the average response times and the transaction throughput as performance criteria. Similar to the calculation of $\overline{W_c}$, the mean waiting time $\overline{W}$ at a local database is found to be

$$\overline{W} = \frac{\lambda^{total} \cdot \sum_{i=1}^{\tau} \frac{2 \cdot \lambda_i^{total}}{\lambda^{total}} \cdot t_i^{\,2}}{2 \cdot \left( 1 - \lambda^{total} \cdot \sum_{i=1}^{\tau} \frac{\lambda_i^{total}}{\lambda^{total}} \cdot t_i \right)} \quad = \quad \frac{\sum_{i=1}^{\tau} \lambda_i^{total} \cdot t_i^{\,2}}{1 - \sum_{i=1}^{\tau} \lambda_i^{total} \cdot t_i}$$

so that the combined average response time over all transaction types results in

$$\overline{R} = \sum_{i=1}^{\tau} a_i \cdot \overline{R_i} \quad , \quad \text{where} \quad \overline{R_i} = \overline{W} + t_i + (1-\ell_i) \cdot (\overline{W_c} + t_c^{send\_i} + t_c^{return\_i})$$

is the response time for transactions of type *i*. On average a transaction (of type *i*) needs to wait for $\overline{W}$ seconds at a database to receive a service of $t_i$ seconds. Additionally, with probability $(1-\ell_i)$ a transaction needs to be forwarded to a remote site which takes $\overline{W_c}$ seconds to wait for plus the time to be sent and returned.

In steady state, the throughput of the local databases equals the arrival rate $\lambda$ but is bounded by the limited system capacity. Specifically, the throughput can grow until either a local database server or a communication server (the network) is saturated, i.e. its utilization ($\rho_D$ or $\rho_C$ respectively) reaches 1. Since the utilization equals the product of arrival rate and mean service time ($\rho = \lambda \cdot E(B)$ ) the utilization $\rho_D$ of a local database can be expressed as

$$\rho_D = \lambda^{total} \cdot \sum_{i=1}^{\tau} \frac{\lambda_i^{total}}{\lambda^{total}} \cdot t_i \quad = \sum_{i=1}^{\tau} \lambda_i \cdot \big(1 \;+\; (1-q_i)\cdot(r_1 \cdot sel_i)\cdot(r_2-1)\cdot k\big)\cdot t_i \,.$$

Now the equation $\rho_D = 1$ can be solved for the arrival rate $\lambda$ which yields the maximum database limited throughput $T_D$:

$$T_D = \lambda = \left( \sum_{i=1}^{\tau} a_i \cdot \big(1 \;+\; (1-q_i)\cdot(r_1 \cdot sel_i)\cdot(r_2-1)\cdot k\big)\cdot t_i \right)^{-1} .$$

The utilization $\rho_C$ of a local communication server is

$$\rho_C = \frac{\overline{N}}{n} \cdot E(B_c)$$

and solving $\rho_C = 1$ for $\lambda$ results in the maximum communication limited throughput $T_C$:

$$T_C = \left( \sum_{i=1}^{\tau} \big((1-\ell_i)\cdot a_i + (1-q_i)\cdot(r_1 \cdot sel_i)\cdot(r_2-1)\cdot k \cdot a_i\big)\cdot t_c^{send\_i} + q_i \cdot (1-\ell_i)\cdot a_i \cdot t_c^{return\_i} \right)^{-1}$$

The maximum throughput at a database site is $T = min(T_D,T_C)$ because whatever server is saturated first (either the database or the communication server) is the bottleneck and limits throughput. The overall system throughput amounts to $n \cdot T$.

## 5 Applications of Analytical Performance Models

To demonstrate the application range of analytical performance models, we show how 2RC can assist in the evaluation of a variety of common design issues. This clarifies that the expressiveness of a 2D-replication model combined with an advanced communication model provides in depth performance estimations. Although the model allows to consider detailed application characteristics (workload, communication, etc.), we stick to a simple example for ease of presentation. (For complex applications, see [Nicola 99]). We consider 3 transaction types: short queries, updates, and long queries (e.g. statistical evaluations). The results are based on the parameter values shown in Table 5 unless otherwise stated. The base settings are carefully chosen after measurements in database systems for telecom applications [Gallersdörfer, Jarke, Nicola 99]. The values also agree with those found in [Alonso et al. 90], [Ciciani et al. 90].

| ***Parameter*** | ***Base setting*** |
|---|---|
| $n$ | 50 |
| $\lambda$ | 3 (i.e. 150 TPS total) |
| $\tau$ | 3 |
| $q_i$ | $q_1 = 1$, $q_2 = 0$, $q_3 = 1$ |
| $a_i$ | $a_1 = 0.85$, $a_2 = 0.1$, $a_3 = 0.05$ |
| $loc_i$ | 0.04 |
| $f\_plcmt_i$ | 0.55 |
| $f\_sel_i$ | -0.8 |
| $k$ | 1 |
| $t_i$ | $t_1 = 0.06$, $t_2 = 0.125$, $t_3 = 0.5$ |
| $bps$ | 64000 (e.g. ISDN) |
| $size_c^{send\_i}$ | 400 byte each |
| $size_c^{return\_1}$ | 2000 byte |
| $size_c^{return\_3}$ | 3500 byte |

*Table 5: Parameter base values*

Although we will mention absolute values to refer to characteristics in the diagrams below, we consider the general trends and shapes of the graphs as the primary results. We will discuss several parameter variations which a part of an sensitivity analysis which can not be completely presented here. However, it showed that parameter variations affect the performance values in a reasonable way, i.e. the model is stable and realistic modifications of the parameter settings change the absolute values of the performance measures rather than their general trends.

In the following sections we first present the main throughput and response time results and then examine how variations in the network capacity, system size, and quality of replication affect the system's throughput, bottleneck, and scalability characteristics. These performance metrics are also discussed with respect to availability. Particular replica control methods (e.g. quorum protocols) or specific fault tolerance examinations allow for different definitions of availability. Since we assume the primary copy approach we define availability as the *local read availability* of logical data items and consider it proportional to the extent of replication, i.e. the overall number of replicas.

## 5.1 Throughput

A typical goal in the design of a distributed information system is to achieve both a high transactional throughput and sufficient data availability. Obviously, this involves a trade-off because full replication provides optimum data availability but is known to deteriorate performance in most cases. The trade-off can be estimated using the result in Figure 10 which shows the maximum throughput $T(r_1,r_2) = min(T_D,T_C)$ over the $r_1$-$r_2$-space in transactions per second (TPS). A 1-dimensional replication model considers either the „$r_1$ = 1-edge" of the graph, or the „$r_2$ = 50-edge". Either case merely expresses that the throughput increases with a moderate degree of replication ($r_1$ = 0.3 or $r_2$ = 10) but decreases remarkably when replication is medium or high. However, the 2D-model tells us more: As long as less than 35% of the data items are replicated ($r_1$ < 0.35) the throughput can be maximized by placing copies on all sites ($r_2$ = 50), reaching its highest peak of 325 TPS for $r_1$ = 0.3. If availability considerations require more data items to be replicated (e.g. 50%), a medium number of copies yields the maximum throughput (e.g. $r_2$ = 30 for $r_1$ = 0.5). When striving for very high availability, it is worthwhile to consider that replicating 75% of the data items to 40 sites (out of 50) allows a twice as high throughput as full replication. Such results cannot be obtained with a performance model that considers 1D-replication or unlimited communication capacity. Furthermore, Figure 10 shows that the two 1D-models of replication differ considerably in their throughput estimations: while the *all-objects-to-some-sites* model predicts a maximum throughput of 250 TPS (for $r_2$ = 10) the *some-objects-to-all-sites* model predicts 325 TPS, which is 30% higher.

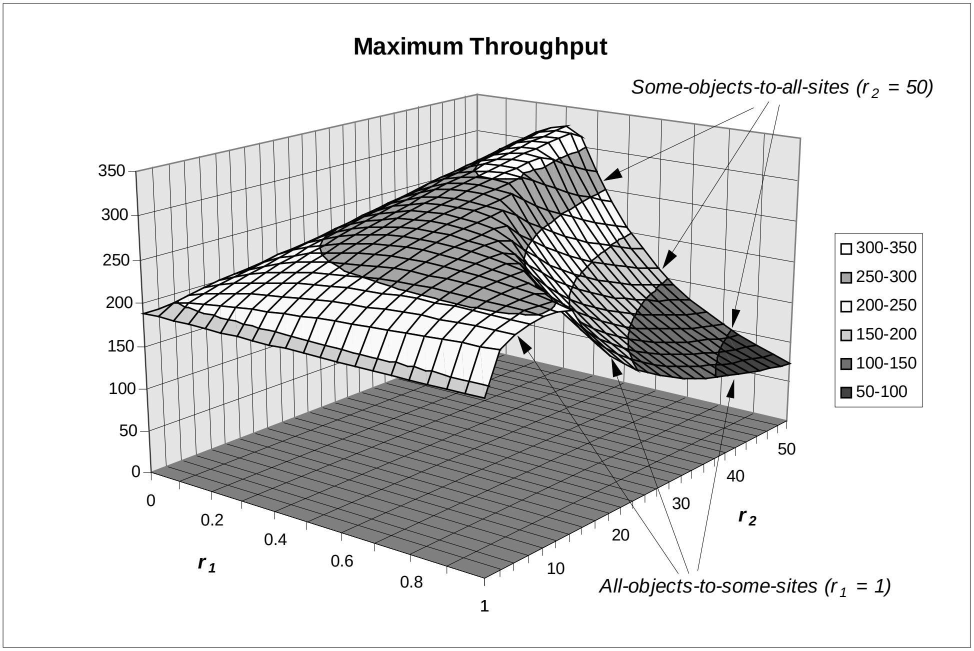

*Figure 10: Overall system throughput*

Let us briefly illustrate a result mentioned in the survey (cf. section 3.2): Models which assume unlimited network capacity cannot foresee, that a moderate degree of replication can increase the overall transaction throughput. This is because an infinite service capacity can not capture situations in which the network starts getting congested. Figure 11 shows the different throughput results produced by models which assume limited or unlimited network capacity respectively. To illustrate the situation clearly, we kept $r_2$ at a fixed value and varied $r_1$ only. If $r_1 > 0.35$, the database nodes are the bottleneck and both modeling approaches agree that the throughput decreases as replication is extended. For $r_1 < 0.35$, the network is the bottleneck, which is only captured by models which consider limited network capacity. In this case, extending replication from $r_1 = 0$ to $r_1 = 0.3$ increases the probability of local access without causing too much update propagation traffic on the network. Hence, the network is relieved and the throughput increases.

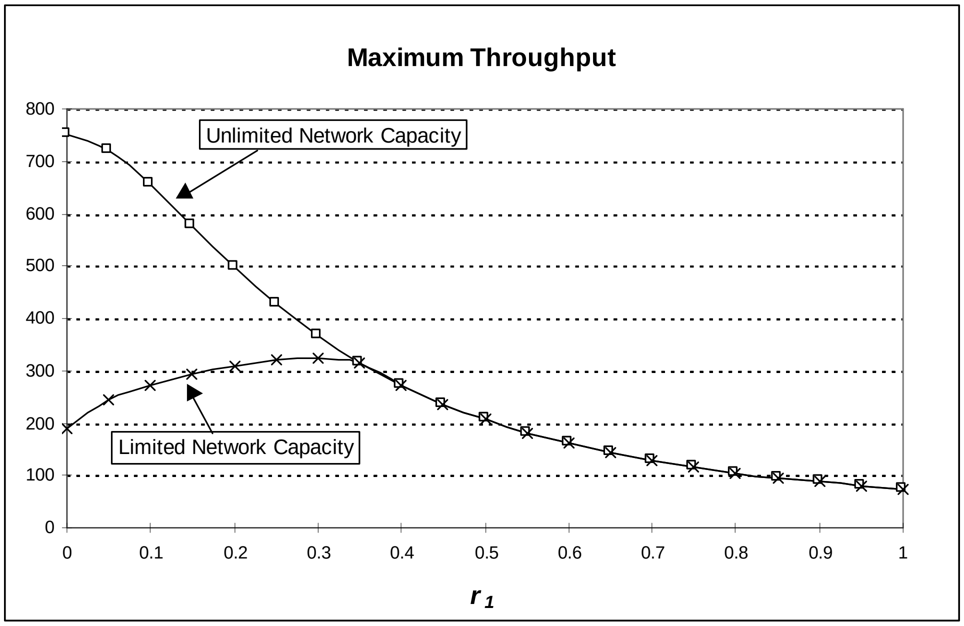

*Figure 11: Comparison of modeling limited vs. unlimited network capacity*

If a distributed application requires a higher throughput for a given replication schema, one important question is whether more database or more network capacity is needed. Since the throughput calculation considered limited network capacity, a bottleneck analysis is possible for any replication schema and can be read off Figure 12: Low replication ($r_1 < 0.3$ or $r_2 < 10$) entails low local read availability which in turn causes a lot of remote data access. This communication saturates the network earlier than update propagation saturates the local databases. Thus, for $r_1 < 0.3$ or $r_2 < 10$ the network is the throughput limiting bottleneck.

When replication is extended to medium or high replication (i.e. $r_1 > 0.5$, $r_2 > 25$) there are three simultaneous effects on the distributed system that can be read off the resulting performance formulas: (1) More update propagation

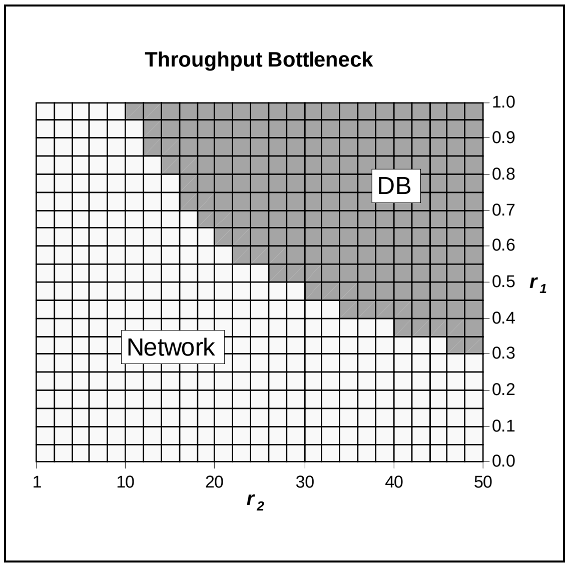

*Figure 12: Bottleneck Analysis*

generates higher network traffic, (2) the local data availability increases and relieves the network, and (3) the transaction load to update secondary copies at the database nodes grows. Figure 12 proves that the effects (2) and (3) eventually outweigh effect (3) and show for which replication schemata the local databases become the throughput bottleneck.

## 5.2 Response time

In most distributed applications there is typically a number of different possible replication strategies. Our model helps to compare the influence of different replication schemata on the response time. Figure 13 shows the combined average response time $\overline{R}$ (cf. section 4.7) over the *$r_1$-$r_2$-space* in seconds. A moderate degree of replication ($r_1 < 0.5$, $r_2 < 20$) leads to increased local data access which avoids communication delay and thus reduces response time from over 1 to less than half a second. From here on, either one of the 1D-models of replication (i.e. $r_1 = 1$ or $r_2 = 50$ case respectively) predicts that a further extension of replication rapidly saturates the system with propagated updates which predominates the advantage of local read access and causes high response times. However, the 2D-model reveals that this is only true if replication is extended along *both* dimensions and that low response times are still possible if replication is extended along one dimension but kept moderate in the other. Replication schemata as different as ($r_1 = 1$, $r_2 = 10$), ($r_1 = 0.3$, $r_2 = 50$) or ($r_1 = 0.5$, $r_2 = 30$) could be chosen to satisfy an application's individual requirements regarding reliability and availability, while retaining similarly low response times. A comparison of the bottleneck analysis in Figure 12 with the

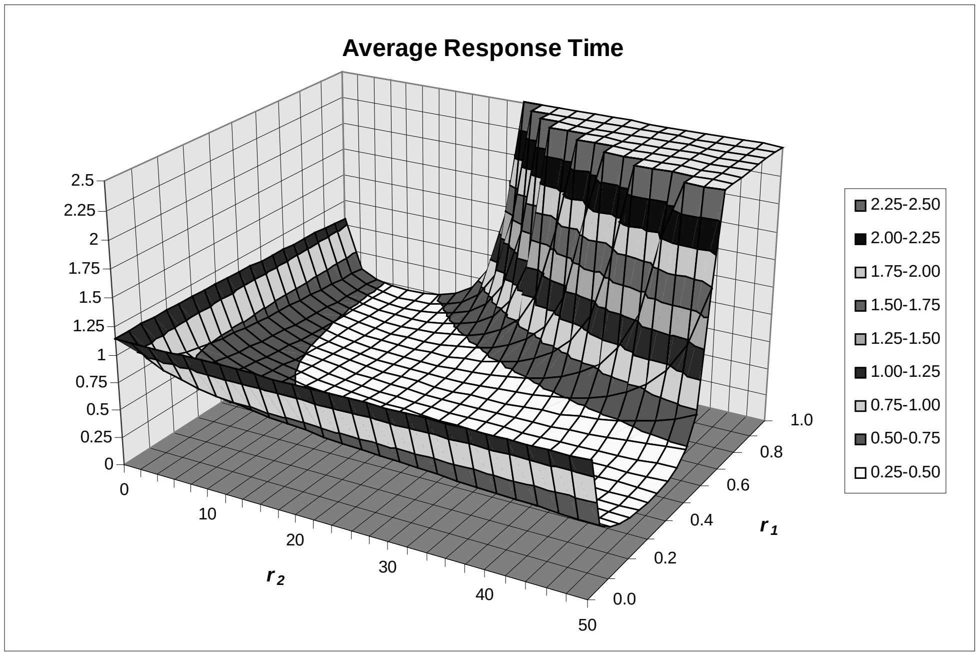

*Figure 13: Average overall response time* $\overline{R}(r_1, r_2)$

response time result in Figure 13 indicates that the very high response times for extensive replication (i.e. $r_1 > 0.7$, $r_2 > 35$) does not result from a saturation of the network with update propagation messages. Instead, the local database nodes are saturated with the corresponding transactions which update secondary copies. An examination of the expressions in the response time formula proves this finding.

Figure 14 illustrates how relaxed coherency can remarkably improve response time. A fixed value of $r_2 = 25$ is assumed. The case $k = 1$ represents asynchronous but immediate propagation of updates. In this situation the response time can be minimized by replicating about 40% of the data since this leads to increased local data access whereas a higher degree of replication causes too much update propagation overhead. This can be remedied if the coherency requirements are relaxed. A deferred update strategy for replicas could be applied which propagates only the *latest* value of a primary copy instead of *all* intermediate updates. If it reduces the update propagation overhead lets say by a factor of 2, this results in a coherency index of $k$ = 0.5 for which a higher degree of replication ($r_1$ = 0.6) yields the optimum response time. Additionally, replicating all data items ($r_1$ = 1) becomes feasible which could be required for availability reasons. For very low coherency requirements ($k$ < 0.25) the response time can be minimized by means of full replication. A wider discussion of relaxed coherency can be found in [Gallersdörfer, Nicola 95], [Alonso et al. 90].

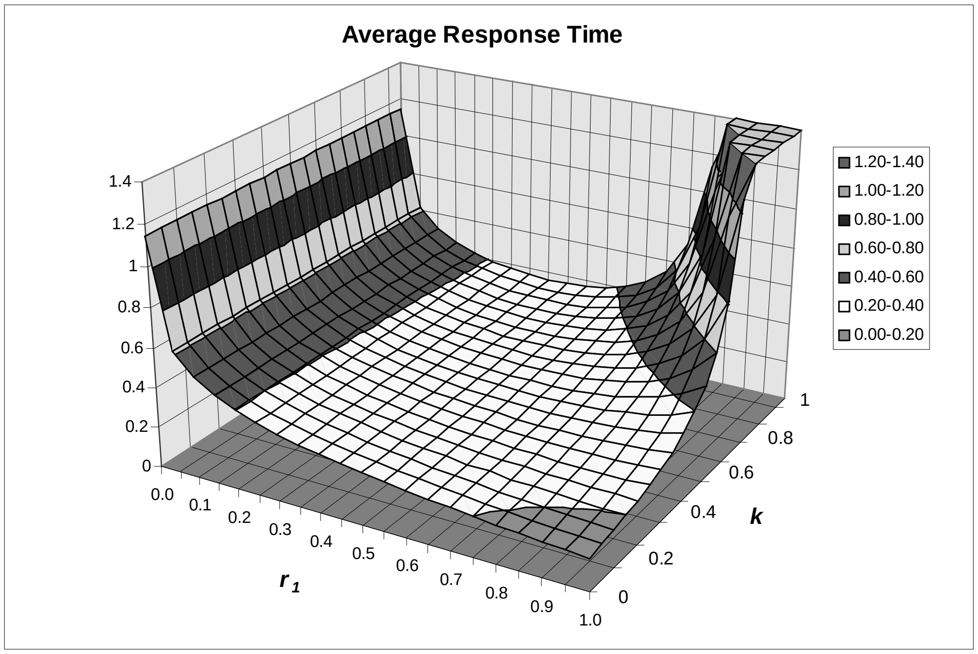

*Figure 14: Average overall response time* $\overline{R}(r_1, k)$

## 5.3 Network capacity

The results discussed so far assumed a wide area network with an effective communication bandwidth of 64 kbps. The following graphs present a sensitivity analysis on the bandwidth parameter and illustrate its significant impact on the overall system throughput. Furthermore, they demonstrate that parameter variations affect the performance values in a reasonable way, i.e. the model is stable.

Figure 15 shows the maximum system throughput and the corresponding bottleneck analysis for an effective communication bandwidth of 128 kbps. (Compare to Figure 10 and Figure 12.) As expected, the increased transmission capacity allows a higher communication limited transaction throughput $T_C$ which in turn improves the total system throughput. Specifically, the increased bandwidth makes remote access less costly in terms of communication delay, so that the benefit of local read access in case of medium or extensive replication ($r_1$ > 0.2, $r_2$ >8) is drastically outweighed by the overhead of processing update propagating transactions in the database nodes. Hence, the network is the throughput bottleneck only in case of no or low replication. In spite of the increase network capacity, Figure 15 shows that (for the given combination of parameter values) communication is not fast enough to make replication obsolete and to completely replace replication with remote access. A low degree of replication is still recommended to increase throughput, because no replication allows a maximum throughput of 375 TPS while careful replication can yield 485 TPS (for $r_1$ = 0.95, $r_2$ = 4), 534 TPS (for $r_1$ = 0.3, $r_2$ = 18), or even 577 TPS for $r_1$ = 0.15 and $r_2$ = 50. Comparing the two graphs in Figure 15 clarifies that the throughput decreases rapidly as soon as the local databases become the bottleneck. Thus, data should be replicated *cautiously* because (informally speaking) „too much replication“ can deteriorate throughput much more than „too little replication“.

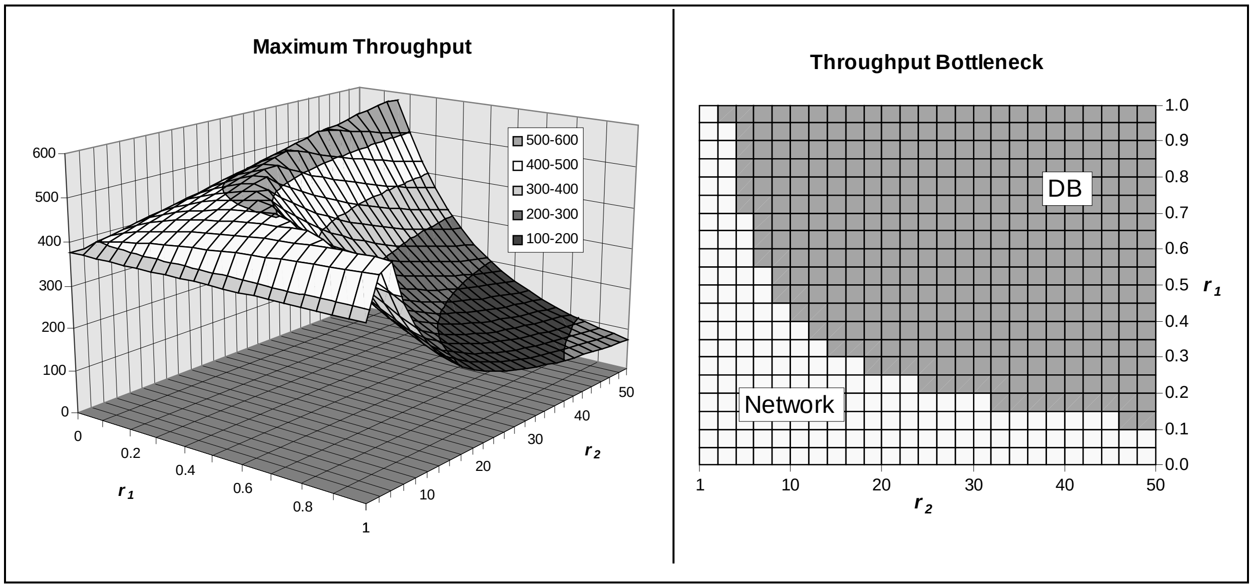

*Figure 15: Throughput and bottleneck analysis for 128 kbps network capacity*

For comparison, Figure 16 provides the throughput and bottleneck results for a lower communication bandwidth of 33.6 kbps. Such low transmission capacities are typical for wireless and mobile computing environments [Ebling et al. 94], [Imielinski, Badrinath 94], [DeSimone, Nanda 95]. In this case where remote access introduces substantial communication delay, replication is crucial to increase local data accessibility and thus to improve

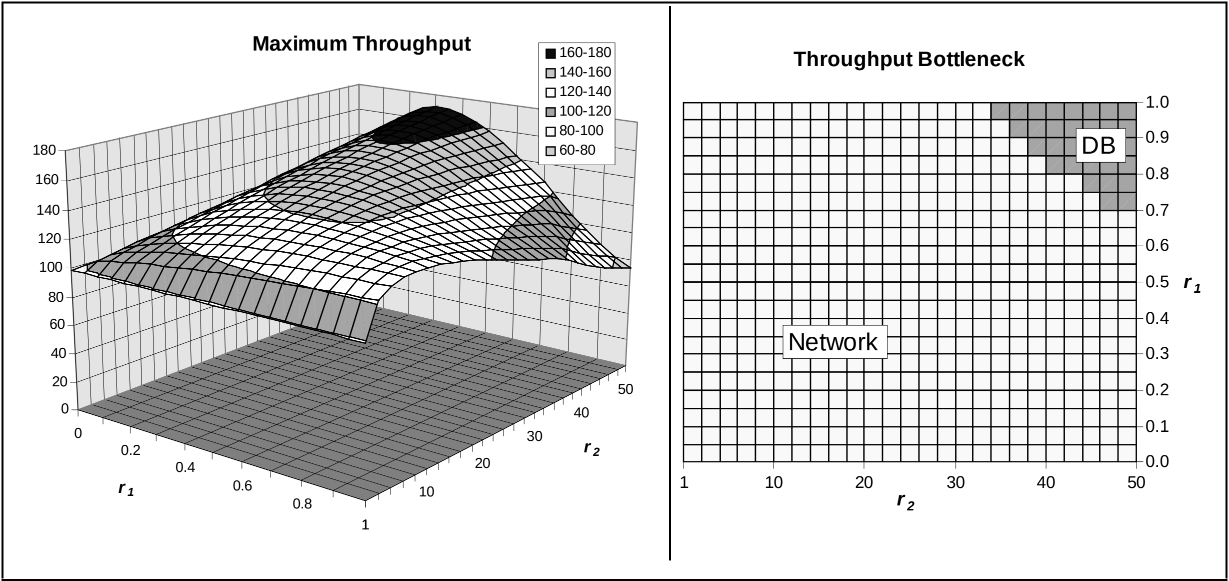

*Figure 16: Throughput and bottleneck analysis for 33.6 kbps network capacity*

response time and throughput of read transactions. As long as the workload is not heavily update intensive, replication relieves the network due to local read availability much more than it burdens the network with update propagation messages. Hence, for a low network capacity a proper extent of replication can improve performance remarkably. Almost every replication schema leads to an increase of the transaction throughput except for full or nearly full replication ($r_1 > 0.85$, $r_2 > 40$) when the database servers become the bottleneck. For any combination of the replication parameters in the ranges $0.20 \le r_1 \le 0.55$ and $28 \le r_2 \le 50$ throughput is at least 50% higher than in the non replicated case with the highest peak of 170 TPS for $r_1 = 0.30$ and $r_2 = 50$.

## 5.4 Scalability and quality of replication

Another important issue in the design of distributed database systems concerns the number of sites and the related scalability. The ideal distributed system should provide *linear scalability*, which means that the system's performance grows linearly with its size [DeWitt, Gray 92]. In the following we examine the impact which replication and the quality of a replication have on throughput and scalability. The results provide an estimation of how many sites are needed to meet given throughput requirements and how replication can help to keep the number of required sites as low as possible.

Figure 17 depicts the maximum throughput and the corresponding bottleneck analysis as a function of $r_1$ and the system size $n$. As an example, $r_2$ is set to $2n/3$ for any value of $n$, i.e. a fraction $r_1$ of the data items is replicated to 2/3 of the sites. If all data items are replicated (i.e. $r_1 = 1$), the throughput grows to about 100 TPS as the number of sites is increased to 30. Larger systems do not achieve a significantly higher throughput because without relaxed coherency high replication in large systems causes considerable update propagation overhead which prevents scalability. Reducing replication ($r_1 < 1$) gradually improves scalability. If less than 40% of the data objects are replicated (i.e $r_1 < 0.4$), far fewer update propagation transactions have to be processed so that the databases are not the bottleneck anymore and throughput can grow to over 520 TPS for $r_1 = 0.25$. However, very low or no replication does not yield the maximum throughput nor the optimum scalability because for $r_1 \le 0.15$ the local read availability decreases rapidly. This saturates the network with a large number of remote accesses which increases proportionally with the number of sites and thus hinders scalability.

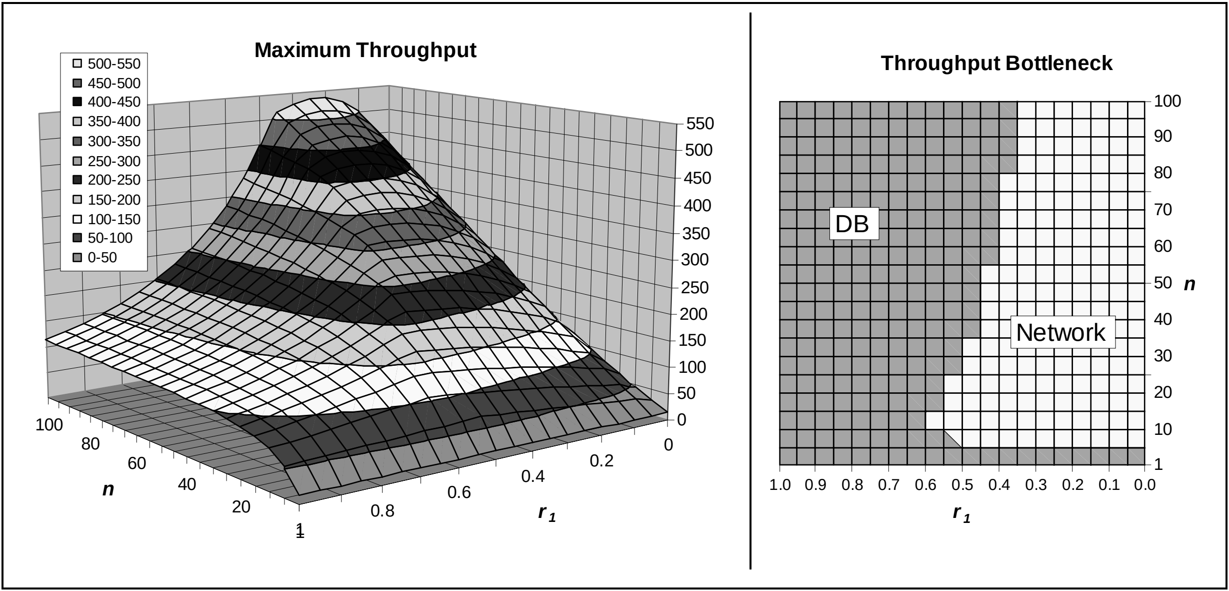

*Figure 17: Throughput and bottleneck analysis for a varying system size*

The way throughput and scalability can be improved by means of data replication depends critically on the selection of data items for replication and the placement of their copies. In Figure 18 we examine the impact of the quality of replication on the throughput by considering a „bad“ replication schema, i.e. one in which (a) not only read intensive but also a considerable amount of update intensive data items are replicated and (b) the copies are placed at randomly chosen sites rather than at remote sites where they are read particularly often. This can be modeled by the parameter settings $f\_plcmt_i = 0$ and $f\_sel_i = 0.5$ (cf. Figure 8 and Figure 9). Such a replication schema causes substantial replica maintenance overhea which drastically deteriorates throughput and scalability. Consequently, the throughput increases as replication is reduced towards 0%, which means that *no* replication is better than *„bad“* replication. However, replication might still be required to meet the availability requirements and results like clarify the trade-off involved. Some might consider 100 an unreasonable high number of database sites, but the *general* findings represented by the figures 15 and 16 do not drastically change if the maximum

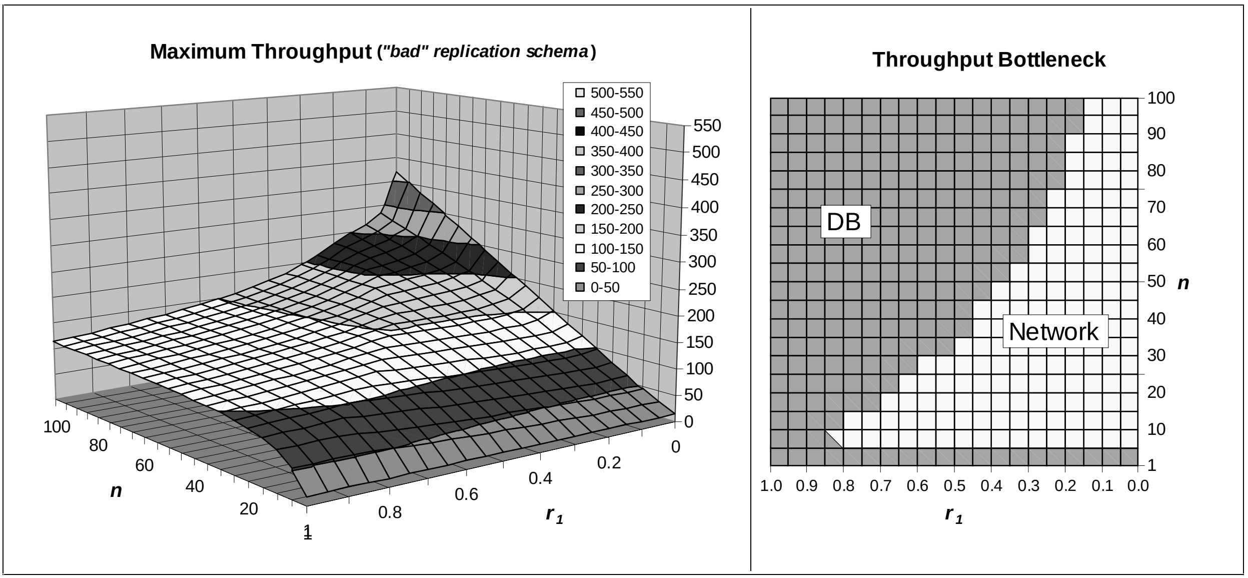

*Figure 18: Throughput and bottleneck analysis for a bad replication schema*

number of nodes is reduced to 40 or 20 sites, except for absolute values. Furthermore, depending upon the application, the number of sites may indeed range from only a few sites to several hundred sites [Anderson et al. 98], [Payne 92].

## 6 Validating Analytical Models through Measurements

Numerous parameter variations can be investigated in an analytical model as opposed to simulations or measurements. Consequently, a validation through simulations or measurements in *all* dimensions of the parameter space is hardly possible. Still, analytical performance models are in need of validation to (1) examine the effects of simplifying modeling assumptions on the accuracy of the performance estimations, and (2) to increase the acceptance of the model and its results. A validation through measurements fulfills these objectives better than simulations which rely on modeling assumptions themselves. However, measurements in a distributed database are very expensive and time consuming. In this section we describe a framework for systematic measurements in a distributed database and compare them to our analytical results. [Nicola 99] presents further validations through (1) a discrete-event simulation of a distributed database and (2) the queueing network analyzer QNAUT [Haverkort 95].

## 6.1 Existing Approaches for Measurements in Distributed Databases

Existing database benchmarks are tailored primarily to centralized databases. While some of them can also be applied to parallel database machines and database clusters, many aspects of distributed databases are not captured in the benchmark design. There is no standard benchmark for distributed databases. This may be due to a lack of agreement on practical and standard distributed database applications [Dietrich et al. 96]. Moreover, studies reporting measurements in distributed databases are *extremely* rare.

[Orji 91] proposes a benchmarking methodology for distributed databases based on three variables: (1) the number of nodes, (2) the network configuration, and (3) the data distribution. However, the experiments reported use only 2 database nodes, no replication at all, and a read-only workload. [Dietrich et al. 96] propose the $D^3S$ benchmark[7] which is an extension of the Wisconsin benchmark that matches a warehouse scenario in which items are stocked in several distributed warehouses, ordered by customers, and provided by suppliers. The measurements were conducted in a system of three database nodes where one table was fully replicated while the remaining data was not replicated at all. This environment was used to compare query execution plans generated with and without an intelligent query optimizer, but response time or throughput results are not presented. [Helal, Bhargava 95] compare the performance of a quorum protocol with that of the traditional ROWA method through measurements. They use the database schema and transaction profile of the DebitCedit benchmark, 8 database nodes, and 1-dimensional variation of partial replication. (The database schema and transaction profile of the DebitCedit benchmark was also used in the definition of the TPC-A and TPC-B benchmarks.)

## 6.2 The DR-DebitCredit

We extend the original DebitCredit benchmark to a Distributed & Replicated DebitCredit benchmark (DR-DebitCredit). The intention of this approach is not to compare different database systems but to analyze different configurations, replication schemata, and workload patterns in distributed databases, as well as their effects on the system performance. The database schema of the DR-DebitCredit follows the original DebitCredit banking schema [Gray 91]. The main extensions of the DR-DebitCredit are an additional read-only transaction in the workload, an arbitrarily adjustable selectivity of these queries, an arbitrarily adjustable ratio of read- to update and local to remote transactions, 2-dimensional variations of the replication schema, and the emulation of low bandwidth networks, as [Orji 91] recommends considering different network capacities. The update transaction is the same as in the original DebitCredit benchmark. The additional read-only transactions read *s* records from an `account` table. According to the workload ratios, a number of *m* parallel workload generators submit transactions without think time in a TPC-B like manner.

The measurement testbed consists of 10 MS-SQL Servers 6.5 running under NT 4.0 on 10 Pentium II machines with 128 MB main memory. The asynchronous primary copy replication management is realized very similar to the implementation described in [Gallersdörfer, Jarke, Nicola 99].

### Data Distribution and Replication

We consider 50 branches of a bank with 10 tellers and 100.000 accounts each. These are considered primary copies. Data distribution is defined uniformly on the level of branches. Each of the 10 database nodes holds 5

[7] $D^3S$ = Distributed Database Decision Support

branches together with the related 50 tellers and 500.000 accounts. Replication is also introduced at the level of branches and in two dimensions. Replicating a branch means replicating the branch record together with its 10 teller and 100.000 account records. In the first dimension it is possible to select 0, 1, 2, 3, 4, or 5 branches of each database node for replication. This number of replicated branches per node corresponds to the parameter $r_1$ in the analytical model and its values 0%, 20%, 40%, 60%, 80%, and 100%. In the second dimension, each branch which is selected for replication can have 1 to 10 copies. The number of copies corresponds to $r_2$ in the analytical model. A cyclic allocation scheme for secondary copies ensures that each database node holds the same amount of replicas. For full replication, the distributed database holds 500 branch records, 5000 teller records, and 50.000.000 accounts.

## 6.3 Measurement Experiments and Results

We conducted various experiments for different configurations of the distributed database and its workload, and measured the throughput and the response time. A central measurement console allows to execute series of experiments automatically. For each configuration or workload setting, measurements are performed for a complete grid of the replication parameters $r_1$ and $r_2$. The parameter $r_1$ (number of replicated branches) is increased from 0 through to 5, while $r_2$ (number of copies) assumes the values 1, 2, 4, 6, 8, 10. This yields 36 combinations. 11 of them identically represent no replication so that 26 combinations are to be measured. For each combination, the measurements are repeated 5 times to gain more confidence and detect variances. The results of five such runs typically show a deviation of only a few percent (up to 10% in rare cases). In each of the five runs up to 10.000 transactions are executed. Hence, measuring a complete $r_1$-$r_2$-grid takes more than a million transactions and 5 to 8 hours. A substantial share of this time is used up by the automatic collection and integration of measurement results and by the reconfiguration for each $r_1$-$r_2$-combination.
For each of the experiments we represented the measurement scenario in the analytical model and compared the analytical throughput and response time estimations with the measurement results. Here we present only two brief examples; further results and details are provided in [Nicola 99].

**Experiment 1** considers 10% updates, a query selectivity of 2100 bytes, and a 64kbit network. The measured throughput in the left of Figure 19 shows that the throughput increases as the degree of replication is extended in both dimensions, $r_1$ and $r_2$. For no or low replication ($r_1 \rightarrow 0$, $r_2 \rightarrow 1$) the low bandwidth capacity is the throughput bottleneck. As replication is gradually increased, more queries can be executed locally without any communication delay such that the throughput grows. The update propagation overhead outweighs the benefits of local and parallel read access and thus deteriorates the throughput only for a very high degree of replication ($r_1 \rightarrow 1$, $r_2 \rightarrow 10$). Modeling this experiment with the analytical 2RC approach produces the analytical throughput estimation in the right side of Figure 19. The shapes of the graphs show that the analytical model predicts the *tendencies* of how replication affects the throughput quite accurately. However, the increase and decrease of the throughput measurements are steeper than those in the analytically derived graph. The steeper decrease for high degrees of replication is due to the fact, that our implementation of the update propagation mechanism incurs more overhead than captured by the analytical model (e.g. logging of updates by triggers, etc.)

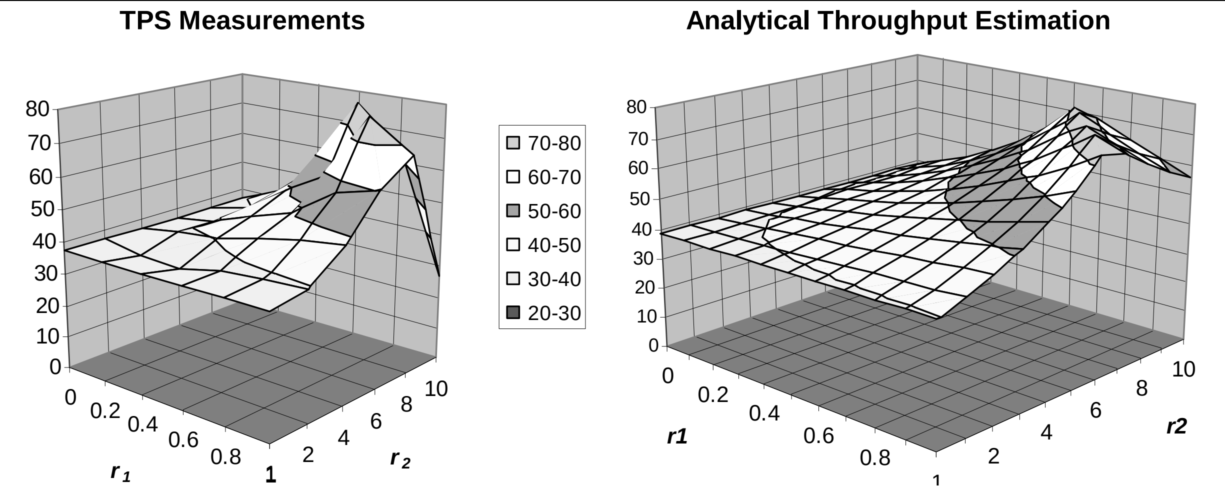

*Figure 19: Measurements and analytical results* (*64 kbit, 10% updates*)

**Experiment 2** considers 30% updates, a selectivity of 700 bytes, and a bandwidth of 64kbit. The reduced selectivity of queries in this experiment entails that the transmission of query results is less costly (in terms of communication delay) than in the previous example. Hence, the benefit of local read access is not as large as compared to cases of higher selectivity. Additionally, due to the higher percentage of updates the advantages of replication are quite small as compared to its drawbacks. Consequently, replication does not lead to performance improvements and the maximum throughput is obtained for no replication. This effect is illustrated in Figure 20. Any increase of the amount of replicated data lessens the transactions rate. If a higher degree of replication is desired, e.g. for reliability reasons, using full replication instead of no replication reduces the throughput by a factor of 4.17 from ~125 TPS to ~30 TPS. This trend is very accurately foreseen by the queueing model. Extending replication from no to full replication in the analytical model decreases the throughput from 152 to 38 TPS, i.e. by the factor 4 as seen in the measurements. These are examples, how the analytical model is a reliable tool for performance estimations of distributed and replicated databases. Similar to Figure 19, the difference between the convex measurement result and the concave shape of the analytical graph (for high degrees of replication in Figure 20) is due to higher update overhead in the implementation than assumed by the model.

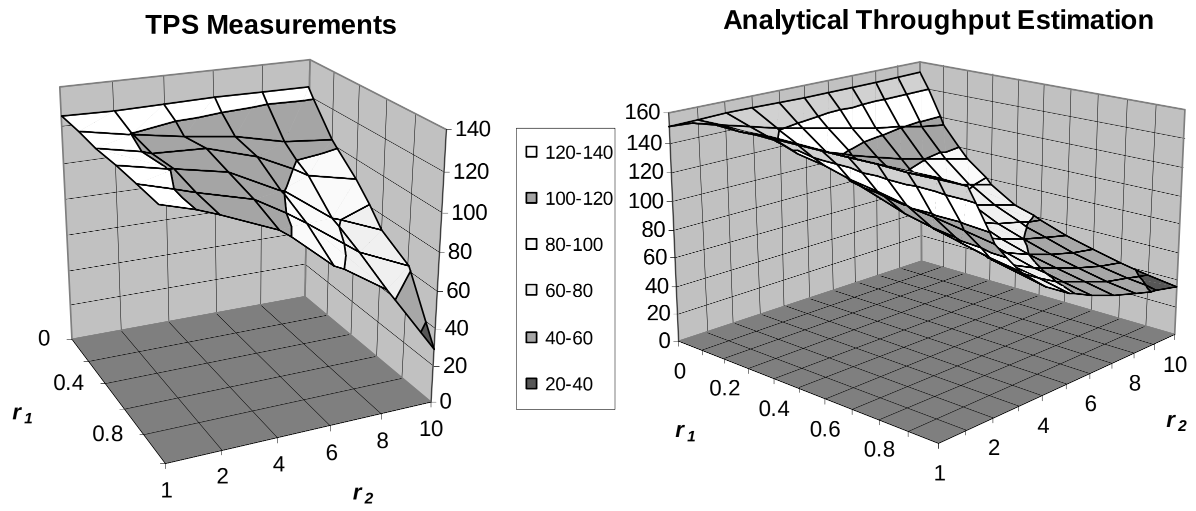

*Figure 20: Measurements and analytical results* (*64 kbit, 30% updates,* )

## 7 Summary

The main contribution of this paper is a survey on alternatives in performance modeling of distributed databases. This structured analysis of existing performance models focused on the components of a virtual modeling toolkit: (1) the general concepts to model database nodes, (2) the options in considering interdatabase communication, (3) the submodels to account for replication, (4) the assumptions concerning data access patterns, (5) the transaction processing models and (6) the interdependencies between all these aspects which are (or are not) captured in existing models.

Existing studies usually model some aspects of a real system in detail while others are either neglected or modeled in simplistic manners. Typically, the models concentrate on either the database or the communication issues while the other part is subject to simplifying and restrictive assumptions. Surprisingly, the capability of simulation models to capture more details of a real system than analytical approaches has usually been exploited to evaluate complex concurrency control protocols, but rarely to model advanced replication or communication characteristics.

As an illustrating example for the work that was surveyed, we presented the development of an analytical performance model called 2RC which integrates a 2-dimensional model of replication with an advanced communication model. 2RC captures the increasingly close interplay between replication and communication and represents a balanced model of both the database and the communication part of distributed and replicated database systems. 2RC also allows to model the quality of a replication schema as well as relaxed coherency and arbitrary transaction and communication patterns of real-world applications. The results show how partial 2D-replication schemata, which have not been evaluated previously, affect response time, throughput and scalability. This clarified that a 2D-replication model is more expressive than 1-dimensional approaches. Moreover, we demonstrated how a bottleneck analysis can reveal, for which replication schemata or system size the network or the local databases are the throughput limiting factor. Finally we defined a distributed and replicated version of the DebitCredit benchmark and discussed two out of 40 measurement results that validate the analytical model.

Concluding, we believe that continuous effort in the development of advanced performance models for replicated databases is needed so that they can keep pace with the evolution of distributed information systems. As one example, the vision of „database access anywhere anytime“ has several implications that influence the systems' performance. Data distribution and replication continues to increase, wireless communication links suffer from lower reliability and bandwidth, large database serves are replaced by clusters of workstations, etc. Such aspects must be taken into account by future models and should be integrated with existing, proven modeling concepts.

## Acknowledgement

This work was supported in part by the Deutsche Forschungsgemeinschaft under its Focused Ph.D. Programme "Information and Engineering" and under Collaborative Research Center IMROVE (SFB 476) at RWTH Aachen, and by the European Union under ESPRIT Long Term Research Project DWQ (Foundations of Data Warehouse Quality)

# References

[Abbott, Garcia-Molina 87] R. Abbott, H. Garcia-Molina: „Reliable Distributed Database Management“, Proceedings of the IEEE, 75 (5), pp. 601-620, May 1987.

[Acharya, Zdonik 93] Swarup Acharya, Stanley B. Zdonik: „*An Efficient Scheme for Dynamic Data Replication*“, Technical Report CS-93-43, Brown University, September 1993.

[Agrawal et al. 87] R. Agrawal, M.J. Carey, M. Livny: „Concurrency Control Performance Modeling: Alternatives and Implications“, ACM Transactions on Database Systems, Vol. 12, No. 4, pp. 609-654. December 1987.

[Alonso 97] G. Alonso: „*Partial Database Replication and Group Communication Primitives*“, Proceedings of the 2nd European Research Seminar on Advances in Distributed Systems (ERSADS'97), March 1997.

[Alonso et al. 90] R. Alonso, D. Barbará, H. Garcia-Molina: „*Data Caching Issues in an Information Retrieval System*“, ACM Transactions on Database Systems, Vol. 15, No. 3, pp. 359-384, 1990.

[Anderson et al. 98] T. Anderson, Y. Breitbart, H. Korth, A. Wool: „*Replication, Consistency, and Practicality: Are These Mutually Exclusive* ?“, Proceedings of the ACM SIGMOD International Conference on Management of Data, June 1998.

[Bacelli, Coffman 83] F. Bacelli, E.G. Coffmann: „*A database replication analysis using an M/M/m queue with service interruptions*“, Performance Evaluation Review, Vol. 11, No. 4, pp. 102-107, 1983.

[Banerjee et al. 94] Sujata Banerjee, Victor O K Li, Chihping Wang: „*Performance analysis of the send-on-demand: A distributed database concurrency control protocol for high-speed networks*“, Computer Communications, Vol. 17, No. 3, pp. 189-204, March 1994.

[Barbará, Garcia-Molina 82] D.Barbara, H. Garcia-Molina: „*How Expensive is Data Replication? An Example.*“, 2nd International Conference on Distributed Computing Systems, pp. 263-268, February 1982.

[Beuter, Dadam 96] T. Beuter, P. Dadam: „*Principles of replication control in distributed database systems*“ (in German), Informatik Forschung und Technik, Vol. 11, No. 4, pp. 203-212, 1996.

[Bondi, Jin 96] A.B. Bondi, V. Jin: „*A performance model of a design for a minimally replicated distributed database for database driven telecommunication services*“, Journal on Distributed and Parallel Databases, Vol. 4, No. 4, pp. 295-317, October 1996.

[Born 96] Eike Born: „*Analytical performance modelling of lock management in distributed systems*“, Distributed Systems Engineering, Vol. 3, No. 1, pp. 68-76, March 1996.

[Bouras, Spirakis 96] C.J. Bouras, P.G. Spirakis: „*Performance modeling of distributed timestamp ordering: Perfect and imperfect clocks*“, Performance Evaluation, Vol. 26, No. 2, pp. 105-130, April 1996.

[Burger et al. 97] A. Burger, V. Kumar, M.L. Hines: „*Performance of Multiversion and Distributed Two-Phase Locking Concurrency Control Mechanisms in Distributed Database*“, Information Sciences, Vol. 96, No. 1-2, pp. 129-157, January 1997.

[Cai 87] Jian Cai: „*Simulation and Evaluation of Distributed Database Systems*“, Springer Informatik Fachberichte 154, pp. 313-326, October 1987.

[Carey, Livny 88] Michael J. Carey, Miron Livny: „*Distributed Concurrency Control Performance: A Study of Algorithms, Distribution, and Replication*“, 14th Conference on Very Large Databases, pp. 13-25, 1988.

[Carey, Livny 96] Michael J. Carey, Miron Livny: „*Conflict Detection Tradeoffs for Replicated Data*“, in V.Kumar (Ed.): ‘Performance of Concurrency Control Mechanisms in Centralized Database Systems’, Prentice Hall, 1996.

[Cellary et al. 88] Wojciech Cellary, Erol Gelenbe, Tadeusz Morzy: „*Concurrency Control in Distributed Database Systems*“, Elsevier Science Publishers, Holland 1988.

[Ceri et al. 91] S. Ceri, M.A.H. Houtsma, A.M.Keller, P.Samarati: „*A Classification of Update Methods for Replicated Databases*“, Technical Report STAN-CS-91-1392, Stanford University, Oct.1991.

[Chen, Pu 92] Shu-Wie Chen, Calton Pu: „*A Structural Classification of Integrated Replica Control Mechanisms*“, Technical Report CUCS-006-92, Columbia University New York, 1992.

[Cheung et al. 92] S.Y. Cheung, M.H. Ammar, M. Ahamad: „*The Grid Protocol: A High Performance Scheme for Maintaining Replicated Data*", IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 6, pp. 582-592, December 1992.

[Ciciani et al. 90] B. Ciciani, D.M. Dias, P.S. Yu: „*Analysis of Replication in Distributed Database Systems*", IEEE Transactions on Knowledge and Data Engineering, Vol. 2, No. 2, pp. 247-261, June 1990.

[Ciciani et al. 92] B. Ciciani, D.M. Dias, P.S. Yu: „*Analysis of Concurrency-Coherency Control Protocols for Distributed Transaction Processing Systems with Regional Locality*", IEEE Transactions on Software Engineering, Vol. 18, No. 10, pp. 899-914, October 1992.

[Coan et al. 86] B.A. Coan, B.M. Oki, E.K. Kolodner: „*Limitations on Database Availability when Networks partition*", Proceedings of the 5th ACM Symposium on Principles of Distributed Computing, pp. 187-194, August 1986.

[Coffmann et al. 81] E.G. Coffmann, Erol Gelenbe, Brigitte Plateau: „*Optimization of the number of copies in a distributed system*", IEEE Transactions on Software Engineering, Vol. 7, pp. 78-84, January 1981.

[Davidson, Garcia-Molina 85] Susan B. Davidson, Hector Garcia-Molina: „*Consistency in Partitioned Networks*", Computing Surveys 17(3), pp. 341-370, 1985.

[DeSimone, Nanda 95] Antonie DeSimone, Sanjiv Nanda: „*Wireless Data: Systems, Standards, Services*", Wireless Networks, Vol. 1, No. 3, pp. 241-253, 1995.

[DeWitt, Gray 92]. D. DeWitt, J. Gray: „*Parallel Database Systems: The Future of High Performance Database systems*", Communications of the ACM, Vol. 35, No. 6, June 1992, pp. 85-98.

[Dias et al. 87] D.M. Dias, P.S. Yu, B.T. Bennett: „*On centralized versus geographically distributed database systems*", 7th International Conference on Distributed Computing Systems, pp. 64-71, September 1987.

[Dietrich, et al. 96] Suzanne Dietrich and Changguan Fan, E. Cortes-Rello: „*An Application of Fragmentation Transparency in a Distributed Database System: A Case Study*"; Journal on Systems and Software, Vol. 35; pp. 185-197, 1996.

[Ebling et al. 94] M. Ebling, L. Mummert, D. Steere: „*Overcoming the Network Bottleneck in Mobile Computing*", Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, 1994.

[Gallersdörfer, Jarke, Nicola 99] R. Gallersdörfer, M. Jarke, Matthias Nicola: "*The ADR Replication Manager*", International Journal of Cooperative Information Systems (IJCS), Vol. 8, No. 1, pp. 15-45, March 1999.

[Gallersdörfer, Nicola 95] Rainer Gallersdörfer, Matthias Nicola: „*Improving Performance in Replicated Databases through Relaxed Coherency*", 21th Conference on Very Large Databases, pp. 445-456, September 1995.

[Garcia-Molina 82] H. Garcia-Molina: „*Performance of the Update Algorithms for Replicated Data in a Distributed Database*", Ph.D. Dissertation, revised, Computer Science Dept., Stanford University, North Holland, 1982.

[Garcia-Molina, Wiederhold 82] Hector Garcia-Molina, Gio Wiederhold: „*Read-Only Transactions in a Distributed Database*", ACM Transactions on Database Systems, Vol. 7, No. 2, June 1982, pp 209-234.

[Gray 91] Jim Gray: „*The Benchmark Handbook for Database and Transaction Processing Systems*", Morgan Kaufmann, 1991.

[Gray et al. 96] Jim Gray, P. Helland, P. O'Neil, D. Shasha: „*The dangers of replication and a solution*", SIGMOD Record, Vol. 25, No. 2, pp. 173-182, June 1996.

[Gray, Reuter 93] Jim Gray, Andreas Reuter: „*Transaction Processing: Concepts and Techniques*", Morgan Kaufmann, 1993.

[Gross, Harris 85] D. Gross, C.M. Harris: „*Fundamentals of Queueing Theory*", John Wiley & Sons, 1985.

[Haverkort 95] Boudewijn R. Haverkort: „*Approximate Analysis of Networks of PH/PH/1/K Queues: Theory & Tool Support*", Quantitative Evaluation of Computing and Communication Systems, LNCS 977, pp. 239-253, 1995.

[Helal et al. 96] A.A. Helal, A.A. Heddaya, B.B. Bhargava: „*Replication Techniques in Distributed Systems*", Kluwer Academic Publishers, 1996.

[Helal, Bhargava 95] A. Helal, B. Bhargava: „*Performance Evaluation of The Quorum Consensus Replication Method*“, International Computer Performance and Dependability Symposium, pp. 165-172, 1995.

[Hung, Lam 92] S.L. Hung, K.Y. Lam: „*Performance Study of 2 Phase Locking in Distributed Database System with Mixed Transaction Classes*“, Proceedings of the 24th Annual Computer Simulation Conference, pp. 289-293, 1992.

[Hwang et al. 96] S.Y. Hwang, K.S. Lee, Y.H. Chin: „*Data Replication in a Distributed System: A Performance Study*“, 7th International Conference on Database and Expert Systems Applications, pp. 708-717, 1996.

[Imielinski, Badrinath 94] Tomasz Imielinski, B.R. Badrinath: „*Mobile wireless computing: Challenges in Data Management*“, Technical Report DCS-TR, Department of Computer Science, Rutgers University, 1994.

[Jain 91] R. Jain: „*The Art of Computer Systems Performance Analysis - Techniques for Experiment Design, Measurement, Simulation and Modeling*“, John Wiley & Sons, 1991.

[Jenq et al. 88] Bao Chyuan Jenq, W.H. Kohler, D. Towsley: „*A Queueing Network Model for a Distributed Database Testbed System*“, IEEE Transactions on Software Engineering, Vol. 14, No. 7, pp. 908-921, July 1988.

[Jenq et al. 89] B.C. Jenq, B.C. Twitchell, T.W. Keller: „*Locking Performance in a Shared Nothing Parallel Database Machine*“, IEEE Transactions on Knowledge and Data Engineering, Vol. 1, No. 4, pp. 530-543, Dec. 1989.

[Kähler, Risnes 87] Bo Kähler, Oddvar Risnes: „*Extending Logging for Database Snapshot Refresh*“, Proceedings of the 13th International Conference on Very Large Databases, pp 389-398, 1987.

[Kemme, Alonso 98] B. Kemme, G. Alonso: „*A Suite of Database Replication Protocols based on Group Communication Primitives*“, 18th International Conference on Distributed Computing Systems, May 1998.

[Keum et al. 95] C.S. Keum, E.K. Hong, W.Y. Kim, K.Y. Whang: „*Performance Evaluation of Replica Control Algorithms in a Locally Distributed Database System*“, Proceedings of the 4th International Conference on Database Systems for Advanced Database Applications, pp. 388-396, April 1995.

[Kleinrock 75] Leonard Kleinrock: „*Queueing Systems, Volume I: Theory*“, John Wiley & Sons, 1975.

[Kuang, Mukkamala 91] Yinhong Kuang, Ravi Mukkamala: „*Performance Analysis of Static Locking in Replicated Distributed Database Systems*“, IEEE Proceedings of SOUTHEASTCON'91, Vol. 2, pp. 698-701, 1991.

[Kumar, Segev 93] A. Kumar, A. Segev: „*Cost and Availability Tradeoffs in Replicated Data Concurrency Control*“, ACM Transactions on Database Systems, Vol. 18, No. 1, pp. 102-131, March 1993.

[Leung 97] Kin K. Leung: „*An Update Algorithm for Replicated Signaling Databases in Wireless and Advanced Intelligent Networks*“, IEEE Transactions on Computers, Vol. 46, No. 3, pp. 362-367, March 1997.

[Liang, Tripathi 96] D. Liang, S. K. Tripathi: „*Performance Analysis of Long-Lived Transaction Processing Systems with Rollbacks and Aborts*“, IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 5, pp. 802-815, 1996.

[Little, McCue 94] M.C. Little, D.L. McCue: „*The Replica Management System: A Scheme for Flexible and Dynamic Replication*“, Proceedings of the 2nd Workshop on Configurable Distributed Systems, March 1994.

[Mariasoosai, Singhal 90] W. Mariasoosai, M Singhal: „*A Concurrency Control Algorithm for Replicated Database Systems*“, Proceedings of the ISMM International Conference, New York, pp. 143-147, October 1990.

[Mc Dermott, Mukkamala 94] John Mc Dermott, Ravi Mukkamala: „*Performance Analysis of Transaction Management Algorithms for the SINTRA Replicated Architecture Database Systems*“, IFIP Transactions (Computer Science and Technology), Vol. A-47, pp. 215-234, 1994.

[Miyanishi et al. 96] Y. Miyanishi, K. Nakamura, F. Sato, T. Watanabe, T. Mizuno: „*An Analysis of data updating performance in distributed systems and a proposal of a data updating algorithm*“, Proceedings of the 10th International Conference on Networking, pp. 53-59, 1996.

[Mukkamala 87] Ravi Mukkamala: „*Design of partially replicated distributed database systems*“, Technical Report TR-87-04, University of Iowa, Department of Computer Science, 1987.

[Mukkamala 89] Ravi Mukkamala: „*Measuring the Effects of Data Distribution Models on Performance Evaluation of Distributed Database Systems*“, IEEE Transactions on Knowledge and Data Engineering, Vol.1, No.4, pp. 494-507, December 1989.

[Mukkamala 92] Ravi Mukkamala: „*Measuring the Effects of Distributed Database Models on Transaction Availability Measures*“, Performance Evaluation, Vol.14, pp. 1-20, 1992.

[Mukkamala, Bruell 90] R. Mukkamala, S.C. Bruell: „*Efficient Schemes to Evaluate Transaction Performance in Distributed Database Systems*“, The Computer Journal, Vol. 33, No. 1, pp. 79-89, February 1990.

[Nelson, Iyer 85] Randolph D. Nelson, Balakrishna R. Iyer: „*Analysis of a Replicated Database*“, Performance Evaluation, Vol. 5, pp. 133-148, 1985.

[Nicola 99] Matthias Nicola: „*Performance Evaluation of Distributed, Replicated, and Wireless Information Systems*”, Doctoral Thesis AIB-99-10, Technical University of Aachen, Department Informatik V, October 1999.

[Noe, Andreassian 87] J.D. Noe, A. Andreassian: „*Effectiveness of Replication in Distributed Computer Networks*“, Proceedings of the 7th International Conference on Distributed Computing Systems, pp. 508-513, 1987.

[Orji 91] Cyril U. Orji: „*A Methodology for Benchmarking Distributed Database Management Systems*”, Proceedings of the 7th International Conference on Data Engineering, pp. 612-619, 1991.

[Pacitti, Simon 97] Esther Pacitti, Eric Simon: „*Update Propagation Strategies to Improve Freshness of Data in Lazy Master Schemes*“, Technical Report No. 3233, INRIA Rocquencourt, France, August 1997.

[Park et al. 94] S.O. Park, C.R. Carlson, T.M. Chen: „*Performance Analysis of Distributed Database System in Inter-Network Environments: A Queueing Analytic Approach*“, Proceedings of the International Conference on Modeling and Simulation, pp. 202-205, 1994.

[Payne 92] Alison Payne: „*Designing the Databases of the Intelligent Network*“, Proceedings of the 8th International Conference on Software Engineering for Telecommunication Systems and Services, pp. 37-41, 1992.

[Raghuram et al. 92] A. Raghuram, T.W. Morgan, B. Rajaraman, Y. Ronen: „*Approximation for the mean value performance of locking algorithms for distributed database systems*“, Annals of Operations Research, Vol. 36, No. 1-4, pp. 299-346, May 1992.

[Ren et al. 96] J.F. Ren, Y. Takahashi, T. Hasegawa: „*Analysis of impact of network delay on multiversion timestamp algorithms in DDBS*“, Performance Evaluation, pp. 21-50, July 1996.

[Saha et al. 96] D. Saha, S. Rangarajan, S. K. Tripathi: „*An Analysis of the Average Message Overhead in Replica Control Protocols*“, IEEE Transactions on Parallel and Distributed Systems, Vol. 7, No. 10, pp. 1026-1034, Oct. 1996.

[Shah, Ghosal 90] A. Shah, Dipak Ghosal: „*A Stochastic Analysis of the Performance of Distributed Databases with Site and Link Failures*“, Department of Computer Science, Cornell University Ney York, Technical Report 90-1072, 1990.

[Shah, Marzullo 89] A. Shah, K. Marzullo: „*Trade-Offs Between Replication and Availability in Distributed Databases*“, Department of Computer Science, Cornell University New York, Technical Report 89-1065, 1989.

[Sheth et al. 85] A.P. Sheth, A. Singhal, M.T. Liu: „*An Analysis of the Effect of Network Parameters on the Performance of Distributed Database Systems*“, IEEE Transactions on Software Engineering, Vol. 11, No. 10, pp. 1174-1184, October 1985.

[Shyu, Li 90] A.C. Shyu, V.O.K. Li: „*Performance Analysis of Static Locking in Distributed Database Systems*“, IEEE Transactions on Computers, Vol. 39, No. 6, pp. 741-751, June 1990.

[Silberschatz, Galvin 94] Abraham Silberschatz, Peter B. Galvin: „*Operating System Concepts*“ , Addison Wesley, 4th Edition 1994.

[Simha, Majumdar 97] R. Simha, A. Majumdar: „*An Urn Model with Application to Database Performance Evaluation*“, Computers & Operations Research, Vol. 24, No. 4, pp. 289-300, April 1997.

[Singhal 86] Mukesh Singhal: „*Concurrency Control Algorithms and their Performance for Replicated Database Systems*“, Ph.D. dissertation, Department of Computer Science, University of Maryland, 1986.

[Singhal 90] Mukesh Singhal: „*Update Transport: A New Technique for Update Synchronization in Replicated Database Systems*“, IEEE Transactions on Software Engineering, Vol. 16, pp. 1325-1336. 1990.

[Son, Haghighi 90] S.H. Son, N. Haghighi: „*Performance Evaluation of Multiversion Database Systems*“, Proceedings of the 6th International Conference on Data Engineering, pp. 129-136, February 1990.

[Son, Kouloumbis 91] S.H. Son, S. Kouloumbis: „*Performance Evaluation of Replication Control Algorithms for Distributed Database Systems*“, Technical Report, CS-TR-91-11, University of Virginia, 1991.

[Son, Zhang 95] S.H. Son, F. Zhang: „*Real-Time Replication Control for Distributed Database Systems: Algorithms and Their Performance*“, 4th International Conference on Database Systems for Advanced Database Applications, pp. 214-221, April 1995.

[Stonebraker 79] Michael Stonebraker: „*Concurrency Control and Consistency of Multiple Copies of Data in Distributed Ingres*“, IEEE Transactions on Software Engineering, 5(3):188-194, 1979.

[Tai, Meyer 96] A.T. Tai, J.F. Meyer: „*Performability Management in Distributed Satabase Systems: An Adaptive Concurency Control Protocol*“, Proceedings of the 4th International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 212-216, 1996.

[Tay et al. 85] Y.C. Tay, N. Goodman, R. Suri: „*Locking Performance in Centralized Databases*“, ACM Transactions on Database Systems, Vol. 10, No. 4, pp. 415-462, December 1985.

[Thanos et al. 88] C. Thanos, E. Bertino, C. Carlesi: „*The Effect of Two-Phase Locking on the Performance of a Distributed Database Management System*“, Performance Evaluation, Vol. 8, No. 2, pp. 129-157, 1988.

[Thomasian 93] Alexander Thomasian: „*Determining the Number of Remote Sites Accessed in Distributed Transaction Processing*“, IEEE Transactions on Parallel and Distributed Systems, Vol. 4, No. 1, pp. 99-103, January 1993.

[Thomasian 96] A. Thomasian: „*Database Concurrency Control: Methods, Performance and Analysis*“, Kluwer Academic Publishers, 1996.

[Thomasian 98] A. Thomasian: „*Concurrency Control: Methods, Performance and Analysis*“, ACM Computing Surveys, Vol. 30, No. 1, pp. 70-119, March 1998.

[Triantafilliou, Taylor 95] P. Triantafilliou, D.J. Taylor: „*Achieving Efficiency and Availability in Distributed Systems*”, IEEE Transactions on Software Engineering, Vol. 21, No. 1, Jan. 1995.

[Triantafilliou, Taylor 96] Triantafilliou, D.J. Taylor: „*VELOS – A New Approach for Efficiently Achieving High Availability*”, IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 2, April 1996.

[Triantafillou 91] P. Triantafillou: „*Employing Replication to Achieve High Availability and Efficiency in Distributed Systems*“, Research Report CS-91-28, University of Waterloo, 1991.

[Triantafillou 96] Peter Triantafillou: „*Independent Recovery in Large-Scale Distributed Systems*“, IEEE Transactions on Software Engineering, Vol. 22, No. 11, pp. 812-826, November 1996.

[Ulusoy 94] Ö. Ulusoy: „*Processing Real-Time Transactions in a Replicated Database System*“, Journal on Distributed and Parallel Databases, Vol. 2, No. 4, pp. 405-436, October 1994.

[Ulusoy, Belford 92] Ö. Ulusoy, G.G. Belford: „*A Simulation Model for Distributed Real-Time Databases*“, Proceedings of the 25th Annual Simulation Symposium (IEEE), pp. 232-240, April 1992.

[Wolfson et al. 97] Ouri Wolfson, S. Jajodia, Y. Huang: „*An Adaptive Data Replication Algorithm*“, ACM Transactions on Database Systems, Vol. 22, No. 2, pp. 255-314, June 1997.

[Wu 93] Chienwen Wu: „*Replica Control Protocols that guarantee High Availability and Low Access Cost*“, Ph.D. Dissertation, University of Illinois, 1993.

[Yu et al. 93] Philip S. Yu, Danial M. Dias, Stephen S. Lavenberg: „*On the Analytical Modelling of Database Concurrency Control*“, Journal of the ACM, Vol. 40, No. 4, pp. 831-872, September 1993.

[Zhang, Hsu 96] B. Zhang, M. Hsu: „*Modeling Performance Impact of Hot Spots*“, In Vijay Kumar (Ed.): Performance of Concurrency Control Mechanisms in Centralized Database Systems, Prentice Hall, pp. 148-165, 1996.

[Zhou et al. 96] Shaoyu Zhou, M.H. Williams, H. Taylor: „*Practical Throughput Estimation for Parallel Databases*“, Software Engineering Journal, Vol. 11, No. 4, pp. 255-263, July 1996.