1. Record Nr. UNINA9910299240803321

Autore	Xu Chen
Titolo	Quality-aware Scheduling for Key-value Data Stores / / by Chen Xu, Aoying Zhou
Pubbl/distr/stampa	Berlin, Heidelberg : , : Springer Berlin Heidelberg : , : Imprint : Springer, , 2015
ISBN	3-662-47306-2
Edizione	[1st ed. 2015.]
Descrizione fisica	1 online resource (102 p.)
Collana	SpringerBriefs in Computer Science, , 2191-5768
Disciplina	005.7565
Soggetti	Database management Application software Operating systems (Computers) Database Management Information Systems Applications (incl. Internet) Operating Systems
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Description based upon print version of record.
Nota di bibliografia	Includes bibliographical references at the end of each chapters.
Nota di contenuto	Preface; Acknowledgments; Contents; 1 Introduction; 1.1 Application Scenarios; 1.2 The Research Significance and Challenges; 1.3 Implementation Framework; 1.4 Overview of the Book; References; 2 Literature and Research Review; 2.1 Metrics for Quality-Aware Scheduling; 2.1.1 QoS Metrics; 2.1.2 QoD Metrics; 2.2 Quality-Aware Scheduling in Data Management System; 2.2.1 Quality-Aware Scheduling in RTDBMS; 2.2.2 Quality-Aware Scheduling in DSMS; 2.2.3 Quality-Aware Scheduling in RDBMS; 2.2.4 Quality-Aware Scheduling in Key-Value Stores; 2.3 Summary; References; 3 Problem Overview 3.1 Background Knowledge3.1.1 Data Organization; 3.1.2 Data Replication and Consistency; 3.1.3 User Queries; 3.1.4 System Updates: State-Transfer Versus Operation-Transfer; 3.2 Problem Statement; 3.2.1 QoS Penalty; 3.2.2 QoD Penalty; 3.2.3 Combined Penalty; 3.3 Summary; References; 4 Scheduling for State-Transfer Updates; 4.1 On-Demand (OD) Mechanism; 4.1.1 WSJF-OD; 4.2 Hybrid On-Demand (HOD) Mechanism; 4.2.1 WSJF-HOD; 4.3 Freshness/Tardiness (FIT) Mechanism; 4.3.1 WSJF-FIT; 4.4 Adaptive Freshness/Tardiness (AFIT)

Mechanism; 4.4.1 Query Routing; 4.4.2 Query Selection; 4.4.3 WSJF-AFIT
4.5 Popularity-Aware Mechanism4.5.1 Populairty-Aware WSJF-OD; 4.5.2 Populairty-Aware WSJF-HOD; 4.5.3 Popularity-Aware WSJF-FIT; 4.5.4 Popularity-Aware WSJF-AFIT; 4.6 Experimental Study; 4.6.1 Baseline Policies; 4.6.2 Parameter Setting; 4.6.3 Impact of Query Arrival Rate; 4.6.4 Impact of Update Cost; 4.6.5 Impact of Different QoS and QoD Preferences; 4.6.6 Impact of Popularity; 4.7 Summary; References; 5 Scheduling for Operation-Transfer Updates; 5.1 Hybrid On-Demand (HOD) Mechanism; 5.1.1 WSJF-HOD; 5.2 Freshness/Tardiness (FIT) Mechanism; 5.2.1 WSJF-FIT; 5.3 Popularity-Aware Mechanism
5.3.1 Popularity-Aware WSJF-HOD5.3.2 Popularity-Aware WSJF-FIT; 5.4 Experimental Study; 5.4.1 Parameter Setting; 5.4.2 Impact of Update Arrival Rate; 5.4.3 Impact of Popularity and Approximation; 5.5 Summary; References; 6 AQUAS: A Quality-Aware Scheduler; 6.1 System Overview; 6.1.1 System Goals; 6.1.2 System Design; 6.2 System Performance; 6.2.1 Benchmark; 6.2.2 Evaluation Result; 6.3 A Demonstration on MicroBlogging Application; 6.3.1 Timeline Queries in AQUAS; 6.3.2 A Case Study; 6.4 Summary; References; 7 Conclusion and Future Work; 7.1 Conclusion; 7.2 Future Work; References

Sommario/riassunto

Key-value stores, which are commonly used as data platform for various web applications, provide a distributed solution for cloud computing and big data management. In modern web applications, user experience satisfaction determines their success . In real application, different web queries or users produce different expectations in terms of query latency (i.e., Quality of Service (QoS)) and data freshness (i.e., Quality of Data (QoD)). Hence, the question of how to optimize QoS and QoD by scheduling queries and updates in key-value stores has become an essential research issue. This book comprehensively illustrates quality-ware scheduling in key-value stores. In addition, it provides scheduling strategies and a prototype framework for a quality-aware scheduler, as well as a demonstration of online applications. The book offers a rich blend of theory and practice, making it suitable for students, researchers and practitioners interested in distributed systems, NoSQL key-value stores and scheduling.