Resource Allocation in Vehicular Cloud Computing

Puya Ghazizadeh
Old Dominion University

RESOURCE ALLOCATION IN VEHICULAR CLOUD COMPUTING

by

Puya Ghazizadeh
B.S. September 2005, University of Kurdistan, Iran
M.S. April 2009, Science and Researche Branch of Azad University, Iran

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
August 2014

Approved by:

Ravi Mukkamala (Director)

Stephan Olariu (Director)

Irwin Levinstein (Member)

Michele Weigle (Member)

Jin Wang (Member)

ABSTRACT

RESOURCE ALLOCATION IN VEHICULAR CLOUD COMPUTING

Puya Ghazizadeh
Old Dominion University, 2014
Director: Dr. Ravi Mukkamala and Dr. Stephan Olariu

Recently, we have witnessed the emergence of *Cloud Computing*, a paradigm shift adopted by information technology (IT) companies with a large installed infrastructure base that often goes under-utilized. The unmistakable appeal of cloud computing is that it provides scalable access to computing resources and to a multitude of IT services. Cloud computing and cloud IT services have seen and continue to see a phenomenal adoption rate around the world.

Recently, Professor Olariu and his coworkers through series of research introduced a new concept, Vehicular Cloud Computing. A Vehicular Cloud (VC) is a network of vehicles in a parking lot that can provide computation services to users. In this model each vehicle is a computation node. Some of the applications of a VC include a datacenter at the airport, a data cloud in a parking lot, and a datacenter at the mall.

The defining difference between vehicular and conventional clouds lies in the distributed ownership and, consequently, the unpredictable availability of computational resources. As cars enter and leave the parking lot, new computational resources become available while others depart, creating a dynamic environment where the task of efficiently assigning jobs to cars becomes very challenging.

Our main contribution is a number of scheduling and fault-tolerant job assignment strategies, based on redundancy, that mitigate the effect of resource volatility in vehicular clouds. We offer a theoretical analysis of the expected job completion time in the case where cars do not leave during a checkpoint operation and also in the case where cars may leave while checkpointing is in progress, leading to system failure. A comprehensive set of simulations have shown that our theoretical predictions are accurate.

We considered two different environments for scheduling strategy: deterministic and stochastic. In a deterministic environment the arrival and departure of cars are known. This scenario is for environments like universities where employees should be present at work with known schedules and the university rents out its employees' cars as computation nodes to provide services as a vehicular cloud. We presented a scheduling model for a vehicular cloud based on mixed integer linear programming. This work investigates a job scheduling problem involving non-preemptive tasks with known processing time where job migration is allowed. Assigning a job to resources is valid if the job has been executed fully and continuously (no interruption). A job cannot be executed in parallel. In our approach, the determination of an optimal job schedule can be formulated as maximizing the utilization of VC and minimizing the number of job migrations. Utilization can be calculated as a time period that vehicles have been used as computation resources. For dynamic environment in terms of resource availability, we presented a stochastic model for job assignment. We proposed to make job assignment in VC fault tolerant by using a variant of the checkpointing strategy. Rather than saving the state of the computation, at regular times, the state of the computation is only recorded as needed. Also, since we do not assume a central server that stores checkpointed images, like conventional cloud providers do, in our strategy checkpointing is performed by a car and the resulting image is stored by the car itself. Once the car leaves, the image is lost. We consider two scenarios: in the first one, cars do not leave during checkpointing; in the second one, cars may leave during checkpointing, leading to system failure. Our main contribution is to offer theoretical predictions of the job execution time in both scenarios mentioned above. A comprehensive set of simulations have shown that our theoretical predictions are accurate.

ACKNOWLEDGEMENTS

I would like to express my gratitude to God for providing me the blessings to complete this work. Also, I could not have completed this thesis without the help of many individuals to whom I would like to express my appreciation. First and foremost, I would like to thank my advisors, Professor Ravi Mukkamala and Professor Stephan Olariu, who have put a great deal of time and effort into the guidance of this work. They supported me scientifically and guided me as their son. Their office was always open to help me solve any technical or personal problem I had during the program. Next, I would like to thank my PhD committee Drs. Irwin Levinstein, Michele Weigle and Jin Wang. Their expertise, thorough reviewing, continuous support, and valuable suggestions have led to a greatly improved dissertation. I am also grateful to my family for their encouragement and support. Finally, special thanks to my father and mother for their understanding and patience during the time I spend completing this work.

TABLE OF CONTENTS

Page

LIST OF TABLES viii

LIST OF FIGURES xiii

Chapter

1. INTRODUCTION 1
   - 1.1 MOTIVATION 1
   - 1.2 OBJECTIVES AND GOALS 2
   - 1.3 CONTRIBUTIONS 2
   - 1.4 ROADMAP 5

2. RELATED WORK 6
   - 2.1 CLOUD COMPUTING 6
   - 2.2 VEHICULAR NETWORK 17
   - 2.3 CONNECTED VEHICLES 19
   - 2.4 SMART CARS 21
   - 2.5 VOLUNTEER COMPUTING 21
   - 2.6 VEHICULAR CLOUDS 22
   - 2.7 SECURITY AND PRIVACY 23
   - 2.8 EXAMPLES OF VEHICULAR CLOUDS 24
   - 2.9 SUMMARY 26

3. JOB SCHEDULING IN DETERMINISTIC ENVIRONMENT 27
   - 3.1 INTRODUCTION 27
   - 3.2 RELATED WORK 27
   - 3.3 OUR APPROACH 29
   - 3.4 IMPLEMENTATION AND EXPERIMENT RESULTS 33
   - 3.5 CONCLUSION 33

4. FAULT-TOLERANT JOB ASSIGNMENT 36
   - 4.1 INTRODUCTION AND MOTIVATION 36
   - 4.2 RELEVANT PREVIOUS WORK ON VEHICULAR CLOUDS 37
   - 4.3 OUR CONTRIBUTIONS 37
   - 4.4 THE JOB ASSIGNMENT STRATEGY J2 38
   - 4.5 WHEN CHECKPOINTS DO NOT FAIL 38
   - 4.6 THE GENERAL CASE 41
   - 4.7 SIMULATION RESULTS 46
   - 4.8 CONCLUDING REMARKS 48

5. MEAN TIME TO FAILURE 55
5.1 INTRODUCTION AND MOTIVATION 55
5.2 RELEVANT PREVIOUS WORK ON VEHICULAR CLOUDS 55
5.3 OUR CONTRIBUTIONS 55
5.4 THE JOB ASSIGNMENT STRATEGIES 56
5.5 BASICS 57
5.6 THE MTTF FOR THE J2 STRATEGY 57
5.7 THE MTTF FOR THE J3 STRATEGY 61
5.8 SIMULATION RESULTS 64
5.9 CONCLUDING REMARKS 70

6. ONE CAR APPROACH J1 71
6.1 OUR CONTRIBUTIONS 71
6.2 SIMULATION 72
6.3 COMPARING J1 AND J2 STRATEGIES 76
6.4 CONCLUDING REMARKS 78

7. COMPARISON OF JOB ASSIGNMENT STRATEGIES 86
7.1 JOB EXECUTION TIME J3 86
7.2 COMPARISON OF JOB COMPLETION TIME 96
7.3 COMPARISON OF MEAN TIME TO FAILURE 104
7.4 PROBABILITY OF CHECKPOINT FAILURE 105
7.5 NUMBER OF CHECKPOINTS 106

8. CONCLUDING REMARKS AND DIRECTIONS FOR FUTURE WORK 119
8.1 CONCLUDING REMARKS 119
8.2 DIRECTION FOR FUTURE WORK 121

APPENDIX 125

VITA 136

LIST OF TABLES

Table Page

1. Different Types of Operating System. 16

2. Different Types of Application. 16

LIST OF FIGURES

Figure Page

1. Google Apps as Software as a Service 7
2. Cloud Services 9
3. Types of Cloud 11
4. Queue Service 12
5. Hosted Virtualization. 13
6. Hypervisor Virtualization. 14
7. Application Virtualization 15
8. GM MyLink Telematics System: Photo Credit (http://gm-volt.com/) 20
9. Microsoft-based Ford Sync: Photo Credit (http://www.ford.com/) 20
10. Google Car: Photo Credit (http://techcrunch.com/) 21
11. Scheduling 0-1 Model (Jobs' Specification) 33
12. Parking Occupancy Model 34
13. Scheduling 0-1 Model (Result) 34
14. Illustrating the job assignment and checkpointing strategy. 39
15. Markov Process 42
16. Illustrating the proof of Lemma 4.6.1 42
17. Scenario 1. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed 48
18. Scenario 2. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed 49
19. Scenario 3. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed 49
20. Scenario 4. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed 50

21. Scenario 5. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed 50

22. Scenario 6. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed 51

23. Scenario 7. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed 51

24. Scenario 8. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed 52

25. Scenario 9. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed 52

26. Scenario 1. Illustrating Job Completion Time J2 Strategy Checkpoint Failure Allowed 53

27. Scenario 2. Illustrating Job Completion Time J2 Strategy Checkpoint Failure Allowed 53

28. Illustrating the J2 job assignment strategy. 56

29. Illustrating the Markov process corresponding to strategy J2. 58

30. Illustrating the proof of Lemma 5.6.1. 59

31. Illustrating the Markov process corresponding to strategy J3. 62

32. Scenario 1. Illustrating Mean Time to Failure in J2 Strategy 67

33. Scenario 2. Illustrating Mean Time to Failure in J2 Strategy 67

34. Scenario 3. Illustrating Mean Time to Failure in J2 Strategy 68

35. Scenario 4. Illustrating Mean Time to Failure in J3 Strategy 68

36. Scenario 5. Illustrating Mean Time to Failure in J3 Strategy 69

37. Scenario 6. Illustrating Mean Time to Failure in J3 Strategy 69

38. Scenario 1. Illustrating Job Completion Time in J1 Strategy 74

39. Scenario 2. Illustrating Job Completion Time in J1 Strategy 75

40. Scenario 3. Illustrating Job Completion Time in J1 Strategy 76

41. Scenario 1. Illustrating Comparison between J1 and J2 Strategy. 77

42. Scenario 2. Illustrating Comparison between J1 and J2 Strategy. 78

43. Scenario 3. Illustrating Comparison between J1 and J2 Strategy. 79

44. Scenario 4. Illustrating Comparison between J1 and J2 Strategy. 80

45. Scenario 5. Illustrating Comparison between J1 and J2 Strategy. 81

46. Scenario 6. Illustrating Comparison between J1 and J2 Strategy. 82

47. Scenario 7. Illustrating Comparison between J1 and J2 Strategy. 83

48. Scenario 8. Illustrating Comparison between J1 and J2 Strategy. 84

49. Scenario 9. Illustrating Comparison between J1 and J2 Strategy. 85

50. Scenario 1. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed 91

51. Scenario 2. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed 91

52. Scenario 3. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed 92

53. Scenario 4. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed 92

54. Scenario 5. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed 93

55. Scenario 6. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed 93

56. Scenario 7. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed 94

57. Scenario 8. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed 94

58. Scenario 9. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed 95

59. Scenario 1. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy. 96

60. Scenario 2. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy. 97

61. Scenario 3. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy. 98

62. Scenario 4. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy. 99

63. Scenario 5. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy. 100

64. Scenario 6. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy. 101

65. Scenario 7. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy. 102

66. Scenario 8. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy. 103

67. Scenario 1. Illustrating the Comparison of Mean Time to Failure in J2 Strategy. 104

68. Scenario 2. Illustrating the Comparison of Mean Time to Failure in J3 Strategy. 105

69. Scenario 1. Illustrating the Probability of Checkpoint Failure. 107

70. Scenario 2. Illustrating the Probability of Checkpoint Failure. 108

71. Scenario 3. Illustrating the Probability of Checkpoint Failure. 109

72. Scenario 1. Illustrating the Number of checkpoints. 110

73. Scenario 2. Illustrating the Number of checkpoints. 111

74. Scenario 3. Illustrating the Number of checkpoints. 112

75. Scenario 4. Illustrating the Number of checkpoints. 113

76. Scenario 5. Illustrating the Number of checkpoints. 114

77. Scenario 6. Illustrating the Number of checkpoints. 115

78. Scenario 7. Illustrating the Number of checkpoints. 116

79. Scenario 8. Illustrating the Number of checkpoints. 117

80. Scenario 9. Illustrating the Number of checkpoints. 118

81. Hadoop Architecture 122

82. Illustrating the migration of VMs in a parking lot. 123

83. Illustrating Virtual Data of Parking lot (Number of vehicles vs. Hours). 124

84. Illustrating the sub-domains D_1 and D_2. 126

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Inspired by the success and promise of conventional cloud computing, Olariu et al. [1–4] have introduced the concept of a *Vehicular Cloud*, (VC, for short), a non-trivial extension of the cloud computing paradigm. Their key insight was that present-day cars are endowed with powerful on-board computers [5,6] and that most of these vehicles spend many hours per day parked in a parking garage, parking lot, or driveway. At the moment, the computing and storage resources of these vehicles are untapped and the opportunity for their use is wasted.

Advances in automotive industry have made the dream of smart cars come true. Ford has announced four million vehicles in the US have its SYNC system (Ford SYNC is a factory-installed, integrated in-vehicle communications and entertainment system). Based on the Ford announcement, in the next three years, nine more million vehicles will have SYNC. Google has tested a self-driving car which is equipped with a central computer, video cameras, radar sensors, GPS, etc. In the Google car which is driver-less, computer takes control of driving. Ford predicts an autonomous-driving mode will be available in its cars by 2017.

Nowadays vehicles are beyond transportation machines. In fact they are the machines with high computing power. People can get different services (navigation, weather, entertainment, etc) while they are in the vehicle as a driver or passenger. In the other hand advances of cloud computing have encouraged companies and users to move their computations, IT services or digital entertainments to the cloud. The main factor of cloud computing is computation power. Vehicles in parking lots could be potential resources for computation power.

It is very likely that, given the right incentives, the owners of vehicles will decide to rent out their on-board capabilities on-demand or on a per-hour or per-day basis, just as owners of large computing or storage facilities find it economically appealing to rent out their excess capacity. For example, we anticipate that in the near future air travelers will park and plug their cars in airport long-term parking lot. In return for free parking and other perks, they will allow their vehicle to participate, during their absence, in the airport datacenter. While these VCs may be used much like traditional clouds, their true novelty and technical challenge lies in the fact that the vehicles come and go, making the assignment of computing resources to jobs challenging.

1.2 OBJECTIVES AND GOALS

This thesis proposes scheduling model and job assignment strategy for vehicular cloud. As we mentioned earlier in Section 1.1, the defining difference between vehicular and conventional clouds lies in the distributed ownership and, consequently, the unpredictable availability of computational resources. As cars enter and leave the parking lot, new computational resources become available while others depart creating a dynamic environment where the task of efficiently assigning jobs to cars becomes very challenging.

We considered two different environment in terms of resource availability: deterministic and stochastic. In deterministic environment, arrival and departure of cars are known. This scenario happens in places like universities and companies where employees schedule is known. Therefore residency time of their cars are deterministic variables.

1.3 CONTRIBUTIONS

We present a scheduling model based on linear programming. We define our model by using linear constraints. The input for this model is jobs with known processing time. We use migration to handle interruption that may be caused by car departure. The objective functions of this model are resource utilization and number of migration. This model minimizes the number of migration and maximizes the resource utilization. Results of simulation show that optimal and near optimal results are fount by this model.

Another contribution of this thesis is a fault-tolerant job assignment strategy, based on redundancy, that mitigates the effect of volatility of resource availability in vehicular clouds. We offer a theoretical analysis of the expected job completion time in the case where cars do not leave during a checkpoint operation and also in the case where cars may leave while checkpointing is in progress, leading to system failure. A comprehensive set of simulations have shown that our theoretical predictions are accurate. We designed our model for different levels of redundancy to explore the trade-offs based on job completion time and mean time to failure.

We propose different job assignment strategies that we call J1, J2 and J3. In J1 strategy we assign a job to one car at a time randomly. Every T time period checkpoint will be created to save the current state of computation. If the car assigned to the job departs, a new car will be recruited and the job will be restarted. The computation will continue from the last saved checkpoint. In this model central server is needed to save the checkpoint. Also optimal checkpointing period should be calculated by using differentiation. We present theoretical prediction of job completion time for J1 strategy. Simulation results confirm our analytical results.

In J2 strategy, each job is assigned to two random cars at a time. Job will be executed on both cars simultaneously. If one of the cars leave, the remaining car undergoes a checkpoint period during which its state of the computation is saved. Once the checkpoint is finished, a new car will be recruited. Job will be restarted on both job using last checkpoint and execution continues. This process continues until the job is completed. In this model there is no need for central server because the checkpoint will be saved on the car. This avoids the problems that may happen because communication between clients and server during bandwidth shortage. In our model we consider two different cases. In first case we assume that car do not leave during checkpointing. This case happens when checkpoint duration is short with respect to car residency time. The second case is when car may leave during checkpointing which leads to failure. We offer theoretical prediction for job completion time for both cases. We also offer theoretical predictions for mean time to failure. Mean time to failure is defined as the time that it takes that failure occurs in system. Simulation results confirm the accuracy of theoretical predictions. We model J2 strategy as Markov process with $N+1$ states. The first state is state zero where no checkpoint has been taken. The second state is 1 where first checkpoint has been successful. Process is in state i, if the all i checkpoint have been successful. In this Markov process transition from state i to $i+1$ is successful if the checkpoint number $i+1$ is completed. If the checkpoint is not successful the process returns to state zero which means failure happened in system.

J3 strategy is similar to J2 strategy. In this strategy each job is assigned to three cars at a time. We model this strategy as a Markov process. We present theoretical prediction of job completion time and mean time to failure for this model. Simulation results confirm the accuracy of analytical results. In J3 strategy mean time to failure is larger that J2 strategy but J2 strategy outperforms J3 strategy in terms of job completion time. Also J2 strategy outperforms J3 strategy in terms of resource utilization, because number of resource assigned to a job is smaller than J3. We present comparison results of J2 and J3 strategies for job completion time.

Number of checkpoints and probability of checkpoint failure are important factors to determine system reliability. By analysing the probability of checkpoint failure, system administrator can determine if the system is reliable for executing jobs. Also by comparing expected number of checkpoints with actual number of checkpoints that has been taken, system administrator can detect if there has been any malicious behaviour. We provide simulation and analytical results for expected number of checkpoints and probability of checkpoint failure.

Our contributions have been published or will appear in the following conferences.

- **Puya Ghazizadeh**, Ravi Mukkamala and Samy El-Tawab: "Scheduling in Vehicular Cloud Using Mixed Integer Linear Programming", The First International Workshop on Mobile Sensing, Computing and Communication (MSCC), Philadelphia, ACM MobiHoc 2014, August 11 - 14, 2014.

- **Puya Ghazizadeh**, Ravi Mukkamala and Stephan Olariu: "Reasoning about Mean Time to Failure in Vehicular Clouds", MILCOM 2014, Military Communications Conference, Baltimore, Maryland, October 6-8, 2014 (Under Review).

- **Puya Ghazizadeh**, Ravi Mukkamala and Stephan Olariu: "Towards Fault-Tolerant Job Assignment in Vehicular Cloud Computing", IEEE MASS 2014, 11th IEEE International Conference on Mobile Ad hoc and Sensor Systems, Philadelphia, Pennsylvania, October 28-30, 2014 (Under Review).

- **Puya Ghazizadeh**, Stephan Olariu and Ravi Mukkamala: "Towards Dynamic Job Assignment in Vehicular Cloud Computing", Thirteenth Annual Graduate Research Symposium (GRS), College of William & Mary, Williamsburg, USA, March 21 and 22, 2014.

- **Puya Ghazizadeh**, Ravi Mukkamala and Stephan Olariu: "Redundancy in Vehicular Cloud Computing", Fourth Annual Graduate Research Achievement Day (GRAD Day) Old Dominion University, Norfolk, Virginia, USA, Thursday, March 27, 2014.

1.4 ROADMAP

The remainder of this dissertation is organized as follows. Chapter 2 provides the necessary background on cloud computing, vehicular network, volunteer computing and vehicular cloud. Chapter 3 presents scheduling model for vehicular cloud established in parking lot based on mixed integer linear programming. This model uses migration in order to prevent interruptions that may be caused by random departure of vehicles. In Chapter 4 our main contribution is a fault tolerant job assignment strategy, based on redundancy, that mitigates the effect of resource volatility of resource availability in vehicular clouds. We offer a theoretical analysis of the expected job completion time in the case where cars do not leave during a checkpoint operation and also in the case where cars may leave while checkpointing is in progress, leading to system failure. Chapter 5 presents two fault-tolerant job assignment strategies, based on redundancy. We offer a theoretical analysis of the mean time to failure of these strategies. In Chapter 6, we present analytical and simulation model for job completion time in stochastic environment considering one car at a time as a computation node. In Chapter 7 we discuss the trade-offs for proposed models based on completion time and mean time to failure. Finally, in Chapter 8, we put the work in perspective, offer concluding remarks and highlight directions for future research.

CHAPTER 2

RELATED WORK

Vehicular Cloud Computing involves different research areas from cloud computing to Intelligent Transportation Systems (ITS). In this chapter we present the main research areas that are most related to vehicular cloud. The following research areas are directly related or have the similar challenges and problems as Vehicular Computing: Cloud Computing, Vehicular Networks, Volunteer Computing (Metacomputing), Connected Cars.

In this chapter we will identify the similarities of these research areas and their relation with VC.

2.1 CLOUD COMPUTING

The idea of cloud computing originated from different types of technologies like grid computing, distributed systems, cluster computing and utility computing. Cloud computing provides IT services to users by third party over the Internet. In [7], National Institute of Standards and Technology (NIST) defines Cloud Computing as "a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". As [8] and [9] have mentioned, the concept of cloud computing is initiated from the recognition that renting infrastructure and software would be more beneficial rather than investing in them. The idea of cloud computing is supported by low-cost high speed Internet, virtualization and advances in parallel and distributed computing and distributed databases. One of the advantages of cloud computing is offering scalable computing power and IT services. Cloud computing concept has been accepted by many companies in US and all over the world. Most of these companies have under-utilized infrastructure and cloud computing will improves their resource utilization.

With cloud computing, developers that have new ideas do not need to have large capital to spend on hardware and infrastructure. They can simply implement their ideas on cloud infrastructures. Three aspects are novel in cloud computing in terms of hardware point of view:

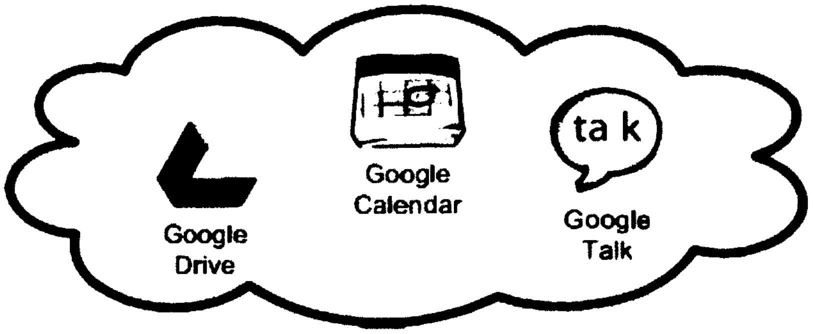


Fig. 1: Google Apps as Software as a Service

1. It gives the user the ability of having unlimited computing power on demand, therefor there is no need for user to plan far ahead for resource provisioning.
2. It eliminates up-front economic commitment by cloud users, therefore it allows them to increase hardware based on the demand.
3. It gives the users the option to pay for computing power on a short-term basis as needed (e.g., processors by the hour and storage by the day) and release them as needed, thereby rewarding conservation by releasing resources (e.g. machines and storage) when they are no longer useful.

Services provided by cloud computing are classified into three categories: Software as a service (SaaS), Platform as a service (PaaS) and Infrastructure as a service (IaaS) [8–13]. Figure 2 shows these services.

SaaS: SaaS is a model that provides applications to end users over the Internet or local networks (e.g. Google Apps: Google Calendar, Google Drive, Google Talk, etc). In SaaS cloud provider is responsible for software licensing which means user only pays for the delivered service rather than software license. Following advantages make SaaS attractive for users: 1- There is no need for user to update the software. 2- All the users will use the same version of software which creates compatibility. 3- User can have access to software from everywhere [14]. Figure 1 illustrates an example of SaaS.

PaaS: In platform as a service, computing platform is provided to users to develop software (e.g. Google's App Engine, which provides Python runtime environment and

APIs for applications to interact with Google's cloud runtime environment). Developers build their software in the distributed environment offered by PaaS. This environment will enable developers to test the software for different scenarios.

IaaS: IaaS is a service that provides fundamental resources like storage, processor and network to users. In this model, service provider is the owner of the infrastructure and is in charge of maintenance (e.g. Amazon Elastic Compute Cloud (EC2)). In this type of service, user controls the hardware which means that user is responsible for scaling and failover processes, because scalability and failover processes are application dependent [15].

2.1.1 CLOUD COMPUTING ADVANTAGES

Cloud computing is growing and companies are moving their computation services to cloud in order to benefit from the services that can be provided by cloud providers. The following are some of the advantages of cloud computing [15]:

1. Unlimited computing power based on the user's need. Cloud computing providers are able to provide high computation powers to users. Virtualization is one of the main reasons in utilizing the servers which eventually increases the performance of servers. Virutalization is simulation techniques to provide virtual resources. We explain virtualization in more details in Section 2.1.5.

2. No need for up-front investment. Users use the cloud computing infrastructures without investing on them.

3. Pay as you go: users only pay for the service that they receive. The other cost like maintenance is not users' concern any more.

4. Scalability: users can scale up or scale down the computation based on demand (e.g. web server for a special event like Olympic games or conferences which the demand is high at a specific period of time).

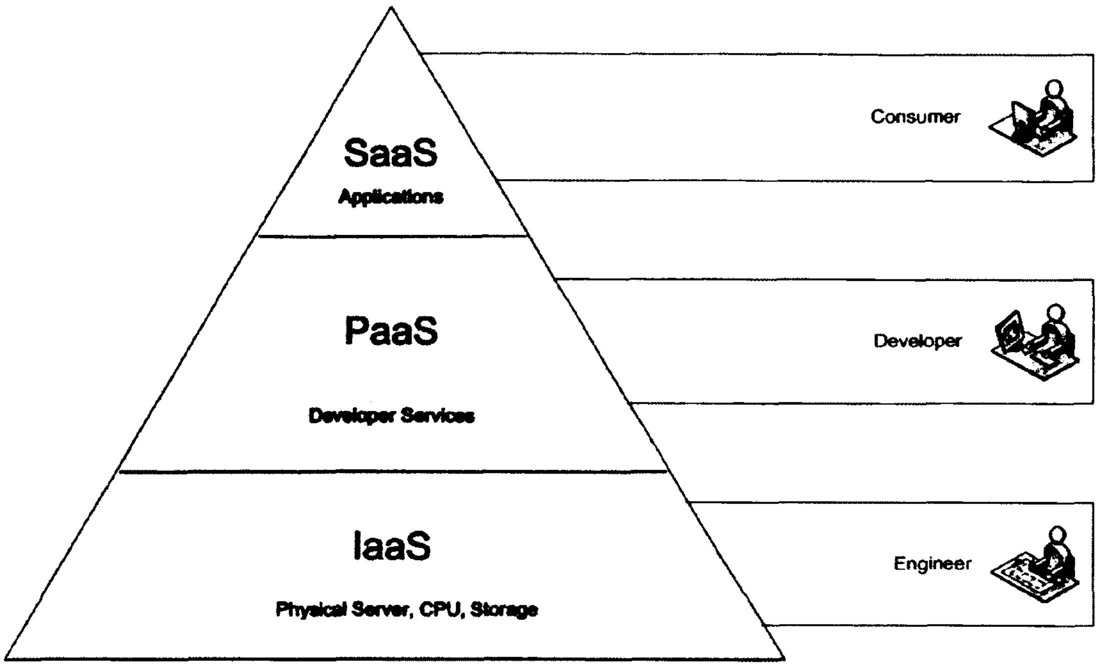


Fig. 2: Cloud Services

2.1.2 APPLICATIONS DESIGNED FOR CLOUD COMPUTING

In order to design an application for cloud, following features should be considered: [16]

1. Each component of application should be scalable. In that case user can scale up or scale down the application based on demand.
2. In order to handle failure in distributed system, application components should be loosely coupled. By having loosely coupled components, failure of one component will not affect the other components. Therefore availability of system will increase.
3. Parallelization is the key point of increasing performance in distributed system. Parallelization will distribute the task over the distributed computation nodes which improves the performance.
4. There should be a recovery mechanism for failures (e.g., checkpoint)

2.1.3 CLOUD TYPES

Based on the ownership and accessibility clouds are classified into three categories: public cloud, private cloud and hybrid cloud.

Public Cloud In public cloud, infrastructure is owned by cloud provider. In this type of cloud, service is available to public through Internet. Service availability in public cloud is high but security level in compared to private cloud is low. User does not have control on public cloud, therefore service flexibility has lower level compared to private cloud.

Private Cloud In private cloud, infrastructure is owned by user. This type of cloud usually is used for application development. User has a high level of control in private cloud (e.g. controlling bandwidth). Also private cloud has high level of security and service flexibility. The other application for private cloud is financial services. High level of security is essential feature for financial services which can be provided by private cloud.

Hybrid Cloud Hybrid cloud is a combination of public cloud and private cloud and they can be coupled to each other. In IT services at a specific period of time or peak period, need of extra capacity may emerge. Since in hybrid cloud, public cloud is coupled with private cloud, as a result, workload can be moved from private cloud to public cloud and vice versa. Therefore in case of failure in one of the clouds, service still can be available to users though the other one. Figure 3 shows hybrid cloud which includes both public and private cloud.

2.1.4 CLOUD ARCHITECTURE

Cloud architecture has features that can handle the following problems that are common in typical distributed computation models [16]:

- Providing all the computation resources that an application needs.
- Providing on-demand resources on time.
- Providing recovery mechanism for failure in distributed computation environment.
- Scaling the computation resources based on the load.

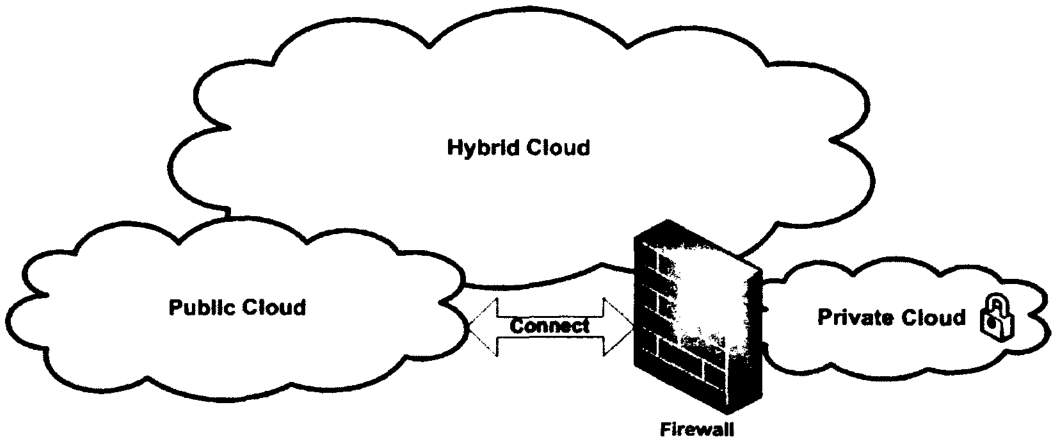


Fig. 3: Types of Cloud

- Releasing the assigned computation resource from terminated task and reassign it to new task.

The main components of cloud computing that are common in other computation models are: database, storage, computation nodes and controller. Computation nodes are independent components therefore failure of one node will not affect the other nodes. Since components are independent, database is used to record the status of each component. By executing query against database, we can obtain the state of system any time. Storage is used for saving input data, fetching data for computation and saving outputs. Storage in clouds will save several copies of data to protect them against damages. Controllers are used to control each component and manage the communication between components. Controllers also manage the communication of client and server applications with data storage. As we mentioned in cloud computing, components are independent. In cases that components should be coupled, they should be defined loosely. In other words, they should be loosely coupled. In order to define loosely coupled components, a queue system will be used that behaves like buffer. The output of one component that is an input for the other component will be stored in a queue before getting to the other component. In that case if the first component fails the second component can continue to executing the queue until the first component recovers. Figure 4 is an example for queue service.

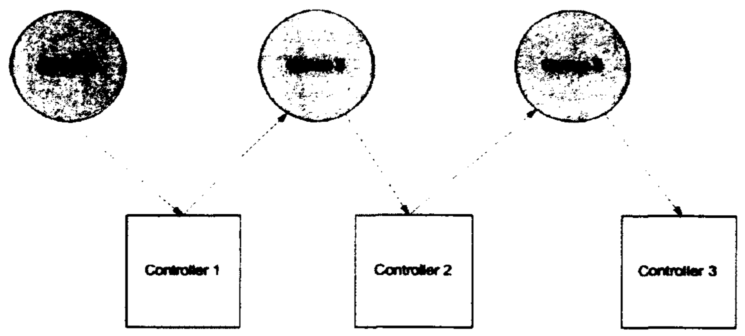


Fig. 4: Queue Service

2.1.5 VIRTUALIZATION

Virtualization, in computing, is simulating techniques that provide a layer between hardware and operating system. This layer can provide a virtual resource (e.g., CPU, storage, network, etc). In other words virtualization provides a logical view of resources rather than physical view. For example server virtualization is the ability of executing multiple applications or operating systems on one physical server. Applications or OSs that are running on one physical server should be isolated in order to be running securely. Virtualization has been started from mid 20th century. The main idea of virtualization is to utilize the resources. Virtual machine is the main factor of cloud computing. Cloud computing has been developed based on VM technology. Some of the virtualization advantages are:

- VM takes away hardware dependency
- Create a pool of resources on hardware
- Virtual machines are set of files that move between physical servers
- VMs can be isolated. Failure of one of them can not affect the other one
- Provides high availability (you can run it everywhere e.g. USB)
- Virtualization is a good solution for sharing resources
- Virtualization utilizes servers

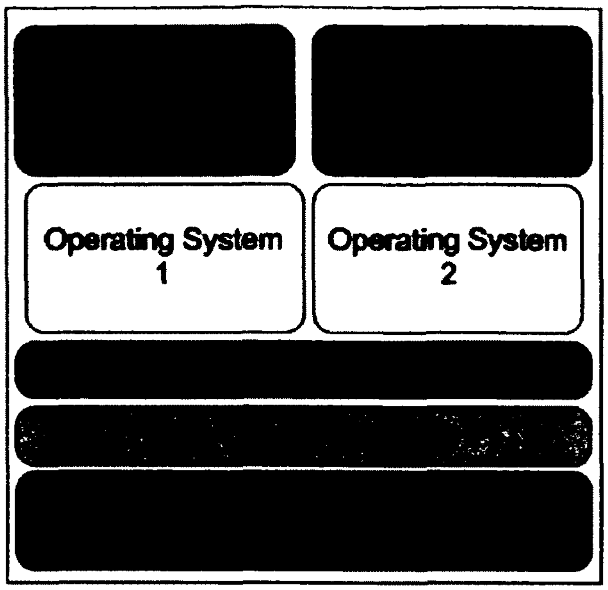


Fig. 5: Hosted Virtualization.

Virtualization Types

Virtualization is classified into three classes: 1-Application Virtualization, 2-Hosted Virtualization, 3-Hypervisor Virtualization [17].

Hosted Virtualization is virtualization on top of the operating system (hosting OS is called domain0). This type of virtualization is for users to have different separate OS for experimenting different software. In other words this type of virtualization is for personal experiment rather then cloud computing. Examples for Hosted Virtualization are VMware Workstation, VMware Fusion and Microsoft VirtualServer 2005. Figure 5 shows Hosted Virtualization.

Hypervisor Virtualization is virtualization on top of hardware. In this type of virtualization hypervisor software will be installed on hardware instead of domain0. This type of virtualization is more secure because there is no security concerns related to hosting OS, domain0. Hypervisor will directly access hardware which increases security and resource management ability. This type of virtualization is used in cloud computing infrastructures. Here are the examples for Hypervisor Virtualization software: VMware ESX Server (vSphere) and Citrix Xen Server. Figure 6 shows different layers in Hypervisor Virtualization.

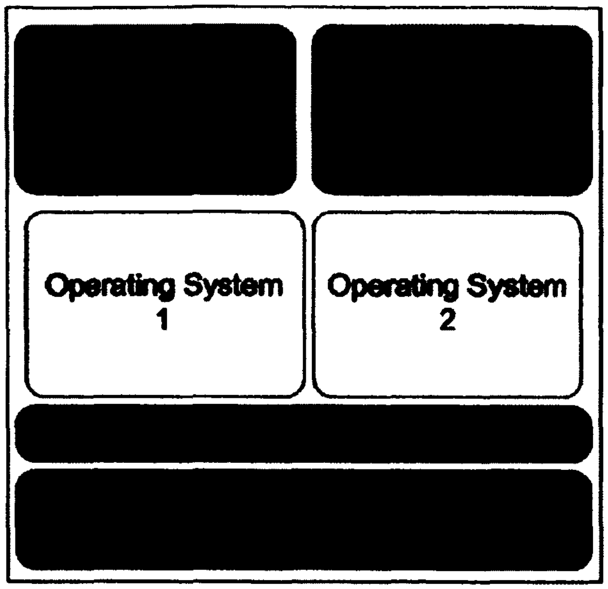


Fig. 6: Hypervisor Virtualization.

Application Virtualization is virtualization for application layer. Application will be installed in sandbox which is testing environment for untested software from unverified party. In this model application does not have direct access to underlying OS. This type of virtualization is for running conflict applications that can not run in the same time (e.g., different version of same software). Examples for this type of virtualization are VMware ThinApp, Microsoft Application Virtualization. Figure 7 shows the order of layers in Application Virtualization.

VM Migration

IT administrators use virtual machine migration in data centers and clusters to manage load balancing, handle failure and lower maintenance cost [18]. Virtualization is a solution for migrating computation between different physical servers because data and different status of computation will be saved as a file. There are two types of virtual machine migration: Live Migration and Cold Migration.

Live Migration: In live migration, applications can be transferred to different physical servers while running. Since during the migration process the application is not suspended, performance of the system does not decrease. This type of migration is resource intensive. Page status copying is the main component of live migration.

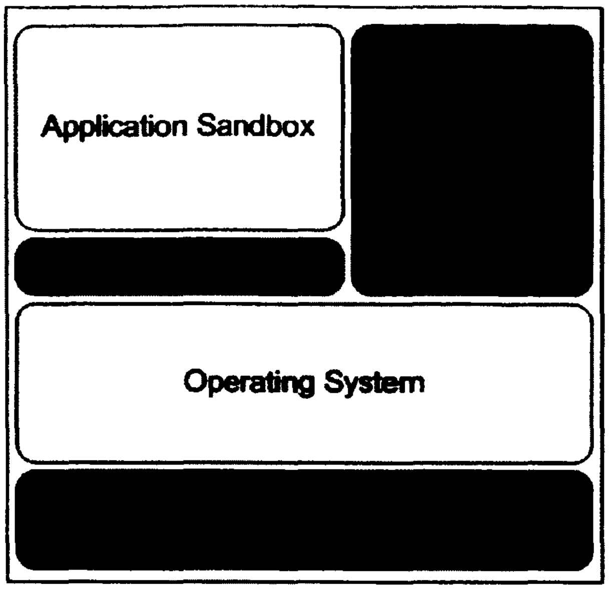


Fig. 7: Application Virtualization

Cold Migration In cold migration, the virtual machine will be turned off which means the operating system and application are suspended before transferring to physical machines. In cold migration, because of the interruption that caused by stopping the application, performance of the system may decrease.

VM Cloning

Cloning is creating a new VM from existing VM with same configuration. New created VM can resume the same process from the previous VM [19]. Cloning instead of installing operating system, applications and configuring, makes copies of pre-configured VMs in order to save time. Cloning is used in the environment that groups of users use same operating system and software.

Migration Time

In this section we estimate a range for virtual machine migration time. This estimation does not affect our approaches that we present later in Sections 3, 4, 5 and 6. The goal of this part is to provide more realistic migration time that can be used in different scenarios that we present in next chapters.

Nelson et al. [20] experience different measurements and show the migration time and

Operating System	Migration Time
Windows	6-8 Seconds
Linux	6-8 Seconds

TABLE 1: Different Types of Operating System.

Application	Migration Time
iometer	6-8 Seconds
memtest86	16 Seconds
dbhammer	16 Seconds

TABLE 2: Different Types of Application.

application downtime for different workloads. In their experiments migration is between two different physical machines for applications on Intel x86-based operating system, including Microsoft Windows and Linux where guest operating system runs over virtual machine.

In their approach virtual machine makes migration possible by capturing the state of hardware and software within the virtual machine. The largest part of state in virtual machine that needs to be migrated is physical memory.

Based on [20], virtual machine migration includes following stages:

1. Start the migration by choosing the virtual machine to migrate and choosing its destination.

2. While VM is running, pre-copy the VM memory state to the destination.

3. Sending the non-memory state and quiescing the VM.

4. Moving the VM control to destination and continue the running at destination.

5. Transferring any remaining memory state and eliminate the dependency on the origin machine.

Nelson et al. [20] experience different scenarios with following software and application:

- idle: An idle Windows 2000 Server.
- kernel-compile: Linux kernel compilation in RedHat 7.2.
- iometer: Iometer [21] running on Windows 2000 Server. Iometer was configured to run disk I/O with 3 worker threads each performing I/O to a 500MB file with up to five outstanding writes.
- memtest86: Memtest86 [22], which continuously reads and writes memory, running test #1 in a loop
- dbhammer: Database Hammer [23], a client/server database load generator running on Windows 2000 Server. The server was migrated while the client was running on a third physical machine.

They emphasize that end-to-end migration time depends on the size of VM's memory. In their measurement for VM's memory with size 512 MB, end-to-end time of migrations are idle: 6-8 seconds, kernel-compile: 6-8 seconds, iometer: 6-8, memtest86: 16 seconds, dbhammer: 16 seconds. Tables 1 and 2 show the list of Operating System and application used in their experiment.

If we consider a normal laptop as on-board computation device in vehicle the VM's memory size can vary up to 10 GB easily. Since VM's memory has a strong effect in migration time, we multiply the previous scenario by $(10 * \frac{1024}{512} = 20)$ to estimate the migration time. Considering running same application as mentioned above the migration time can be estimated as eight minutes for running either memtest86 or dbhammer. Nowadays most of the processors have more than one core (e.g, Intel core i3, i5 and i7 that are typical processors in laptop) which means they can process multiple CPU instructions at the same time. Therefore multiple applications can be processed in the same time. In case of considering execution of both memtest86 and dbhammer the migration time can be considered as fifteen minutes. In a strict scenario we can add the time of setting up the arrived vehicle in parking lot, plugging to Ethernet cord and power cord, and consider the migration time as twenty minutes.

2.2 VEHICULAR NETWORK

Vehicular Ad Hoc Networks (VANET) is a wireless ad hoc network of mobile vehicles. In this network communication can be between vehicle and vehicle (V2V) or vehicle and roadside infrastructure (V2I).

Federal Communication Commission (FCC) allocated a spectrum frequency for communication between vehicles to vehicles and vehicles to roadside infrastructures fifteen years ago. The FCC then allocated 75MHz of spectrum in the 5.850 to 5.925 GHz band for the exclusive use of Dedicated Short Range Communications (DSRC) Service in 2003 [24–27]. The approximate range of the communication is 1000 meters.

The original goal of VANET was to provide drivers with notification of real-time traffic conditions and prevent car accidents.

Based on US Department of Transportation (US-DOT) and National Highway Traffic Safety Administration (NHTSA) statistics, every year traffic jam in highways cost the nation over $70 billion in lost worker productivity and over 8.5 billion gallons of fuel wasted, not to mention high levels of carbon emissions [28–33].

Based on US-DOT highway incidents cause over half of all congestion events [34]. Further, the NHTSA indicates that one of the leading causes of traffic accidents is congested roads. These problems have motivated scientists to use recent technologies and extend the idea of Mobile Ad-hoc Networks (MANET) to roadway and street communications. The new type of networks, referred to as Vehicular Ad-hoc Networks (VANET) that gives drivers advance notification. The original motivation for the interest in VANET was to notify drivers of road condition, congestion, hazardous driving conditions and other similar traffic-related concerns [27,35]. Therefore VANET emphasizes on applications for traffic status reports, collision avoidance, emergency alerts, cooperative driving, and other similar concerns [27, 35]. Many researchers and scientist in VANET community predict that advances in VANET or other developing vehicle-based computing and communications technology, have an enormous impact on society [26, 36–38]. Because of this predicted social impact, many car manufacturers, government agencies and standardization institutes all over the world have created national and international associations dedicated exclusively to VANET. Some of these associations are Hondas Advanced Safety Vehicle Program, Networks-on-Wheels, the Car-2-Car Communication Consortium, the Vehicle Safety Communications Consortium [26,38]. Rapid converge of ITS and VANET that has happened in the past few years is heading to the development of Intelligent Vehicular Networks (IVN). It is expected that IVN changes the way we drive by providing a safe, secure and well-conditioned constant computing environment that will be spread on streets and highways.

2.3 CONNECTED VEHICLES

Now that houses and businesses are connected to the Internet, vehicles are the last major market for connectivity.

Connected vehicles is a concept that has been presented by U.S. Department of Transportation. Connected vehicles is defined as Vehicles that interact with each other (V2V), the roadside infrastructure (V2I), and beyond (V2X) via wireless communications.

The idea of connected vehicles is to solve real-word problems [39]. Following are the issues that have been addressed by connected vehicles.

Safety Application: using Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) Communications for safety purpose.

Collecting and Managing Real-time Data: In this method the gathered traffic, transit and freight information will be analyzed. By analyzing the information, traffic will be managed in most efficient, safest and most environmentally friendly possible way.

Dynamic Mobility Applications: This research investigates the different technologies to improve the transportation. In other words finding fastest and most environmentally friendly trip by using different type of transportation (car, bus, truck, train, etc.) is the topic for this research. This research helps companies to use cross-modal travel system in managing their transportation needs.

Road Weather Management: This research will use weather information in helping travelers and transportation system. This information is vehicle-based which means it has been gathered by monitoring current weather condition via vehicles on roads.

Applications for the Environment Real-Time Information Synthesis (AERIS): This research helps transportation managers to consider environmental impact by using data from tailpipe emissions with combination of other environmental data.

Most of the car manufacturers have embedded a system in their selected models that provide different applications. Every year new applications are introduced by car companies that presents new services. Figures 8, 9 are embedded devices that Ford and GM use to contribute to the idea of connected vehicle.

Connected vehicles provide many advantages for vehicle owners and takes driving experience to the next level. In the same time businesses, and government agencies that own highways can manage the traffic more efficiently.

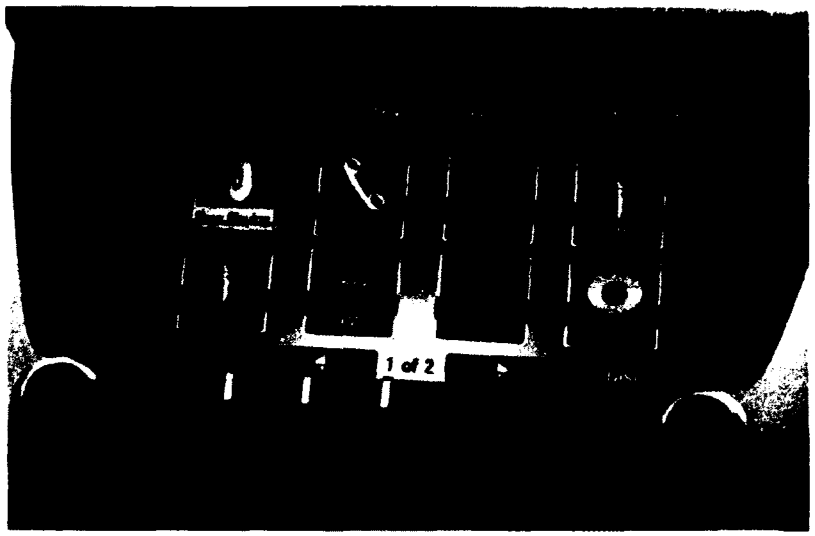


Fig. 8: GM MyLink Telematics System: Photo Credit (http://gm-volt.com/)

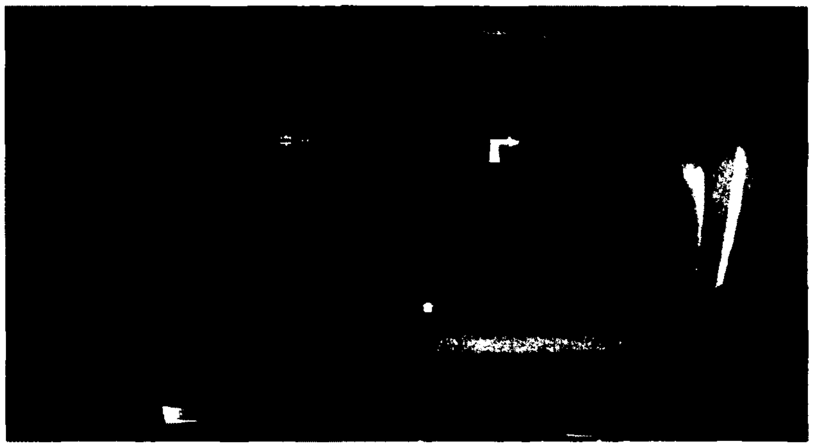


Fig. 9: Microsoft-based Ford Sync: Photo Credit (http://www.ford.com/)

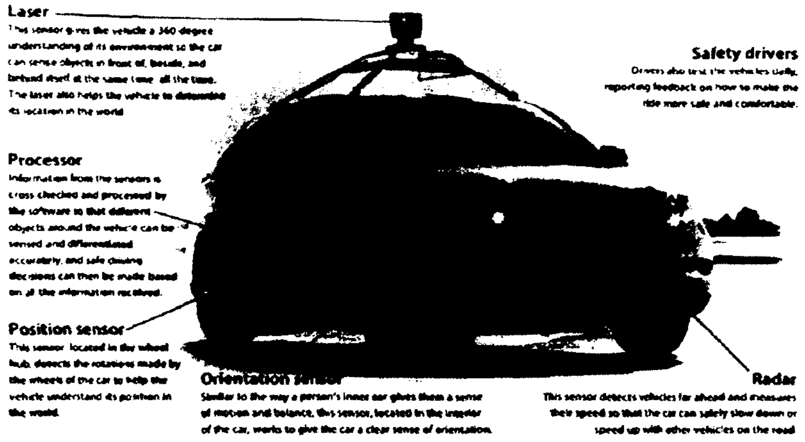

Laser
This sensor gives the vehicle a 360 degree understanding of its environment so the car can sense objects in front of, beside, and behind itself at the same time all the time. The laser also helps the vehicle to determine its location in the world
Safety drivers
Drivers also test the vehicles daily, reporting feedback on how to make the ride more safe and comfortable.
Processor
Information from the sensors is cross checked and processed by the software so that different objects around the vehicle can be sensed and differentiated accurately, and safe driving decisions can then be made based on all the information received.
Position sensor
This sensor located in the wheel hub, detects the rotations made by the wheels of the car to help the vehicle understand its position in the world.
Orientation sensor
Similar to the way a person's inner ear gives them a sense of motion and balance, this sensor, located in the interior of the car, works to give the car a clear sense of orientation.
Radar
This sensor detects vehicles far ahead and measures their speed so that the car can safely slow down or speed up with other vehicles on the road

Fig. 10: Google Car: Photo Credit (http://techcrunch.com/)

The idea of connected vehicle is supported by different companies like Audi, BMW, Ford, GM, Mercedes-Benz, Toyota, Volkswagen, Hyundai, Nissan.

2.4 SMART CARS

Along with connected vehicle concept, there are other concepts that have been introduced by different companies. Google announced its driver-less car. Google self-driving project was started in 2009 and has been a very successful project. The goal of this project is to avoid accidents caused by human error. Based on the statistics 93 percent of incidents are caused by human error. In the past five years that Google car has been on trial, no incident has happened. This means that Google car accident rate is zero. Google car has lasers, positional and orientation sensors, and radar. By using these devices, Google car is able to see 360 degrees.

Ford Evos concept car is a new model that has been introduced to public at 2011 Frankfurt Motor Show. Evos is a cloud connected car. User information and personal data stored in cloud are accessible by Evos. Therefore Evos knows the person's preferences, needs, health and work schedule, etc. Evos is able to fetch and process data from cloud to determine user's preferences based on traffic, weather and work schedule.

2.5 VOLUNTEER COMPUTING

Volunteer computing is a model of computing based on shared resources. In this model users all over the world share their computers as a source of computing power and storage through internet. Volunteer computing emerged in 1990 and created large projects by recruiting hundred volunteer machines.

Vehicular cloud and volunteer computing are similar in different aspects and shares some challenges like ownership, privacy and security. In both of these models service provider is not the owner of the computation resource. Therefore the hosts are not trusted. In the same time host is concerned about its own data privacy and security.

In volunteer computing in order to protect the host privacy, computation will be inside a sandbox that prevents accessing host's data. This protects hosts from viruses and Trojan horses that might try to steal or destroy their data. Thus, potential hosts can feel safe about privacy and security of their data during the computing time.

2.6 VEHICULAR CLOUDS

As mentioned in Section 2.1, cloud computing concept has been very successful in providing and utilizing computing resources.

We believe in a near future vehicles on the roads, streets and parking lots will be considered for potential computing resource that can be organized and utilized by third party to provide computation resources. These vehicles everyday spend hours in parking lot or driveway and during these hours, their computational and storage capabilities are wasted. These features make vehicles interesting resource to be considered as computation node. We envision that by giving right incentives, the owner of cars are willing to rent out their on-board computation device. For example traveler can park and plug their car in airport parking lot when they are on travel. Vehicles can use the airport facility as a resource of electricity power to run the computation. Also vehicles can have access to parking garage data-center. In the similar scenario vehicles that are stuck in traffic jam can use their computation power to manage the traffic through collaborating in running complex simulation designed for traffic control. For example this storage can be rented by data-center manager. Another example is to use the storage in content delivery and p2p applications where a file is decomposed into several blocks distributed across nodes of the network and interested users need to collect different blocks to reconstruct the file.

2.6.1 SERVICE TYPES

Vehicular cloud computing can provide at least two types of services: Storage and Computation service.

Storage as a Service

Vehicles with additional storage capacity can participate in storage service. Because of the low cost of storage, it is predictable that computers on cars will have multiple Terabytes of storage attached to them. Because of the available space in cars, accommodating a hard drive is possible easily. The available storage can be used by many application in the cloud.

Computation as a Service

Vehicle that are computation nodes can be used to execute distributed scientific computing. These distributed resources can be used to run large-scale tasks. This task can range from community-based initiatives, such as Wikipedia, Delicious, Slashdot to open source software development project or it can be used to provide on-demand computation to individual like Amazon EC2.

2.7 SECURITY AND PRIVACY

One of the main concerns in shared resources is privacy and security which is the case in vehicular cloud. In vehicular cloud, security and privacy of both vehicle's owner and user who uses the rented vehicle must be protected.

Virtualization technique is able to address these privacy concerns [40–42]. In this regard, virtualization refers to a concept in which access to a single underlying hardware is controlled so multiple guest operating system can share that single piece of hardware, with no guest operating system being aware that it is actually sharing anything at all. Therefore by using virtualization many customers at same time can rent on-board computation power in a vehicle and none of them can violate the privacy and security of each other or vehicle owner. This will protect both privacy and security of all the users on the system. A number of successful virtualization vendors exist today including, among others, Citrix [43] and VMWare [44].

2.7.1 ATTACKS IN VEHICULAR CLOUD

Vehicular clouds are more subject of failure because of their nature. Failure are usually caused by crashing node or leaving cars. Beside the unintentional faults that can be caused by data loss or corruption due to faulty network links or faulty processors, there could be intentional faults. This type of fault can be caused by malicious nodes submitting invalid data. This type of attack where nodes submit invalid results intentionally is called sabotage.

Dishonest user may try to get paid just by returning random numbers instead of doing real work. In this case dishonest user will create error in the system not only by submitting incorrect results also by propagating error that will affect other computations.

Another type of attack is when users steal information from data that are given. This type of attack is called espionage. In this case one company may get access to its competitor's data by spying as a computation node.

Protecting against malicious users is very difficult. Possible ways of addressing this problem includes using cryptographic techniques such as digital signatures and checksums to protect against volunteers sending back random results, encrypted computation to protect against espionage and attempts to forge digital signatures and checksums, and periodic obfuscation (i.e., instruction scrambling) techniques to simulate encrypted computation when it is not possible.

Cryptography is a solution that has been proposed protects computation against malicious users. Cryptographic techniques include digital signatures and checksums. Other data integrity evaluation techniques also have been presented by researchers [45]. In cases that these methods are not applicable, redundant computation can be used. In redundant computing we give the same task to more than one computation node, and have them to vote on correct answer. Researchers have shown that redundancy reduces error rates exponentially [46]. In traditional fault-tolerance technique, malicious users may be able to crack the code and create valid checksums for invalid results. But redundant computation can be used even in the presence of malicious users.

2.8 EXAMPLES OF VEHICULAR CLOUDS

In this section we demonstrate the power of Vehicular Cloud concept. We discuss several possible instances of Vehicular Clouds that use the computing power of vehicles in parking lots.

2.8.1 A DATACENTER AT THE AIRPORT

Long-term parking lot in airport where cars are parked for days while their owners are on travel is a good potential for vehicular cloud. The airport can provide Ethernet connection (e.g., 100 Megabit Ethernet, 1G, or even 10G Ethernet connections) and power source (e.g., regular power outlet) to parked vehicles. Travelers that allow their vehicle to participate in airport vehicular cloud will share their travel plans (e.g., their departing and arriving flights), making it easier for the airport to schedule the existing resources.

By having the schedule of travelers, the long term parking lot at the airport offers a relatively stable, long-term availability of resources. Finding parking spot in airport long term parking lot has always been a challenge because most of the time they are full. Having a parking lot full of cars indicates that the amount of resources is likely to be plentiful. We expect that in near future the computation resources in different airport parking lots can be joined together to make more stable environment in term of computation resources. In this case vehicular cloud will have the excess capability of any cloud. In return owner of vehicles who participate in computation as a computation node will get free parking spot with other pecuniary advantages.

2.8.2 DATA CLOUD IN PARKING LOT

The next example of vehicular cloud is the following: consider a company or university with 500 employees. Since 90% of Americans drive to work [47] we expect that at least 350 vehicles be in the parking lot. Company or university by providing appropriate incentives can rent employees vehicle as computation nodes in daily, weekly or monthly bases. In this case company or university instead of renting computation services from outside, it could use the under utilized resources of employee in its parking lot with great price. In this scenario the architecture of the vehicular cloud will be almost identical to the architecture of a conventional cloud, with the additional twist of, perhaps, limiting the interaction to weekdays.

2.8.3 DATACENTER AT THE MALL

Every day many people spend many hours in shopping center. Based on US statistics in 2008 that performed on teens, 95% of shoppers spent more than one hour at the mall while 68% of them spent more than two hours [48]. Therefore hundreds of thousands of customers spend many hours in mall and shopping center while their vehicle is parked and its computation is left under-utilized. These under-utilized resource can be used and provide IT services. The mall can manage these resources by giving incentives to car owners.

2.8.4 SPECIAL EVENT MANAGEMENT

Based on the statistics every year in US 24,000 special events are planed. These events include concerts, sport activities and festivals where the number of attendance is greater than 10,000 people. These events cause between 93 and 187 million hours of travel delay and between $1.7 and $3.5 billion in additional travel costs [49,50].

Owner of these cars that attend these events and park in the parking lots nearby will be more than willing to allow their cars to participate in vehicular cloud set up by either the municipality or the event organizers in return for free parking. Those cars can use their computing resources and offer additional computing power that can be used to dispersal schedules from the event, or develop alternative traffic flow control strategies in response to incidents or changes in desired departure times from an event.

2.9 SUMMARY

In this chapter we discussed research areas that are related or similar to vehicular cloud like, cloud computing, vehicular network, volunteer computing. Vehicular cloud and traditional cloud have many common challenges. Therefore most of the solutions that have been proposed for traditional cloud can be used in vehicular cloud. But the main difference between vehicular cloud and traditional cloud is in resource availability. We present fault tolerant strategies in Chapters 4 and 5 to overcome this problem.

In this chapter, we discussed how VANET can be used in creating mobile vehicular cloud. We also mentioned the privacy and security challenges in vehicular cloud that are in common with volunteer computing. Finally we presented several examples of vehicular cloud like data center in airport and mall.

CHAPTER 3

JOB SCHEDULING IN DETERMINISTIC ENVIRONMENT

3.1 INTRODUCTION

Traditional cloud includes large numbers of servers. These servers are established in cloud infrastructure and are available to provide service anytime unless there is a failure. Failure can be from software or hardware. In other words traditional cloud has static environment in term of computation power unless there is a failure. In contrast VC has dynamic computation nodes. Vehicles which are the computation nodes can arrive or depart any period of time which makes VC a dynamic environment in terms of computation power. Random arrival and departure of vehicles may interrupt the jobs that are running. In this chapter we present a model for scheduling in VC that prevents interruption in job execution. This model is based on migration.

The remainder of this chapter is organized as follows. In Section 3.2, we describe migration and different scheduling approaches. In Section 3.3, we present our approach with an example. Section 3.4 gives the details of implementation and experiment result. Conclusions is given in Section 3.5.

3.2 RELATED WORK

3.2.1 SCHEDULING

Scheduling problem has been a topic for computer science operations research from mid-twentieth century. Job scheduling problems are defined as allocating limited resources to jobs in order to optimize the objective function.

In terms of computational complexity, most of the job scheduling problems are classified as NP-hard problems. NP-hard problem can be solved in polynomial time using a non-deterministic Turing machine.

Different solutions have been presented for scheduling: artificial intelligence, priority rules, heuristic algorithms (e.g., GA, PSO, etc) and operational research which includes simplex method, cutting plane, branch and bound [51].

Operational Research

In operational research method, problem is separated from solving algorithm. Problem should be defined based on mathematical model and solving algorithm (e.g., simplex method) is used to solve the problem.

Priority Rules

There are different types of rule-based algorithms like First Come First Service (FCFS), Shortest Processing Time (SPT), Longest Processing Time (LPT). Some of these algorithms have been used in CPU scheduling. Priority rules are efficient in terms of execution complexity but in some of the large and complex environment optimal solution can not be found by using priority rules [52]. Priority rules are extracted based on analyzing different experiments. Therefore efficiency of the rules depends on the experiments. priority rules cannot be adapted to all environments [53].

Heuristic Algorithms

Heuristic algorithms have been developed based on the biology principles [54]. Based on these principle solutions should evolve during each generation. Evolution of the solution leads to optimal or near optimal solution. Genetic algorithm (GA) [55], taboo search algorithm [56] and simulated annealing algorithm [57] are examples of heuristic algorithms.

Artificial Intelligence

Neural Network and constraint programming are examples of artificial intelligence scheduling method [58]. One of the advantages of artificial intelligence method is considering the changes in environment and responding to these changes.

There are combined methods which apply two different scheduling algorithms like priority rules and heuristic algorithms in order to create an efficient scheduling algorithm.

In our approach we use linear programming model that belongs to operational research class.

3.2.2 LINEAR PROGRAMMING

Linear programming has been introduced in early mid-20th century [59] and is highly structured mathematical model. In this model objective function and constraints are linear.

Each constraint can be a linear equality or inequality. Objective function is a mathematical function of decision variables that measures the performance of the solution. Based on the decision variables, model can be classified into linear programming, integer linear programming or mixed integer linear programming. In linear programming [59] decision variable are non-integers. Integer programming is another version of linear programming that decision variables are integer values. The model that some of the decision variables are integer and some of them are non-integer is called mixed integer programming. Both integer programming and mixed integer programming mathematical models are linear programming model.

In linear programming solving algorithm and problem model are separated form each other. In other words by changing the problem model, there is no need to change the solving algorithm. This feature makes linear programming a flexible approach. Different version of problem can be defined by adding or modifying the equations without changing the solving algorithm.

3.2.3 MIGRATION

In a typical vehicular cloud (e.g. airport parking lot) vehicles arrive and depart in different period of time. As we mentioned vehicles are the computation resources in vehicular cloud. Therefore computation resources are dynamic in terms of availability. In order to handle the interruption which is caused by vehicle's departure, we use migration. In VC migration is a process of transferring a job from one vehicle to another vehicle or server.

We present an approach for job scheduling in dynamic environment in terms of computing resources. Our model is based on mixed integer linear programming. In our approach, we use live migration in order to transfer VMs to different physical machines. In our model each vehicle is physical machine or computation node.

3.3 OUR APPROACH

3.3.1 PROBLEM DEFINITION

Our scheduling problem can be defined as a parallel machine scheduling. In this type of problem, jobs are assigned to several parallel identical machines. This type of scheduling includes combination and permutation. Permutation is assigning jobs to machines and combination is for defining different sequence of jobs on each machine. There are *m* identical parallel machines and n jobs with known processing time.

This work investigates a job scheduling problem involving non-preemptive tasks with known processing time where job migration is allowed. In non-preemptive task, once a task has been given the resource it cannot be taken away from that task. Assigning a job to resources is valid if job has been executed fully and continuously (no interruption). A job can not be executed in parallel. In this work resources are dynamic in terms of availability.

In our approach, the determination of an optimal job schedule can be formulated as maximizing the utilization of VC and minimizing the number of job migrations. Utilization can be calculated as a time period that vehicles have been used as computation resources.

3.3.2 0-1 MODEL DESCRIPTION

In this model we define a decision variable X_{ijk}. i is the index for jobs. j is the index for hour slots. Each parking spot has an ID and k is the index for parking spots in parking lot. X_{ijk} value is 1 if job i has been assigned to a vehicle in parking spot k at time slot j. Otherwise value for X_{ijk} is 0. In this model parking lot has constant number of parking spots. The first part of the objective function for this model is the following:

$$\sum_{ijk} X_{ijk} * (PMAX)_{jk} \tag{1}$$

PMAX is a matrix that shows the parking occupancy. Each element of this matrix can have value either 1 or -1. Value 1 means that a vehicle will be available as a computation node in parking spot j and hour slot k. Value -1 means in parking spot k and hour slot j vehicle is not available. Constraint 2 is for defining that each parking spot in a specific hour slot can not have more than one job.

$$\forall j,k \qquad \sum_{i} X_{ijk} \leq 1 \tag{2}$$

D_i is the duration of job i. Constraint 3 shows that a job will be assigned to resources if it is executed fully. Otherwise it will not be assigned.

$$\forall i \qquad \sum_{j,k} X_{ijk} - D_i * y_i = 0 \tag{3}$$

y_i is for implementing OR operator and can have value 0 or 1. Constraint 4 shows that a job can be executed only on one parking spot in specific hour slot (no parallelization).

$$\forall i,j \qquad \sum_{k} X_{ijk} \leq 1 \tag{4}$$

Constraint 5 determines the starting time of job i.

$$\forall i,j,k \qquad X_{ijk} * S_i \leq j * X_{ijk} \tag{5}$$

S_i is starting time for job i.

Starting time of jobs are used in constraint 12 to avoid interruption in job execution. Constraint 5 is not linear constraint. We replace a set of linear constraints that enforce constraint 5. Followings are constraints that are used to replace constraint 5. We define a new variable W_{ijk} that replaces the product of two variables X_{ijk} and S_i.

$$W_{ijk} = X_{ijk} * S_i \tag{6}$$

$$W_{ijk} - X_{ijk} * j \leq 0 \tag{7}$$

$$W_{ijk} - X_{ijk} * NHS \leq 0 \tag{8}$$

$$W_{ijk} - S_i \leq 0 \tag{9}$$

$$S_i - NHS + NHS * X_{ijk} - W_{ijk} \leq 0 \tag{10}$$

$$W_{ijk} \geq 0 \tag{11}$$

NHS is number of hour slots.

Following constraint is for executing the units of job continuously.

$$\forall i \qquad \sum_{j,k} X_{ijk} * j - S_i D_i = 0.5(D_i)^2 - 0.5D_i \tag{12}$$

Equation 13 is second part of objective function. In our model we minimize equation 13 in order to minimize the number migrations.

$$\forall i \qquad \sum_{i} (Max_i - Min_i) \tag{13}$$

Max_i is the variable that defines the parking spot with the highest ID number which has been assigned to job i. Min_i is the variable that defines the parking spot with the lowest

ID number which has been assigned to job i. In second part of objective function we try to minimize the distance between Max_i variable and Min_i variable. By minimizing the second part of this objective function, number of the parking spots that have been assigned to job i will decrease. This will decrease the number of migrations. In other words a job tends to be executed on minimum number of parking spots. Here we define the Min_i and Max_i:

$$Min_i * X_{ijk} \leq X_{ijk} * k \tag{14}$$

$$Max_i * X_{ijk} \geq X_{ijk} * k \tag{15}$$

Constraint 14 will enforce the definition of lowest parking spot ID to Min_i. Constraint 15 will enforce the definition of highest parking spot ID to Max_i. Since constraint 14 and 15 are not linear we define a set of constraints to convert constraint 14 and 15 to linear constraints. We define a new variable Q_{ijk} that replaces the product of two variables X_{ijk} and Min_i. Following are the constraints for converting constraint 14 to linear constraint:

$$Q_{ijk} = X_{ijk} * Min_i \tag{16}$$

$$Q_{ijk} - X_{ijk} * k \leq 0 \tag{17}$$

$$Q_{ijk} - X_{ijk} * NM \leq 0 \tag{18}$$

$$Q_{ijk} - Min_i \leq 0 \tag{19}$$

$$Min_i - NM + NM * X_{ijk} - Q_{ijk} \leq 0 \tag{20}$$

$$Q_{ijk} \geq 0 \tag{21}$$

NM is number of machines.

We define a new variable R_{ijk} that replaces the product of two variables X_{ijk} and Max_i. Following are the constraints for converting constraint 15 to linear constraint:

$$R_{ijk} = X_{ijk} * Max_i \tag{22}$$

$$-R_{ijk} + X_{ijk} * k \leq 0 \tag{23}$$

$$R_{ijk} - X_{ijk} * NM \leq 0 \tag{24}$$

$$R_{ijk} - Max_i \leq 0 \tag{25}$$

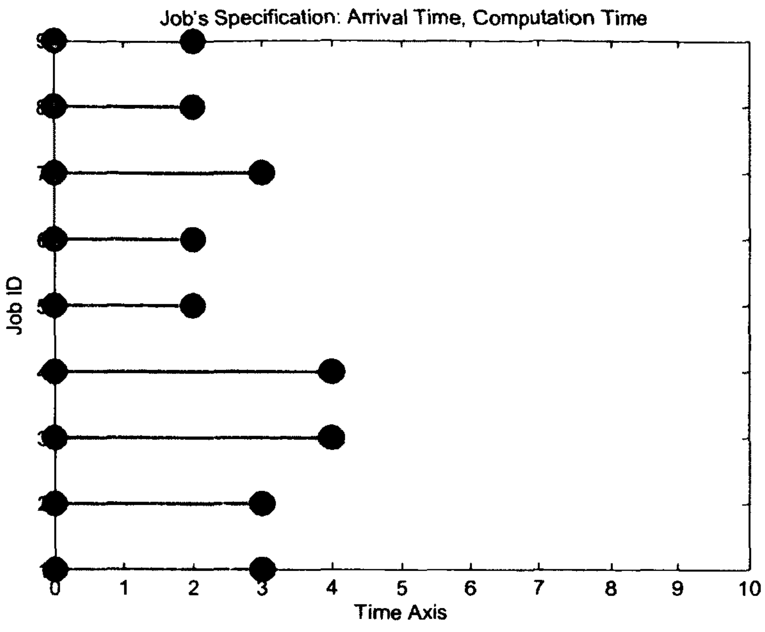


Fig. 11: Scheduling 0-1 Model (Jobs' Specification)

$$Max_i - NM + NM * X_{ijk} - R_{ijk} \leq 0 \tag{26}$$

$$R_{ijk} \geq 0 \tag{27}$$

3.4 IMPLEMENTATION AND EXPERIMENT RESULTS

We implemented the proposed MILP model in Matlab and utilized the Gurobi optimization [60] through its Matlab interface to solve the scheduling problem in VC. Our model prevents interruptions that may occur due to departure of vehicles by using live migration. Figure 11, 12 and 13 shows a scheduling problem for nine jobs. Figure 11 is job specifications that shows duration of each job. Figure 12 shows the occupancy of three parking spots during ten hours. Color red defines that there is not vehicle in that parking spot for that specific hour. Figure 13 is the output of the scheduling model for jobs considering three parking spots during ten hour slots. Jobs are distinguished by their ID that are shown in Figure 13. Zero means there is no job assigned to that parking spot in that specific hour slot. Result in Figure 13 shows that there is a migration for the job with ID two at the end of hour slot eight from parking spot two to parking spot one.

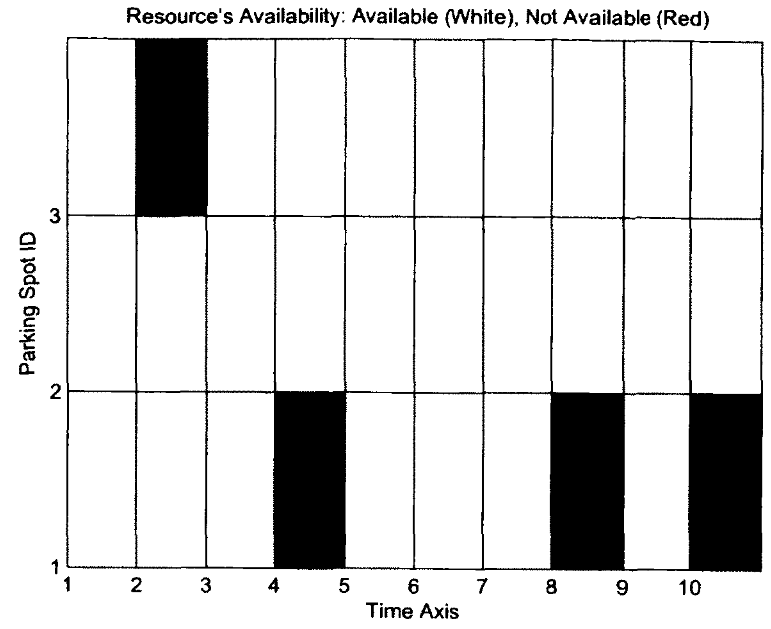


Fig. 12: Parking Occupancy Model

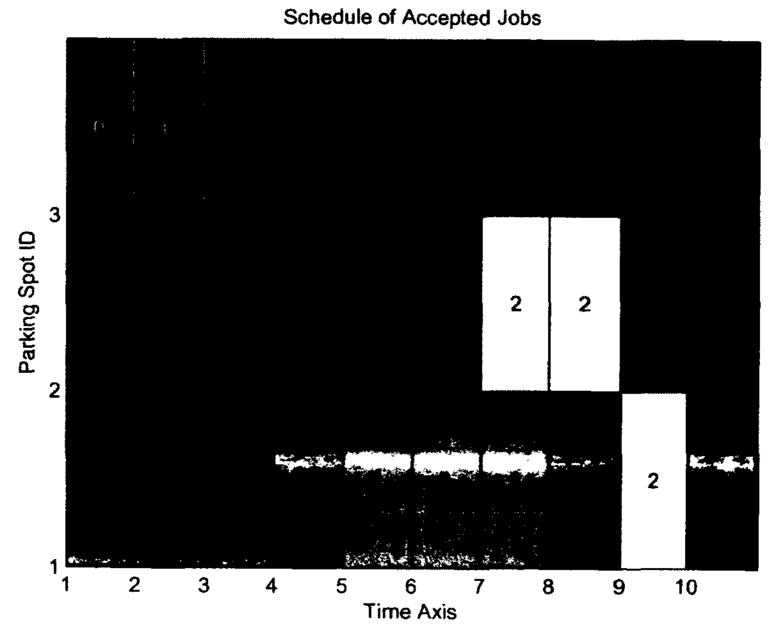


Fig. 13: Scheduling 0-1 Model (Result)

3.5 CONCLUSION

In this chapter we presented a scheduling model for vehicular cloud established in parking lot. We assumed that the arrival and departure of cars are known. This scenario happens in environments like university and hospitals that employees should be present at work with known schedule. In this case university or hospital can rent out its employees' car as a computation resource to provide IT services as a vehicular cloud. Known schedule creates a deterministic environment in terms of resource availability.

Our model is based on Linear Programming. In Linear Programming, model is separated from solving algorithms. Model can be changed easily with changing the constraints without need to change the solving algorithm. This will allow to present different version of the problem in terms of degrees of freedom.

In our model, we assumed that jobs are non-primitive tasks with known duration times. In non-preemptive task, once a task has been given the resource it cannot be taken away from that task. We defined the model based on linear constrains. Departure of cars will cause interruption in executing jobs. In our model in order to prevent interruption, we allow migration. Migration is transferring virtual machine from one physical machine to different physical machine. Our model minimizes the number of migrations and maximizes the resource utilization. Simulations results have shown that our model provides optimum and near optimum job schedule.

CHAPTER 4

FAULT-TOLERANT JOB ASSIGNMENT

4.1 INTRODUCTION AND MOTIVATION

A fundamental way in which VCs differ from conventional clouds is in the ownership of resources. In VCs the ownership of the computational resources is distributed over a large population, as opposed to a single owner as in the conventional clouds run by Amazon, Google, IBM, etc. A corollary of this is that the resources of the VC are highly dynamic as cars may leave unexpectedly. Thus, any job assignment will have to feature some form of fault tolerance.

Fault tolerance was a perennial concern in distributed systems and many solutions have been proposed over the years. Fault tolerance remains a crucial issue in cloud computing and a substantial number of papers have proposed solutions for static environments [61–65].

One of the classic solutions is referred as *checkpointing*, where the state of the computation is saved periodically and the most recent such image is used to roll back the computation in the case of a system failure. Unfortunately, this kind of roll-back recovery is very expensive to implement in VCs because, among others, of the low bandwidth connections. Consider a vehicular cloud established in parking lot, with typical approach periodic checkpoints should be saved on a central server. In this approach vehicles should communicate with server to save the checkpoints. Interaction of large number of nodes with server in same time to save checkpoints will cause low bandwidth connection problem.

In this chapter, we propose to make job assignment in VC fault tolerant by using a variant of the checkpointing strategy. Rather than saving the state of the computation at regular times, the state of the computation is only recorded as needed. Also, since we do not assume a central server that stores checkpointed images like conventional cloud providers do [43,44], in our strategy checkpointing is performed by a car and the resulting image is stored by the car itself. Once the car leaves, the image is lost. We consider two scenarios: in the first one cars do not leave during checkpointing; in the second one cars may leave during checkpointing, leading to system failure.

Our main contribution is to offer theoretical predictions of the job execution time in both scenarios mentioned above. A comprehensive set of simulations have shown that our theoretical predictions are accurate.

The remainder of this chapter is organized as follows. In Section 4.2 we briefly survey relevant work on vehicular clouds; in Section 4.3 we state the main contributions of this work. Next, Section 4.4 offers the details of our fault-tolerant job assignment to cars. Section 4.5 discusses the scenario where cars do not leave during a checkpoint operation and offers analytical performance predictions in this case. In Section 4.6 we allow cars to leave during a checkpoint operation, leading to a system failure. We also offer theoretical performance predictions in this general case. Section 4.7 introduces our simulation model and presents an empirical performance evaluation of our fault-tolerant job assignment scheme through extensive simulation. Finally, Section 4.8 offers concluding remarks.

4.2 RELEVANT PREVIOUS WORK ON VEHICULAR CLOUDS

In order to be able to schedule resources and to assign computational tasks to the various cars in the VC, a fundamental prerequisite is to have an accurate picture of the number of vehicles that are expected to be present in the parking lot as a function of time. What makes the problem difficult is the time-varying nature of the arrival and departure rates. Quite recently, Arif *et al.* [4] have proposed a stochastic prediction model for the parking occupancy given time-varying arrival and departure rates. They provided closed forms for the probability distribution of the parking lot occupancy as a function of time, for the expected number of cars in the parking lot and its variance, and for the limiting behavior of these parameters as time increases. In addition to analytical results, they have obtained a series of empirical results that confirm the accuracy of the analytical predictions.

4.3 OUR CONTRIBUTIONS

As mentioned before, the distributed ownership of the computational resources makes it very challenging to assign, with any degree of confidence, jobs to resident vehicles. To get a feel for the problem, assume that we just assigned a job to a car currently in the parking lot. If the car stays in the parking lot until the job terminates, all is well.

The difficulty arises when the car departs before job completion. In such a case, unless we take special precautions, the entire work done is lost and we have the restart the entire process again, taking chances on another car, and so on until eventually the job is completed.

One possible solution is to set up a checkpointing strategy: this is a policy that prescribes making periodic copies of the state of the computation in such a way that rollback recovery can be used in the case where the car currently running the job leaves. While simple and intuitively satisfying, this first solution is very difficult to implement efficiently. One of the main reasons is that most of these checkpoints are, in fact, unnecessary reducing dramatically the efficiency of the vehicular cloud.

In this work we explore a trade-off. Specifically, we assign each job to two cars and take a checkpoint only when one of the cars leaves the parking lot. As it turns out, this strategy is quite efficient in terms of resource utilization and also competitive in terms of the expected throughput. Extensive simulations have confirmed the accuracy of our theoretical predictions.

4.4 THE JOB ASSIGNMENT STRATEGY J2

Our main goal is to provide a theoretical analysis of the expected job completion time in a VC environment. Consider a vehicular cloud set up in the parking lot of an international airport. We assume that, due to high demand, the parking lot is almost always nearly full and, in particular, it is always possible to find two cars that can be assigned to an incoming job. We assume that, at any moment in time, a car is assigned at most one job and a job is assigned to at most two cars.

Job Assignment Strategy J2: Each job is assigned to two cars selected, at random, from the available cars. When one of the cars departs, the remaining car undergoes a checkpoint period during which its state of the computation is saved; the saved state is then copied to a new car and the job is restarted on both cars.

Refer to Figure 14 for an illustration. Here, the job is started, initially, on cars A and B. When car A leaves, the computation on car B is checkpointed. Once the checkpoint terminated, the job is restarted on cars B and C. At some point, car C leaves. Again the computation on car B is checkpointed and the job is restarted on cars B and D. When car B leaves, the computation on car D is checkpointed and the job is restarted on card D and E. Finally the job terminated before cars D and E leave the parking lot.

4.5 WHEN CHECKPOINTS DO NOT FAIL

We begin by analyzing the simplest case where cars do not leave during checkpointing. While unrealistic in general, this scenario occurs predominantly when checkpoint duration is short with respect to car residency times. Our goal is to compute the expected job

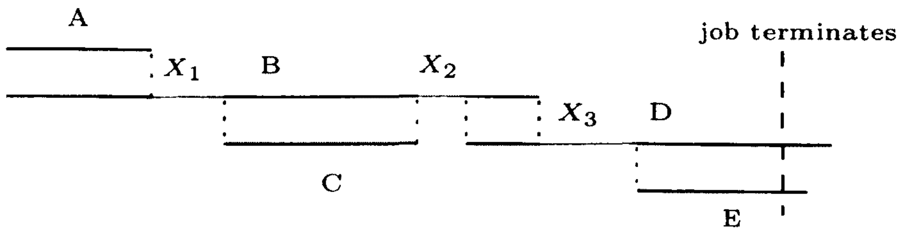


Fig. 14: Illustrating the job assignment and checkpointing strategy.

completion time, including the duration of all checkpoints, in this case.

Here is a list of assumptions we make: The residency time of cars are independent, identically distributed exponential random variables with parameter $\lambda > 0$. In the absence of migrations, the execution time of each job is a random variable T with distribution function G and finite expectation $E[T] < \infty$.

Let there be N checkpoints of durations X_1, X_2, $\cdots$, X_N where N is a random variable that will be discussed shortly. The X_i's, $(1 \leq i \leq N)$, are independent identically distributed (iid) random variables with a common distribution F and finite expectation, $E[X_k] = \mu < \infty$ for $k = 1, 2, \cdots$. Let J be the random variable that keeps track of the total expected execution time of the job, including the duration of all checkpoints, if any. As already mentioned, our goal is to evaluate $E[J]$. In this section we assume that cars do not leave during the checkpointing operation.

It is intuitively clear that the number N of checkpoints is a function of the duration T of the job: the larger T, the more checkpoints are likely to be involved in the execution. The following technical results will be used at several places in the remainder of the chapter. In particular, it will be useful to establish the relationship between T and N.

Lemma 4.5.1. *Let X and Y be independent, exponentially distributed random variables with parameter λ. The random variable $Z = \min\{X, Y\}$ is exponentially distributed with parameter 2λ.*

Proof. For $z \geq 0$ we write

$$\begin{aligned}
\Pr[Z > z] &= \Pr[\min\{X,Y\} > z] \\
&= \Pr[\{Y > z\} \cap \{Y > z\}] \\
&= \Pr[X > z] \Pr[Y > z] \\
&= e^{-\lambda z} e^{-\lambda z} = e^{-2\lambda z}
\end{aligned}$$

and the proof of the lemma is complete. □

Lemma 4.5.1 tells us that the inter-arrival time of checkpoints is exponentially distributed with parameter 2λ and so the the conditional distribution of the random variable N, given that $\{T = t\}$, is Poisson with parameter $2\lambda t$. In other words, for all natural number n

$$\Pr[N = n \mid T = t] = \frac{(2\lambda t)^n}{n!} e^{-2\lambda t}.$$

To get a handle on the probability mass function of the number N of checkpoints, we evaluate $\Pr[N = n]$ by conditioning on T:

$$\begin{aligned}\Pr[N = n] &= \int_{t=0}^{\infty} \Pr[N = n \mid T = t]\, \mathrm{d}G(t) \\ &= \int_0^{\infty} \frac{(2\lambda t)^n}{n!} e^{-2\lambda t}\, \mathrm{d}G(t). \end{aligned} \tag{28}$$

We are now in a position to reveal the relationship between N and T. We proceed as follows:

$$\begin{aligned}E[N] &= \sum_{n=0}^{\infty} n \Pr[N = n] \\ &= \sum_{n=0}^{\infty} n \int_0^{\infty} \frac{(2\lambda t)^n}{n!} e^{-2\lambda t}\, \mathrm{d}G(t) \\ &= \int_0^{\infty} \sum_{n=0}^{\infty} n \frac{(2\lambda t)^n}{n!} e^{-2\lambda t}\, \mathrm{d}G(t) \\ &= 2\lambda \int_0^{\infty} t e^{-2\lambda t}\, \mathrm{d}G(t) \sum_{n-1}^{\infty} \frac{(2\lambda t)^{n-1}}{(n-1)!} \\ &= 2\lambda \int_0^{\infty} t e^{-2\lambda t} e^{2\lambda t}\, \mathrm{d}G(t) \\ &= 2\lambda \int_0^{\infty} t\, \mathrm{d}G(t) \\ &= 2\lambda E[T]. \end{aligned} \tag{29}$$

Next, we are interested in the expected total time spent taking checkpoints (and, of course, restarting the job after checkpoints). For this purpose, we need to evaluate the expectation

$E[X_1+X_2+\cdots+X_N]$. By the law of total expectation we have

$$\begin{aligned}
& E[X_1+X_2+\cdots+X_N] \\
= & \sum_{n=0}^{\infty} E[X_1+X_2+\cdots+X_N \mid N=n]\Pr[N=n] \\
= & E[X_1+X_2+\cdots+X_n]\Pr[N=n] \\
= & \sum_{n=0}^{\infty} nE[X]\Pr[N=n] \\
= & E[X]\sum_{n=0}^{\infty} n\Pr[N=n] \\
= & E[X]E[N] \qquad (30)
\end{aligned}$$

where, recall, $E[X]$ is the expected duration of a checkpoint/restart operation.

Finally, we can write

$$\begin{aligned}
E[J] &= E[T+X_1+X_2+\cdots+X_N] \\
&= E[T]+E[X_1+X_2+\cdots+X_N] \\
&= E[T]+E[X]E[N] \ \text{[by (53)]} \\
&= E[T]+2\lambda E[X]E[T] \ \text{[by (52)]} \\
&= E[T](1+2\lambda E[X]). \qquad (31)
\end{aligned}$$

Observe that the distributions of T and X only occur in (31) through their first moment. This tells us that our result holds for a large number of distributions that share the same first moment.

4.6 THE GENERAL CASE

In this section we allow cars to leave while the checkpoint operation is in progress. Evidently, in such a case the checkpoint will not complete successfully and we say that a *system failure* occurs. The only way to recover from system failure is to restart job execution from scratch.

We inherit the notation and terminology from the previous section. In particular, N is the random variable that counts the number of checkpoints that occur during the execution of a job. We model the general case as a Markov process with states $0,\ 1,\ \cdots,\ N$, such that

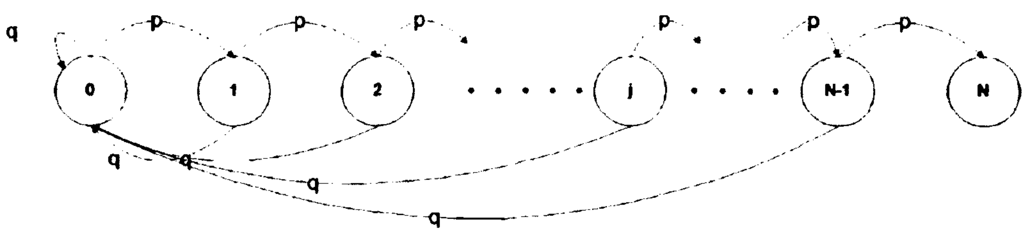


Fig. 15: Markov Process

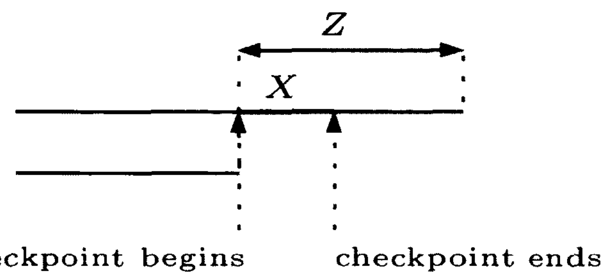


Fig. 16: Illustrating the proof of Lemma 4.6.1.

- the process is initially in state 0;
- the process returns to state 0 upon a system failure;
- the process is in state i, $(1 \leq i \leq N)$, when the first i checkpoints were all successful;
- state N is absorbing.

Recall that car residency times are iid exponential random variables with parameter λ and that checkpoint duration is a random variable with distribution function F and finite expectation μ.

Lemma 4.6.1. *The probability of a successful checkpoint is*

$$p = \int_0^\infty e^{-\lambda u} \mathrm{d}F(u). \tag{32}$$

Proof. Consider what happens when a checkpoint operation is started and refer to Figure 16. Let X be the random variable that denotes the duration of the checkpoint and let Z be the residual life of the residency of the car that is running the checkpoint operation. For the checkpoint to be successful, the event $\{X < Z\}$ must occur. Conditioning on the duration of the checkpoint, we write

$$\begin{aligned}
\Pr[X < Z] &= \int_{x=0}^{\infty} \Pr[X < Z \mid X = x]\,\mathrm{d}F(x) \\
&= \int_{x=0}^{\infty} \Pr[Z > x]\,\mathrm{d}F(x) \\
&= \int_{x=0}^{\infty} e^{-\lambda x}\,\mathrm{d}F(x),
\end{aligned}$$

since it is well known that the residual life of the residency times is exponentially distributed with the same parameter λ. This completes the proof. □

In order to evaluate the expected job completion time in the presence of system failures, we need to define the following costs, that reflect the time elapsed from the previous restart in state 0 until the current system failure.

To understand the cost structure, it is important to be able to assess the *expected* time that the process spends in each state until a successful transition to the next state. Indeed, having reached state i, the process spends an expected $\frac{1}{2\lambda} + \mu$ until it makes the transition to state $i+1$. To see that this is the case, observe that, by Lemma 4.5.1, the expected time until the first car leaves is $\frac{1}{2\lambda}$ to which we add the expected duration of the successful checkpoint.

Let us turn to the cost of a system failure. As will be seen shortly, the cost of a system failure depends on the state in which the failure occurred. Specifically, a system failure that occurs while the process is in state:

- 0 has associated cost $\frac{3}{2\lambda}$. This is because the expected time of the system failure must be $\frac{1}{2\lambda} + \frac{1}{\lambda} = \frac{3}{2\lambda}$. In the previous expression, $\frac{1}{2\lambda}$ is the expected time until one of the cars left and checkpointing was stated, while $\frac{1}{\lambda}$ is the expected departure time of the remaining car. Observe that in the case of a system failure $\frac{3}{2\lambda}$ is smaller that $\frac{1}{2\lambda} + \mu$;

- j, $(j \geq 1)$, the cost consists of

  - the cost of j successful checkpoints, i.e. $\left(\frac{1}{2\lambda} + \mu\right) j$, plus
  - the cost of an unsuccessful transition, which, as we saw is $\frac{3}{2\lambda}$.

Thus, the cost of a system failure that occurs while the process is in state j is

$$\left(\frac{1}{2\lambda}+\mu\right)j+\frac{3}{2\lambda}.$$

Let J be the random variable that keeps track of the total time to complete the job in the presence of system failures. $E[J]$ has the following components

- $E[T](1+2\lambda\mu)p^N$ – the expected completion time when no system failures occur. Indeed, as we saw in Section 4.5 $E[T](1+2\lambda\mu)$ is the expected job completion time if no system failures occur. Since there are N transitions that can fail independently, the probability of this is p^N;
- $\left(\frac{1}{2\lambda}+E[J]\right)(1-p)$ – the expected job completion time in the case of a failed transition from state 0 to state 1;
- $\sum_{j=1}^{N-1}\left[\left(\frac{1}{2\lambda}+\mu\right)j+\frac{3}{2\lambda}+E[J]\right]p^j(1-p)$ – which is the summation of the expected job completion for each j where the transition from state j to $j+1$ fails, while the previous transitions were all successful.

To simplify the notation, we let q denote $1-p$. In this notation, we write

$$\begin{aligned} E[J] &= E[T](1+2\lambda\mu)p^N+\left(\frac{1}{2\lambda}+E[J]\right)q \\ &+ \sum_{j=1}^{N-1}\left[\left(\frac{1}{2\lambda}+\mu\right)j+\frac{3}{2\lambda}+E[J]\right]p^jq \\ &= E[T](1+2\lambda\mu)p^N+\frac{3}{2\lambda}q+E[J]q \\ &+ \sum_{j=1}^{N-1}\left(\frac{1}{2\lambda}+\mu\right)jp^jq+\sum_{j=1}^{N-1}\left(\frac{3}{2\lambda}+E[J]\right)p^jq \\ &= E[T](1+2\lambda\mu)p^N+\sum_{j=1}^{N-1}\left(\frac{1}{2\lambda}+\mu\right)jp^jq \\ &+ \sum_{j=0}^{N-1}\left(\frac{3}{2\lambda}+E[J]\right)p^jq. \end{aligned} \tag{33}$$

Tedious, albeit straightforward computations confirm that

- $\sum_{j=1}^{N-1}\left(\frac{1}{2\lambda}+\mu\right)jp^jq=\left(\frac{1}{2\lambda}+\mu\right)\frac{p(1-p^N)}{1-p}-\left(\frac{1}{2\lambda}+\mu\right)Np^N$
- $\sum_{j=0}^{N-1}\left(\frac{3}{2\lambda}+E[J]\right)p^jq=\frac{3}{2\lambda}\left(1-p^N\right)+E[J]\left(1-p^N\right)$

Upon replacing these expressions in (33) we obtain

$$\begin{aligned} E[J] &= E[T](1+2\lambda\mu)p^N + \left(\frac{1}{2\lambda}+\mu\right)\frac{p(1-p^N)}{1-p} \\ &- \left(\frac{1}{2\lambda}+\mu\right)Np^N \\ &+ \frac{3}{2\lambda}(1-p^N) + E[J](1-p^N) \end{aligned}$$

After a number of elementary algebraic manipulations, we obtain the following expression for $E[J]$.

$$\begin{aligned} E[J] &= \left[E[T](1+2\lambda\mu) - \left(\frac{1}{2\lambda}+\mu\right)N\right] \\ &+ \left(\frac{1}{2\lambda}+\mu\right)\frac{p(1-p^N)}{p^N(1-p)} + \frac{3}{2\lambda}\frac{1-p^N}{p^N} \end{aligned} \tag{34}$$

We claim that as $N \to E[N]$

$$\left[E[T](1+2\lambda\mu) - \left(\frac{1}{2\lambda}+\mu\right)N\right] \to 0.$$

To see that this is the case, recall that by (29) $E[N] = 2\lambda E[T]$. Since $E[N]$ is finite, the values of N are *concentrated* around $E[N]$. thus, as N approaches $E[N]$,

$$\begin{aligned} & E[T](1+2\lambda\mu) - \left(\frac{1}{2\lambda}+\mu\right)N \\ \to\ & E[T](1+2\lambda\mu) - E[T](1+2\lambda\mu) = 0 \end{aligned}$$

With the previous observation, (34) yields the approximation

$$\begin{aligned} E[J] &\approx \frac{3}{2\lambda}\left(\frac{1}{p^{E[N]}}-1\right) + \left(\frac{1}{2\lambda}+\mu\right)\frac{\frac{1}{p^{E[N]}}-1}{\frac{1}{p}-1} \\ &= \left(\frac{1}{p^{E[N]}}-1\right)\left[\frac{3}{2\lambda} + \frac{\frac{1}{2\lambda}+\mu}{\frac{1}{p-1}}\right] \\ &= \frac{1}{2\lambda}\left(\frac{1}{p^{E[N]}}-1\right)\left[3 + \frac{p(1+2\lambda\mu)}{1-p}\right] \end{aligned}$$

4.7 SIMULATION RESULTS

In addition to the theoretical models developed in Sections 4.5 and 4.6 we have evaluated the performance of our fault-tolerant job assignment strategy through an extensive set of simulations. The goal of this section is to report on our experimental results and to show that they closely match the theoretical predictions obtained previously. Specifically, Section 4.7.1 presents the details of our simulation model used to generate the various diagrams. Later, Section 4.7.2 summarizes our findings in the form of various diagrams.

4.7.1 SIMULATION MODEL

In this subsection, we present the details of our simulation model. We assume a large parking lot, similar to a typical parking lot in the downtown area of a large city. Car residency time are assumed to be exponentially distributed with parameter λ. In our simulation λ varied between $\frac{1}{3}$ and $\frac{1}{8}$. Since the expected residency time is given by $\frac{1}{\lambda}$, our choice corresponds to cars spending, on the average, between three and eight hours in the parking lot. Also, we assumed that the number of available cars are sufficiently large, so that upon one car departing, a substitute could be found without delay.

Jobs durations are exponentially distributed random variables with parameter γ. In our simulation γ varied between $\frac{1}{3}$ and $\frac{1}{5}$.

Upon one of the car assigned to a job departing before job termination, the remaining car enters a checkpoint. Individual checkpoint duration is a function of the amount of data that needs to be stored and is taken to be exponentially distributed with parameter μ. This means that the average duration of a checkpoint is $\frac{1}{\mu}$. We experimented several scenarios, with average checkpoint durations of 10 minutes.

This model was developed in MATLAB 7.14. In each of the experiments, a job with one of the parameters described above was submitted to the Vehicular Cloud and the duration of job executed recorded. The same experiment was then repeated 10^5 times and the average of these runs were plotted.

4.7.2 SIMULATION RESULTS

This section presents our simulation results performed in order to validate the analytical results obtained in Sections 4.5 and 4.6. We have performed several sets of experiments that we now describe. In order to evaluate the impact of the various problem parameters discussed in Subsection 4.7.1 on the expected duration of job execution (including checkpointing and, perhaps, system failure) we have conducted a number experiments corresponding to the scenarios that we detail next.

In the first set of experiments, the model has been set up in such a way that system failures due to cars leaving during checkpointing will not occur.

- Scenario 1: In the first scenario we assumed that under ideal conditions (i.e. no checkpoints necessary), the average job duration is three hours. We further assumed an average checkpoint duration of 10 minutes;

- Scenario 2: In this scenario we assume that the average duration of the jobs are 3 hours and the average checkpoint duration is 15 minutes.

- Scenario 3: In this scenario we assume that the average duration of the jobs are 3 hours and the average checkpoint duration is 20 minutes.

- Scenario 4: In order to see the impact of longer job duration, in the second scenario we assumed an average job duration of five hours and an average checkpoint duration of 10 minutes.

- Scenario 5: In this scenario we assume that the average duration of the jobs are 5 hours and the average checkpoint duration is 15 minutes.

- Scenario 6: In this scenario we assume that the average duration of the jobs are 5 hours and the average checkpoint duration is 20 minutes.

- Scenario 7: In order to see the impact of longer job duration, in the second scenario we assumed an average job duration of eight hours and an average checkpoint duration of 10 minutes.

- Scenario 8: In this scenario we assume that the average duration of the jobs are 8 hours and the average checkpoint duration is 15 minutes.

- Scenario 9: In this scenario we assume that the average duration of the jobs are 8 hours and the average checkpoint duration is 20 minutes.

The results of these scenarios are featured in Figures 17-25. In all scenarios described, the simulation results confirmed the accuracy of the theoretical predictions, the results being, essentially, indistinguishable.

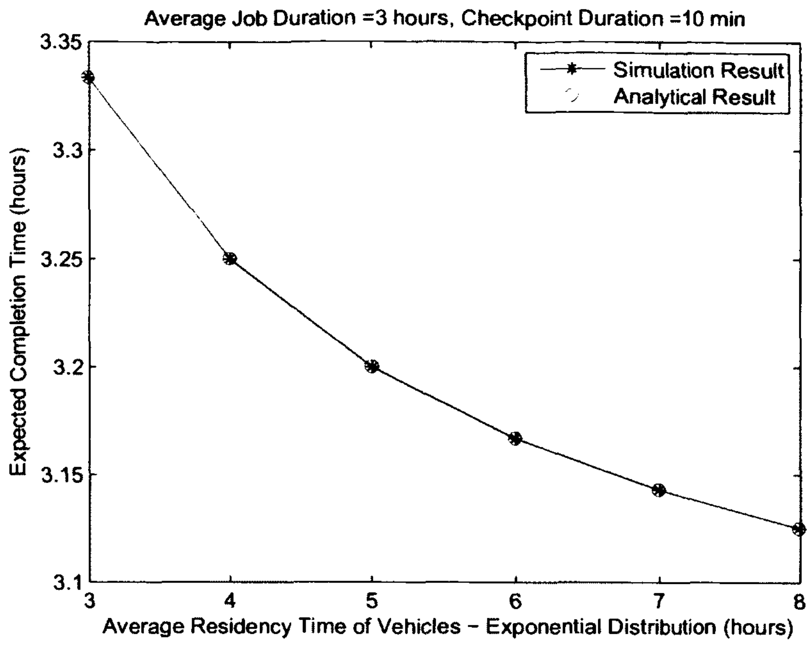


Fig. 17: Scenario 1. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed

In the second set of experiments, system failure was allowed to occur, in other words cars were allowed to leave during checkpointing. Recall, that in the case of a system failure, the execution of the job has to be restarted from scratch.

In order to compare analytical results to simulation results, we consider two scenarios in the second set of simulations for Section 4.6:

- Scenario 1: In this scenario we assume that the average duration of the jobs are 3 hours and the average checkpoint duration is 10 minutes.
- Scenario 2: In this scenario we assume that the average duration of the jobs are 5 hours and the average checkpoint duration is 10 minutes.

Figure 26-27 are the comparison of simulation results and analytical results. Comparison shows that analytical approach and simulation results are consistent.

4.8 CONCLUDING REMARKS

Vehicular clouds are motivated by the abundant computational resources in present-day vehicles and the fact that most of these vehicles are parked every day, for hours on end,

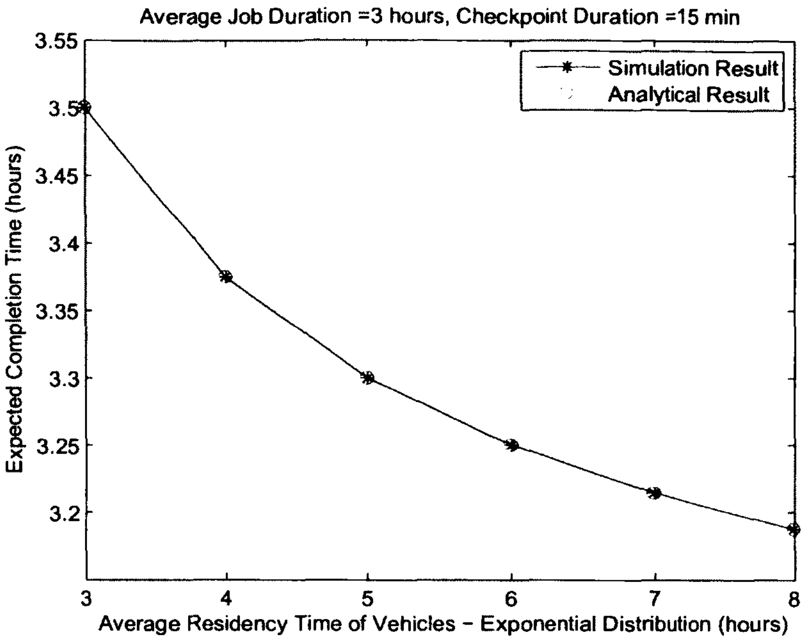


Fig. 18: Scenario 2. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed

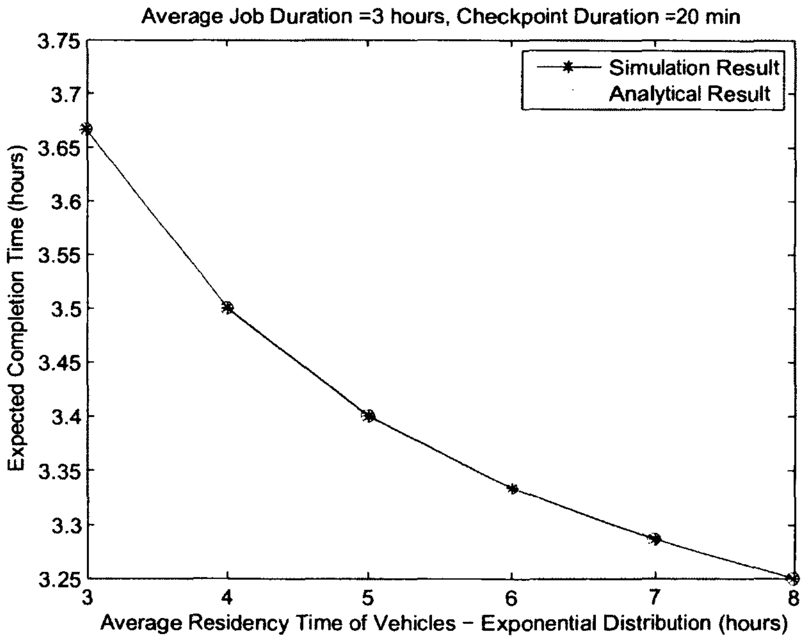


Fig. 19: Scenario 3. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed

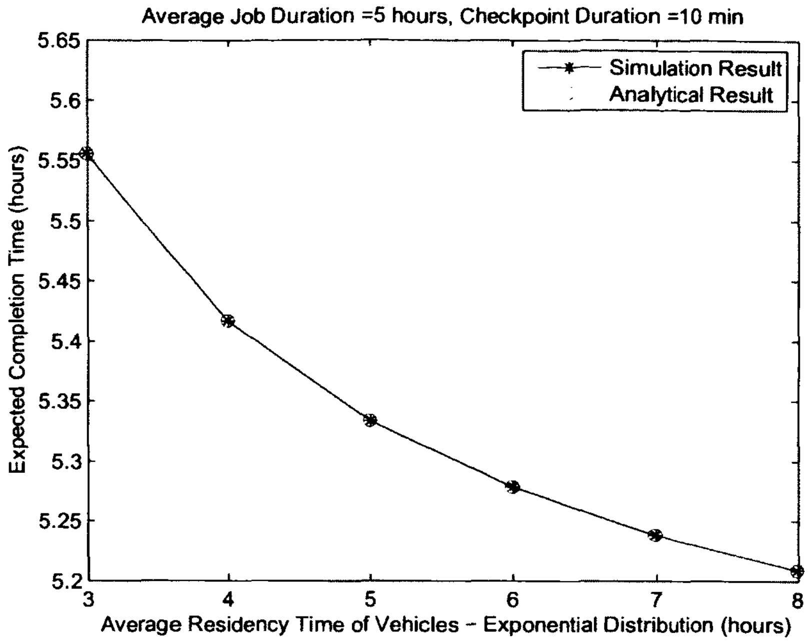


Fig. 20: Scenario 4. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed

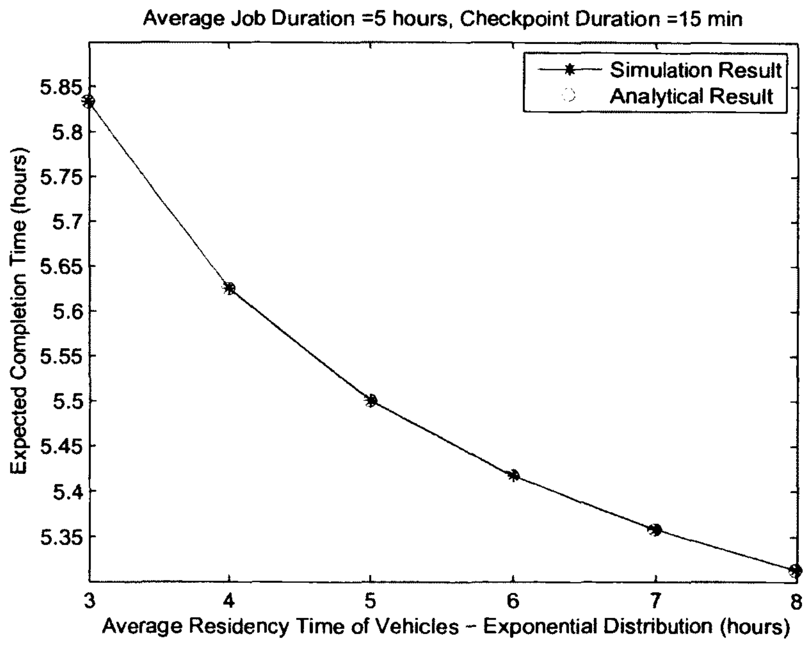


Fig. 21: Scenario 5. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed

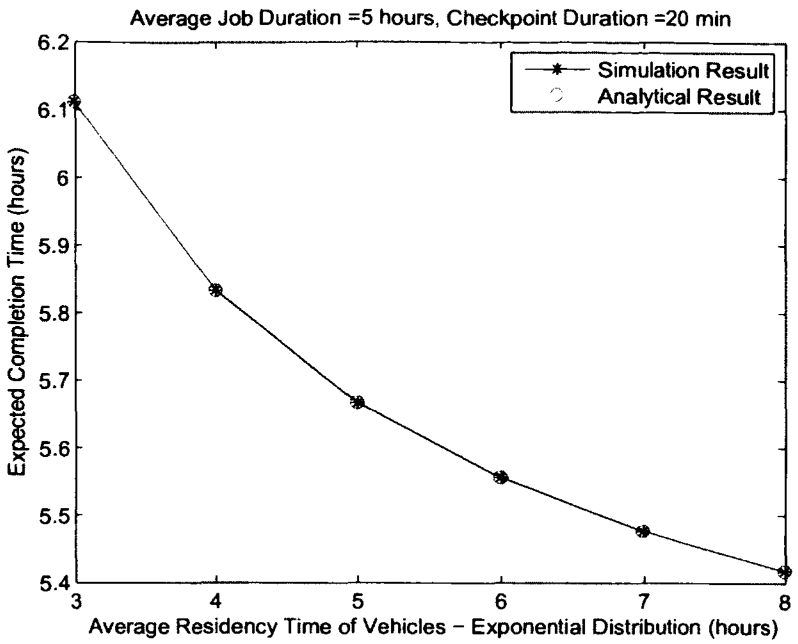


Fig. 22: Scenario 6. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed

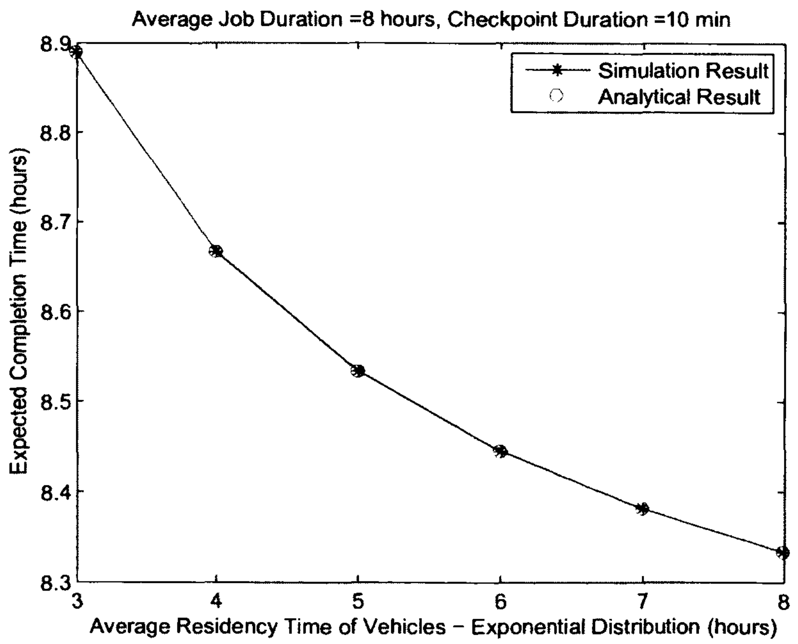


Fig. 23: Scenario 7. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed

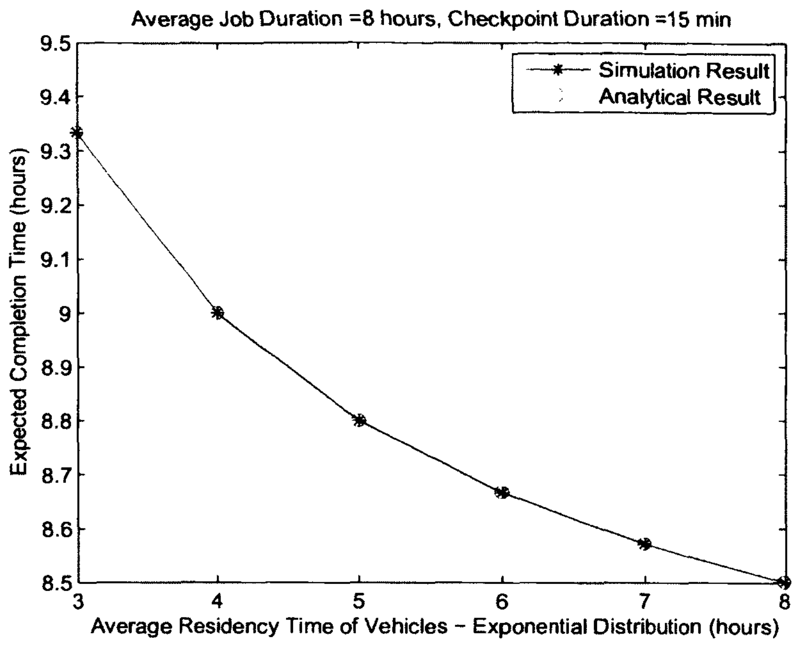


Fig. 24: Scenario 8. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed

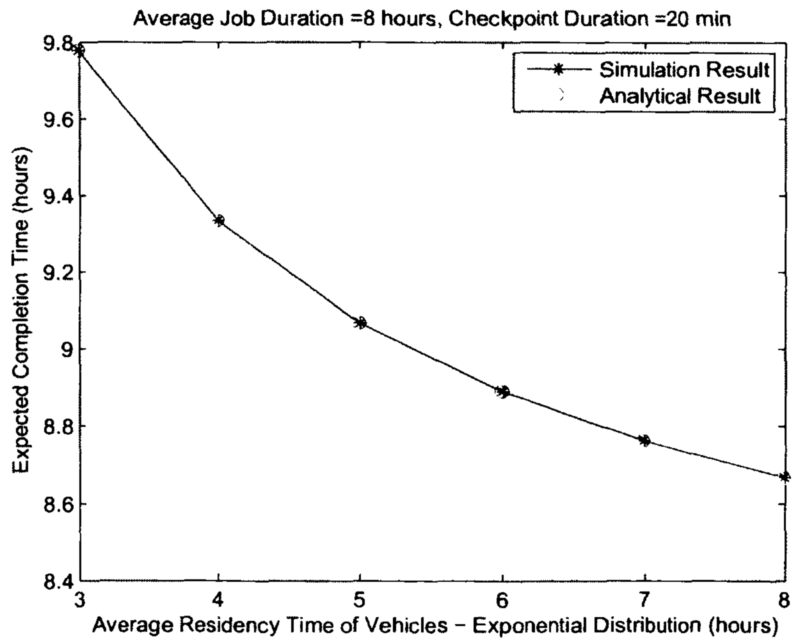


Fig. 25: Scenario 9. Illustrating Job Completion Time J2 Strategy Checkpoint Failure not Allowed

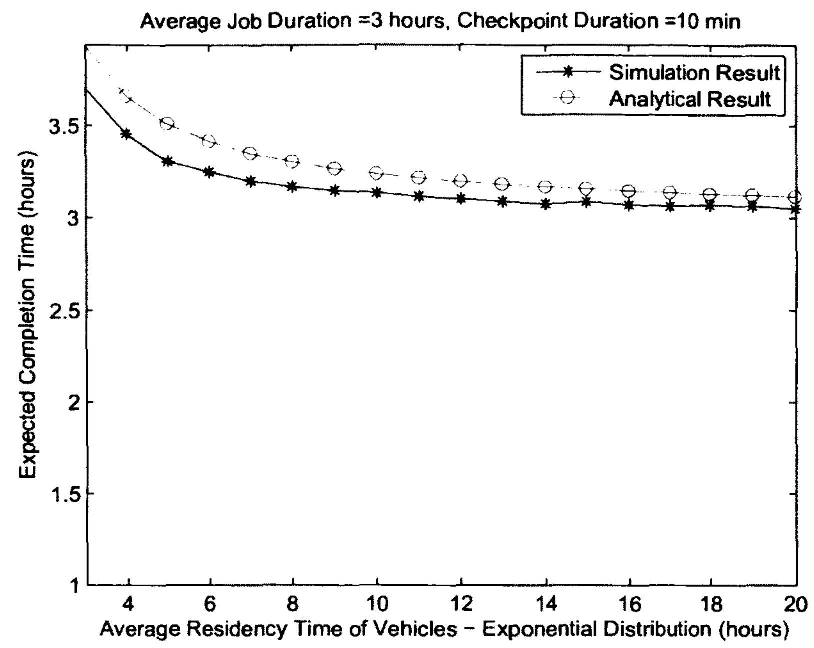


Fig. 26: Scenario 1. Illustrating Job Completion Time J2 Strategy Checkpoint Failure Allowed

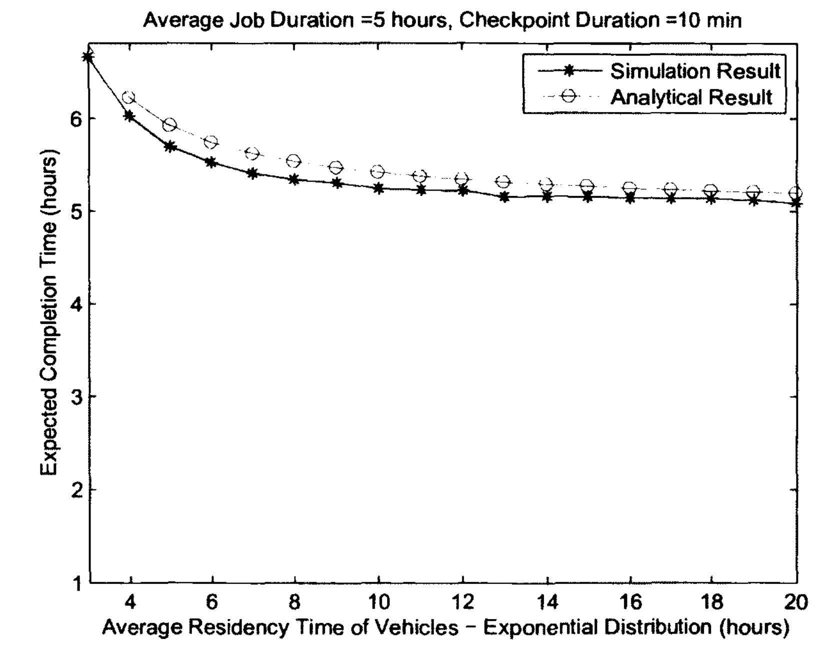


Fig. 27: Scenario 2. Illustrating Job Completion Time J2 Strategy Checkpoint Failure Allowed

while their owner is working, shopping, travelling, etc. Given the huge number of vehicles on our roads and city streets, vehicular clouds are expected to have a huge societal impact.

In this chapter we have proposed a job assignment strategy in vehicular clouds and have analyzed its performance both analytically and by simulation.

Because of the random arrival and departure of cars, resource availability is highly unpredictable. Traditional approach like periodic checkpointing is not applicable because cars should be communicating with central server constantly in order to save checkpoint on server or copy the checkponit from server. In large computation system with hundreds of nodes, this communication is not possible or it creases high overload on system. We present an approach based on redundancy and checkpointing to handle this dynamic environment. In our approach the checkpoint will be saved on the car rather than server. In our approach a job is assigned to two cars at random. Once one of the cars leaves, the other car saves the current state of computation as a checkpoint. Once the checkpoint is finished successfully, another car will be recruited. Job will be restated on both cars and the process continues until the job is completed. We consider two cases. In first case we assume that car does not leave during checkpointing and in the second case we assume that car may leave during checkpointing which leads to failure in system. We present theoretical prediction of job completion time for both cases. Simulation results show the accuracy of theoretical prediction.

CHAPTER 5

MEAN TIME TO FAILURE

In Section 4.4 we presented J2 strategy for job assignment. In this chapter we propose another fault-tolerant job, J3, assignment strategy, based on redundancy, that mitigates the effect of resource volatility of resource availability in vehicular clouds. We offer a theoretical analysis of the mean time to failure of these strategies. A comprehensive set of simulations have confirmed the accuracy of our theoretical predictions.

5.1 INTRODUCTION AND MOTIVATION

One of the recognized metrics of fault tolerance is the mean time to failure (MTTF) defined as the expectation of the random variable that keeps track of the occurrence of a system failure. In this chapter, we investigate the MTTF for two natural job assignment strategies in VCs. Our main contribution is an analytical derivation of the MTTF for each assignment strategy. Extensive simulation results have confirmed the accuracy of our theoretical predictions.

The remainder of this chapter is organized as follows. In Section 5.2 we briefly survey relevant work on vehicular clouds; in Section 5.3 we state the main contributions of this work. Next, Section 5.4 offers the details of our two job assignments. Section 5.6 discusses the MTTF of the first job assignment strategy, while Section 5.7 discusses the MTTF of the second job assignment strategy. Section 5.8 introduces our simulation model and presents an empirical validation of our theoretical predictions. Finally, Section 5.9 offers concluding remarks.

5.2 RELEVANT PREVIOUS WORK ON VEHICULAR CLOUDS

5.3 OUR CONTRIBUTIONS

In this work we explore a trade-off. Specifically, beside the J2 strategy presented in 4.4 we propose another job assignment strategy that we call it J3. In J2, we assign each job to two cars and take a checkpoint only when one of the cars leaves the parking lot. In J3, we assign each job to three cars as will be discussed in detail in Section 5.4. As an evaluation

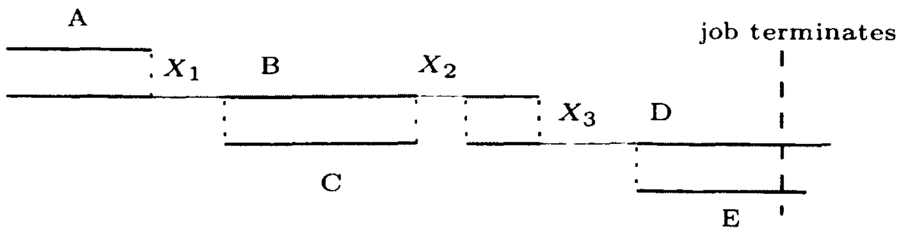


Fig. 28: Illustrating the J2 job assignment strategy.

of the degree of fault tolerance provided by our job assignment strategies, we will provide a detailed theoretical analysis of the mean time to failure of each of J2 and J3. Extensive simulations have confirmed the accuracy of our theoretical predictions.

5.4 THE JOB ASSIGNMENT STRATEGIES

Our main goal is to provide a theoretical analysis of the MTTF in a VC environment. Consider a VC set up in the parking lot of an international airport. We assume that, due to high demand, the parking lot is almost always nearly full and, in particular, it is always possible to find at least two cars that can be assigned to an incoming job. We assume that, at any moment in time, a car is assigned at most one job and a job is assigned to at most two cars.

The J2 Assignment Strategy: We explained this strategy in Section 4.4. Figure 28 shows J2 strategy. For more details refer to Section 4.4.

The J3 Assignment Strategy: The J3 job assignment strategy is very similar to the J2 strategy discussed above, except that each job is assigned to three cars instead of two. When one of the cars leaves, both remaining cars undergo a checkpoint period where they save the state of their respective states of the computation. When one of the remaining cars completes checkpointing, a third car will be recruited and the job will be restarted on all three cars using the saved image. On the other hand, if one of the cars leaves during checkpointing and the remaining car successfully completes checkpointing then two cars will be recruited and the job will be restarted on all three cars using the saved image. In the case where the checkpointing operation does not complete successfully, because both cars leave, a system failure is said to occur.

Intuitively, the strategy J3 is more fault tolerant than J2. In many practical applications where, in addition to fault tolerance, reliability is a major concerned and the result of the computation is accepted on the basis of voting [46, 66–68], J3 is a good candidate. On the other hand, J2 has a higher throughput and could also be used in a voting system.

5.5 BASICS

In the remainder of this chapter we shall assume that car residency times are independent, exponentially distributed random variables with parameter λ and finite expectation. We assume that the duration of checkpoints are independent identically distributed random variables with distribution function G and finial expectation. We further assume that the duration of checkpoints is independent of the residency time of cars.

Cars may leave while a checkpoint operation is in progress. Evidently, in such a case the checkpoint will not complete successfully and we say that a *system failure* occurs. The only way to recover from system failure is to restart job execution from scratch.

Lemma 5.5.1. *Let X and Y be independent, exponentially distributed random variables with parameters λ and μ, respectively. The random variable $Z = \min\{X,Y\}$ is exponentially distributed with parameter $\lambda + \mu$.*

Proof. For $z \geq 0$ we write

$$\begin{aligned}
\Pr[Z > z] &= \Pr[\min\{X,Y\} > z] \\
&= \Pr[\{X > z\} \cap \{Y > z\}] \\
&= \Pr[X > z] \Pr[Y > z] \\
&= e^{-\lambda z} e^{-\mu z} = e^{-(\lambda+\mu)z}
\end{aligned}$$

and the proof of the lemma is complete. □

5.6 THE MTTF FOR THE J2 STRATEGY

The main goal of this section is to determine the MTTF corresponding to the J2 job assignment strategy. Imagine a long-lived job that is running, initially, on two cars, say C_1 and C_2. We adopt the convention that a newly recruited car takes the identity of the departed car. For example, imagine that car C_2 just departed and that car C_1 is running a checkpoint operation. If the checkpoint completes successfully, the newly recruited car will referred to as C_2. Likewise for C_1.

Referring to Figure 29, we model the problem as a Markov process with states C_1C_2, C_1, C_2 and F such that

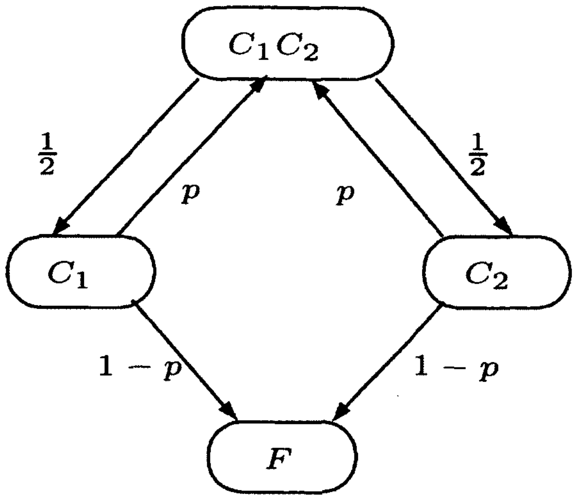


Fig. 29: Illustrating the Markov process corresponding to strategy J2.

- the process is in state C_1C_2 if both cars running a given job are present in the parking lot;
- the process is in state C_1 once car C_2 has left and car C_1 is still in the parking lot;
- the process is in state C_2 when car C_1 has left and car C_2 is still in the parking lot;
- the process is in state F when both cars C_1 and C_2 have left during a checkpoint operation.

We assume that the Markov process is initially in state C_1C_2. Of great interest are the transition rates between these states. Since the car residency times are iid, each of the two cars C_1 and C_2 have an equal likelihood of being the first one to leave, the process makes a transition from state C_1C_2 to states C_1 and C_2 with probability $\frac{1}{2}$.

Next, assume that the process is in state C_1. The process goes back to state C_1C_2 if the checkpointing operation terminates before C_1 leaves. Let the probability of this event be p. In this notation, the transition from state C_1 to state F occurs with probability $1-p$. In order to determine p, recall that, since the residency times are exponentially distributed, the residual residency time of a car is exponentially distributed with the same parameter λ.

Lemma 5.6.1. *The probability of the transition from state C_1 (resp. C_2 to state C_1C_2 is*

$$p = \int_0^\infty e^{-\lambda u} \mathrm{d}G(u). \tag{35}$$

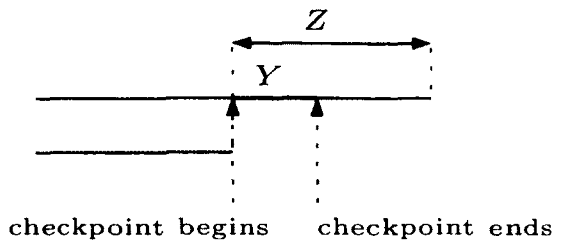


Fig. 30: Illustrating the proof of Lemma 5.6.1.

Proof. Consider what happens when a checkpoint operation is started and refer to Figure 30. Let Y be the random variable that denotes the duration of the checkpoint and let Z be the residual life of the residency of the car that is running the checkpoint operation. For the checkpoint to be successful, the event $\{Z > Y\}$ must occur. Conditioning on the duration of the checkpoint, we write

$$\begin{aligned}\Pr[Z > Y] &= \int_{y=0}^{\infty} \Pr[Z > Y \mid Y = y]\,\mathrm{d}G(y)\\ &= \int_{y=0}^{\infty} \Pr[Z > y]\,\mathrm{d}G(y)\\ &= \int_{y=0}^{\infty} e^{-\lambda y}\,\mathrm{d}G(y).\end{aligned}$$

This completes the proof of Lemma 5.6.1. □

Let T_{12}, T_1, T_2 be the random variables denoting, respectively, the time it takes the Markov process to reach F (i.e., for a system failure to occur), when the process in the states C_1C_2, C_1 and C_2, respectively. In this notation, the MTTF of J2 is simply T_{12}.

By Lemma 5.5.1 the expected amount of time the process spends in state C_1C_2 is $\frac{1}{2\lambda}$. Next, recall that from C_1C_2 the process is equally likely to transit to either C_1 or C_2. Further symmetric considerations tell us that $E[T_1] = E[T_2]$. Thus, using the law of total expectation we write

$$\begin{aligned}E[T_{12}] &= \left[\frac{1}{2\lambda} + E[T_1]\right]\frac{1}{2} + \left[\frac{1}{2\lambda} + E[T_2]\right]\frac{1}{2}\\ &= \frac{1}{2\lambda} + E[T_1]. \quad [\text{since } E[T_1] = E[T_2]] \qquad (36)\end{aligned}$$

Next, let W_1 and W_2 be the random variables denoting the amount of time the process is spending in states C_1 and C_2, respectively, By symmetry, it is clear that W_1 and W_2 and iid with a distribution function that we now discuss. As before, let Y denote the duration of a checkpoint and let Z be the residual residency time of, say, C_1.

$$\begin{aligned}
\Pr[W_1 \le t] &= 1 - \Pr[W_1 > t] \\
&= 1 - \Pr[\min\{Y, Z\} > t] \\
&= 1 - \Pr[\{Y > t\} \cap \{Z > t\}] \\
&= 1 - \Pr[Y > t]\Pr[Z > t] \quad \text{[by independence]} \\
&= 1 - e^{-\lambda t}[1 - G(t)]. \qquad (37)
\end{aligned}$$

Therefore, the expectation $E[W_1]$ of W_1 can be evaluated as follows.

$$\begin{aligned}
E[W_1] &= \int_0^\infty \left[1 - (1 - e^{-\lambda t}[1 - G(t)])\right] \mathrm{d}t \\
&= \int_0^\infty e^{-\lambda t}[1 - G(t)]\,\mathrm{d}t \\
&= \frac{1}{\lambda}\left[1 - \int_0^\infty e^{\lambda u}\,\mathrm{d}G(u)\right] \quad \text{[after integration by parts]} \\
&= \frac{1-p}{\lambda}. \quad \text{[by Lemma 5.6.1]} \qquad (38)
\end{aligned}$$

An identical argument, with C_2 instead of C_1 shows that

$$E[W_2] = E[W_1] = \frac{1-p}{\lambda}.$$

The law of total expectation in conjunction with (38), allows us to write

$$\begin{aligned}
E[T_1] &= E[W_1](1-p) + [E[W_1] + E[T_{12}]p \\
&= \frac{(1-p)^2}{\lambda} + \left[\frac{1-p}{\lambda} + E[T_{12}]\right] p. \qquad (39)
\end{aligned}$$

As before, it is easy to confirm that $E[T_2] = E[T_1]$. This latter observation, in conjunction with (36) and (39), yield a simple system of linear equations that can be solved for $E[T_{12}]$

to yield the MTTF of the J2 job assignment strategy.

$$MTTF_{J2} = E[T_{12}] = \frac{1}{\lambda} + \frac{1}{2\lambda(1-p)}. \tag{40}$$

5.7 THE MTTF FOR THE J3 STRATEGY

In this section we propose to determine the MTTF corresponding to the J3 job assignment strategy. Imagine a long-lived job that is running, initially, on three cars, say C_1, C_2 and C_3. We adopt the convention that a newly recruited car(s) take(s) the identity(ies) of the departed car(s). For example, imagine that car C_2 just departed and that cars C_1 and C_3 are running a checkpoint operation. If one of these cars completes the checkpoint successfully, the newly recruited car will referred to as C_2. If, however, C_1 left the parking lot without completing the checkpoint, assuming that C_3 completes its checkpoint operation successfully, the two recruited cars will be assumed to be C_2 and C_1.

Referring to Figure 31, we model the problem as a Markov process with states $C_1C_2C_3$, C_1C_2, C_1C_3, C_2C_3, D_1, D_2, D_3, D_4, D_5, D_6 and F such that the process is in state

- $C_1C_2C_3$ if all three cars running a given job are present in the parking lot;
- C_1C_2 when car C_3 has left the parking lot;
- C_1C_3 when car C_2 has left the parking lot;
- C_2C_3 when car C_1 has left the parking lot;
- D_1 if the process was in C_1C_2 and car C_2 has left;
- D_2 if the process was in C_1C_2 and car C_1 has left;
- D_3 if the process was in C_1C_3 and car C_3 has left;
- D_4 if the process was in C_1C_3 and car C_1 has left;
- D_5 if the process was in C_2C_3 and car C_3 has left;
- D_6 if the process was in C_2C_3 and car C_1 has left;
- F when all three cars have left during a checkpoint operation.

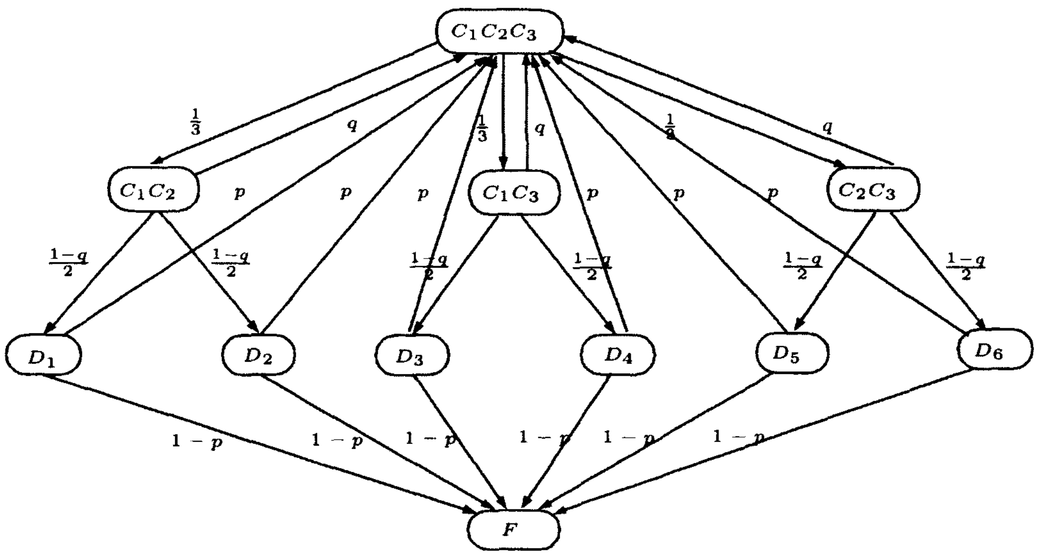


Fig. 31: Illustrating the Markov process corresponding to strategy J3.

We assume that the Markov process is initially in state $C_1C_2C_3$. Of great interest are the transition rates between these states. In spite of the fact that the Markov process corresponding to J3 has more states, the analysis of the transition probabilities is similar to the one of the previous section. Since the car residency times are iid, each of the three cars C_1, C_2 and C_3 have an equal likelihood of being the first one to leave, the process makes a transition from state $C_1C_2C_3$ to each of the states C_1C_2, C_1C_3 and C_2C_3 with probability $\frac{1}{3}$.

Next, assume that the process is in state C_1C_2. The process goes back to state $C_1C_2C_3$ if at least one checkpoint operation terminates before C_1 or C_2 leaves. Let the probability of this event be q. In order to determine q, recall that as already mentioned, the residual residency time of a car is exponentially distributed with parameter λ.

To determine q we let X and Y be the residual residency times of cars C_1 and C_2 and let U be the duration of a checkpoint. In order for the process to return to state $C_1C_2C_3$, the event $\{\min\{X,Y\} > U\}$ must occur. Thus,

$$\begin{aligned} q &= \Pr[\{\min\{X,Y\} > U\}] \\ &= \int_0^\infty \Pr[\{\min\{X,Y\} > U \mid U = u]\, \mathrm{d}G(u) \\ &= \int_0^\infty \Pr[\{\min\{X,Y\} > u]\, \mathrm{d}G(u) \\ &= \int_0^\infty e^{-2\lambda u}\, \mathrm{d}G(u). \text{ [by Lemma 5.5.1]} \end{aligned} \quad (41)$$

By symmetry, (41) guarantees that the probability of the process going from either of the states C_1C_3 and C_2C_3 to state $C_1C_2C_3$ is also q. Notice also that by symmetry, the transition from state C_1C_2 to either of the states D_1 and D_2 occurs with probability $\frac{1-q}{2}$ and that also by symmetry, this matches the probability of a transition from states C_1C_3 to D_3 and D_4 and from state C_2C_3 to D_5 and D_6.

Next, assume that the process is in state D_1, in other words, both cars C_2 and C_3 have left during a checkpoint and only car C_1 remains. If C_1 successfully completes the checkpoint operation, two new cars will be recruited and, consequently, the process makes transition to state $C_1C_2C_3$. We now state a result whose proof mirrors that of Lemma 5.6.1 and is omitted.

Lemma 5.7.1. *The probability of the transition from state D_1 to state $C_1C_2C_3$ is*

$$p = \int_0^{\infty} e^{-\lambda u} \mathrm{d}G(u). \tag{42}$$

Now, a symmetry argument confirms that the transition from any one of the states D_2, D_3, D_4, D_5 and D_6 to state $C_1C_2C_3$ has probability p.

Let W_{123} be the random variable that keeps track of the time the process spends in state $C_1C_2C_3$ before making a transition. Reasoning as in Section 5.6 we have

$$E[W_{123}] = \frac{1}{3\lambda}. \tag{43}$$

Next, let W_{12}, W_{13} and W_{23} denote the time spent in the states C_1C_2, C_1C_3 and C_2C_3 before making a transition. Reasoning as in Section 5.6, we find that

$$E[W_{12}] = E[W_{13}] = E[W_{23}] = \frac{1-q}{2\lambda}. \tag{44}$$

Further, let W_1, W_2, W_3, W_4, W_5 and W_6 denote the time spent in the states D_1, D_2, D_3, D_4, D_5, and D_6 before making a transition. Reasoning as in Section 5.6, we find that

$$E[W_1] = E[W_2] = E[W_3] = E[W_4] = E[W_5] = E[W_6] = \frac{1-p}{\lambda}. \tag{45}$$

Finally, let T_{123}, T_{12}, T_{13}, T_{23}, T_1, T_2, T_3, T_4, T_5 and T_6 be the time it takes the process to reach state F assuming that the process finds itself in states $C_1C_2C_3$, C_1C_2, C_1C_3, C_2C_3, D_1, D_2, D_3, D_4, D_5 and D_6, respectively.

By the law of total expectation we have

$$\begin{aligned} E[T_{123}] &= \frac{E[W_{123}+T_{12}]}{3}+\frac{E[W_{123}+T_{13}]}{3}+\frac{E[W_{123}+T_{23}]}{3} \\ &= E[W_{123}+T_{12}] \text{ [by symmetry]} \\ &= \frac{1}{3\lambda}+E[T_{12}]. \text{ [by (43)]} \end{aligned} \tag{46}$$

Similarly, the law of total expectation along with the observation that $E[T_1] = E[T_2]$ yields

$$\begin{aligned} E[T_{12}] &= E[W_{12}+T_{123}]q+E[W_{12}+T_1]\frac{1-q}{2}+E[W_{12}+T_2]\frac{1-q}{2} \\ &= \left(\frac{1-q}{2\lambda}+E[T_{123}]\right)q+\left(\frac{1-q}{2\lambda}+E[T_1]\right)(1-q) \\ &= qE[T_{123}]+(1-q)E[T_1]+\frac{1-q}{2\lambda}. \end{aligned} \tag{47}$$

A third application of the law of total expectation yields

$$\begin{aligned} E[T_1] &= E[W_1+T_{123}]p+E[W_1](1-p) \\ &= pE[T_{123}]+\frac{1-p}{\lambda}. \end{aligned} \tag{48}$$

Finally, the system of equations (46), (47) and (48) can be solved for $E[T_{123}]$ to yield the $MTTF_{J3} = E[T_{123}]$ of the J3 job assignment strategy.

$$MTTF_{J3} = \frac{1}{\lambda}+\frac{1}{2\lambda(1-p)}+\frac{1}{3\lambda(1-p)(1-q)}. \tag{49}$$

5.8 SIMULATION RESULTS

We have evaluated the accuracy of the MTTF predictions in Sections 5.6 and 5.7 through an extensive set of simulations. The goal of this section is to report on our experimental results and to show that they closely match the theoretical predictions obtained previously. Specifically, Subsection 5.8.1 presents the details of our simulation model. Later, Subsection 5.8.2 summarizes our simulation results.

5.8.1 SIMULATION MODEL

In this subsection, we present the detail of our simulation model. We assume a large parking lot, similar to a typical parking lot in the downtown area of a large city. Car residency time are assumed to be exponentially distributed with parameter λ. In our simulation λ varied between $\frac{1}{3}$ and $\frac{1}{8}$. Since the expected residency time is given by $\frac{1}{\lambda}$, our choice corresponds to cars spending, on the average, between three and eight hours in the parking lot. Also, we assumed that the number of available cars sufficiently large, so that upon one car departing, a substitute could be found without delay.

Our model was developed in MATLAB 7.14. In each of the experiments, a job with one of the parameters described above was submitted to the Vehicular Cloud and the duration of job executed recorded. The same experiment was then repeated 10^5 times and the average of these runs were plotted.

5.8.2 SIMULATION RESULTS

This section presents our simulation results performed in order to validate the analytical results obtained in Sections 5.6 and 5.7. We have performed two sets of experiments that we now describe. In order to evaluate the impact of the various problem parameters discussed in Subsection 5.8.1 on the expected duration of job execution (including checkpointing and, perhaps, system failure) we have conducted a number experiments corresponding to the scenarios that we detail next.

- **Scenario 1:** In the first scenario we considered the J2 job assignment strategy. Each job was assigned to two cars. Checkpoints have an exponential distribution with mean duration of 10 minutes. Car residency times were taken to be exponentially distributed with parameter λ. In our experiments we varied the expected car residency time varies between three and eight hours. The simulation results are featured in Figure 32;

- **Scenario 2:** In this scenario we considered the J2 job assignment strategy. Each job was assigned to two cars. Checkpoints have an exponential distribution with mean duration of 15 minutes. Car residency times were taken to be exponentially distributed with parameter λ. In our experiments we varied the expected car residency time varies between three and eight hours. The simulation results are featured in Figure 33;

- **Scenario 3:** In this scenario we considered the J2 job assignment strategy. Each job was assigned to two cars. Checkpoints have an exponential distribution with mean duration of 20 minutes. Car residency times were taken to be exponentially distributed with parameter λ. In our experiments we varied the expected car residency time varies between three and eight hours. The simulation results are featured in Figure 34;

- **Scenario 4:** In the second scenario we considered the J3 job assignment strategy. Each job was assigned to three cars. Here, the average checkpoint duration was 10 minutes with exponential distribution. Cars residency time were exponentially distributed with parameter λ. We varied the average car residency times between three and eight hours. The actual simulation results are illustrtrated in Figure 35.

- **Scenario 5:** In the second scenario we considered the J3 job assignment strategy. Each job was assigned to three cars. Here, the average checkpoint duration was 15 minutes with exponential distribution. Cars residency time were exponentially distributed with parameter λ. We varied the average car residency times between three and eight hours. The actual simulation results are illustrtrated in Figure 36.

- **Scenario 6:** In the second scenario we considered the J3 job assignment strategy. Each job was assigned to three cars. Here, the average checkpoint duration was 20 minutes with exponential distribution. Cars residency time were exponentially distributed with parameter λ. We varied the average car residency times between three and eight hours. The actual simulation results are illustrtrated in Figure 37.

A glance at Figures 32-37 confirms that the simulation results confirm the accuracy of our theoretical predictions. Analyzing these results show that in both J2 and J3 strategies by increasing the checkpoint duration the mean time to failure will decrease. The main reason is because by increasing the checkpoint duration the probability that the car departs during checkpointing will increase.

Meant time to failure in J3 is longer compared with J2 strategy. The reason is the number of cars that undergo a checkpoint in J3 strategy is larger than J2 strategy. Having more cars taking checkpoint will increases the probability of successful checkpoint. In Chapter 7 we provide comparison result of J2 and J3 strategies in terms of job completion time and mean time to failure.

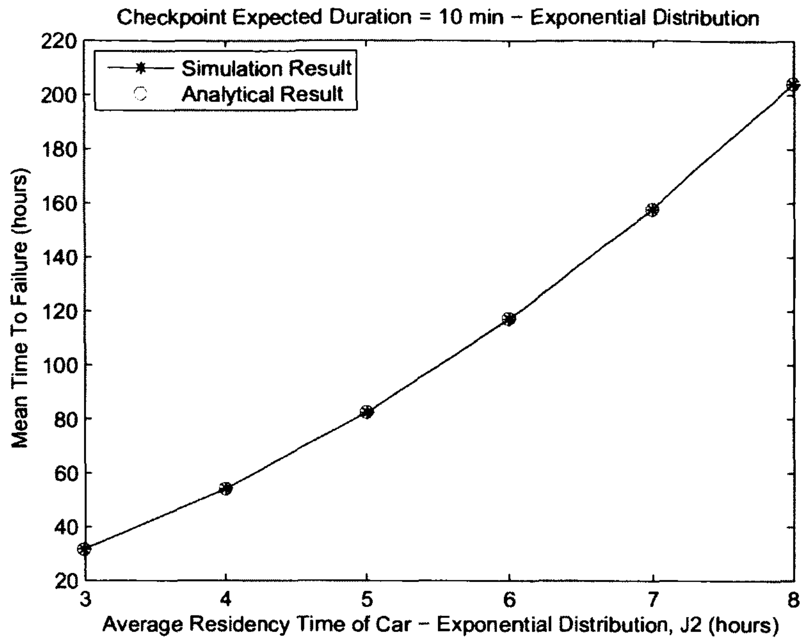


Fig. 32: Scenario 1. Illustrating Mean Time to Failure in J2 Strategy

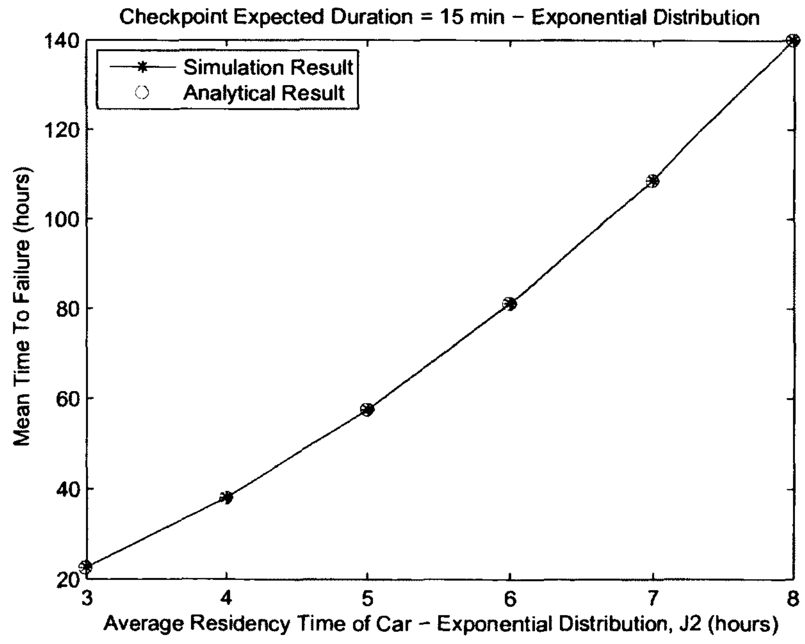


Fig. 33: Scenario 2. Illustrating Mean Time to Failure in J2 Strategy

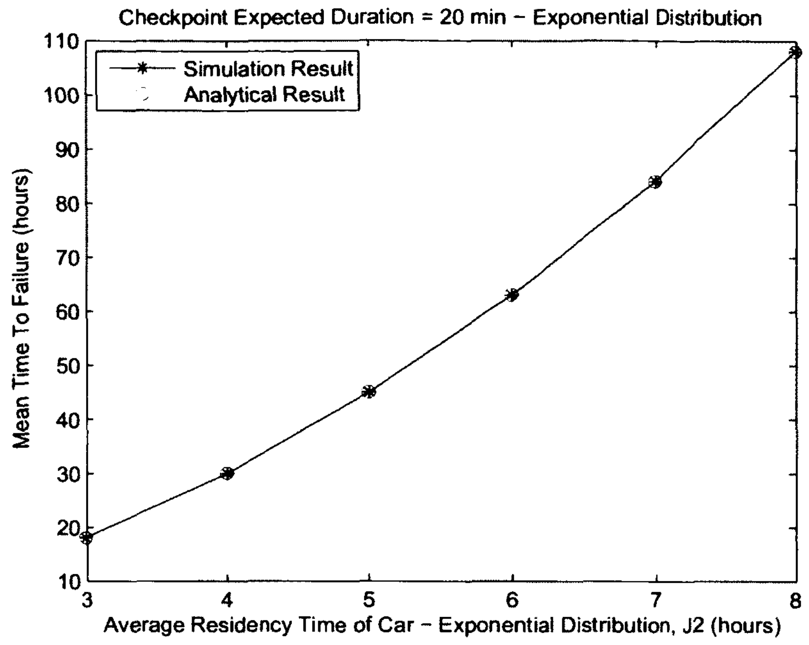


Fig. 34: Scenario 3. Illustrating Mean Time to Failure in J2 Strategy

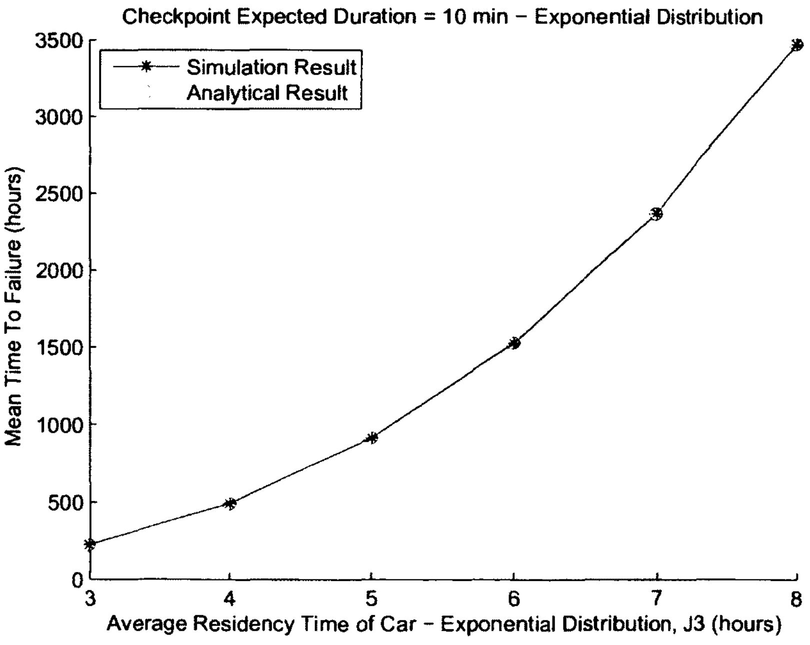


Fig. 35: Scenario 4. Illustrating Mean Time to Failure in J3 Strategy

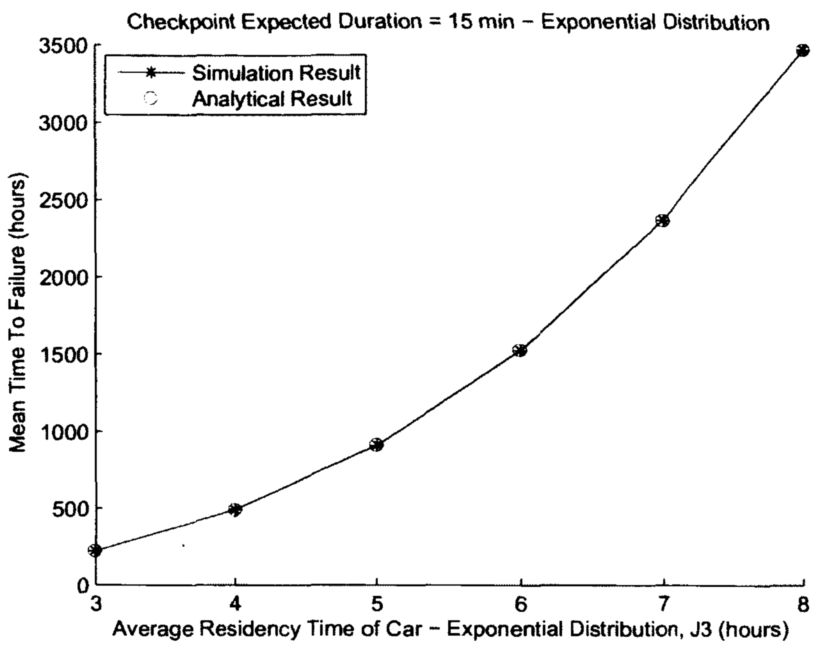


Fig. 36: Scenario 5. Illustrating Mean Time to Failure in J3 Strategy

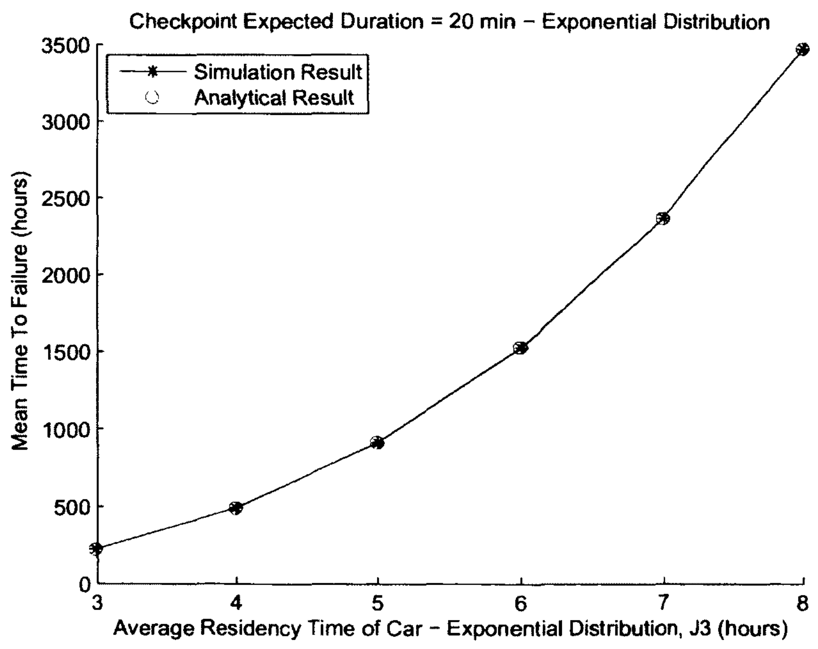


Fig. 37: Scenario 6. Illustrating Mean Time to Failure in J3 Strategy

5.9 CONCLUDING REMARKS

Vehicular clouds are motivated by the abundant computational resources in present-day vehicles and the fact that most of these vehicles are parked every day, for hours on end, while their owner is working, shopping, travelling, etc. Given the huge number of vehicles on our roads and city streets, vehicular clouds are expected to have a huge societal impact. In this chapter we have proposed a job assignment strategy in vehicular clouds and have analyzed its performance both analytically and by simulation.

CHAPTER 6

ONE CAR APPROACH J1

Beside the J2 and J3 strategies that we presented in Sections 4.4 and 5.4, in this chapter we propose the J1 strategy and our main contribution is to offer theoretical predictions of the job execution time.

The goal of this chapter is to provide a benchmark for the two strategies, J2 and J3, that we presented the previous chapters. Later in this chapter we determine the advantage and disadvantage of J1 strategy.

beside completion time that is a criteria for comparison, J2 and J3 strategies have this advantage that

J1 compared to J2 and J3 has a disadvantage of finding optimal checkpoint period.

6.1 OUR CONTRIBUTIONS

Job Assignment Strategy J1: Each job is assigned to one car selected, at random, from the available cars. In this strategy each job is executed on a single car until the car leaves. Every T time units a checkpoint is taken. The checkpoint will be saved on central server. When the car leaves, the computation is rolled back to the last checkpoint and continued from there by a different car.

T is a constant that needs to be differentiated in order to maximize the performance.

- Let there be N checkpoints of durations $X_1, X_2, \cdots, X_N$. The X_i's, $(1 \leq i \leq N)$, are independent identically distributed (iid) random variables with a common distribution F and finite expectation, $E[X_k] = \mu < \infty$ for $k = 1, 2, \cdots$.

- Let J be the random variable that keeps track of the total expected execution time of the job, including the duration of all checkpoints, if any. As already mentioned, our goal is to evaluate $E[J]$.

- Assume that the residency times of cars in the parking lot are iid random variables with an exponential distribution where $\lambda > 0$.

In this approach we assume that cars do not leave during a checkpoint. P is the probability that residency time of car is greater than z.

$$\Pr[Z > z] = 1 - \Pr[Z \leq z] = 1 - (1 - e^{-\lambda z}) = e^{-\lambda z}$$

It is clear that random variable of number of cars that fail until one car stays at least T is geometrically distributed thus:

$$\frac{1}{P} = e^{\lambda z}$$

The amount of time that failed car stays in parking lot:

$$\int_0^T e^{-\lambda z} dz = \frac{1 - e^{-\lambda T}}{\lambda} \tag{50}$$

The time that it takes until one car stays at least for T time:

$$e^{\lambda T} * \frac{1 - e^{-\lambda T}}{\lambda} = \frac{e^{\lambda T} - 1}{\lambda}$$

L is job duration and expected job completion time is:

$$E[J] = \frac{E[L]}{T} * (\frac{e^{\lambda T} - 1}{\lambda} + \mu) + \Delta$$

$$\Delta = L - (\lfloor \frac{L}{T} \rfloor * T)$$

6.2 SIMULATION

In addition to the theoretical models developed in Section 6.1 we have evaluated the performance of our job assignment strategy through an extensive set of simulations. The goal of this section is to report on our experimental results and to show that they closely match the theoretical predictions obtained previously. Specifically, Subsection 6.2.1 presents the details of our simulation model used to generate the various diagrams.

Later, Subsection 6.2.2 summarizes our findings in the form of various diagrams.

6.2.1 SIMULATION MODEL

In this subsection, we present the detail of our simulation model. We assume a large parking lot, similar to a typical parking lot in the downtown area of a large city. Car residency time are assumed to be exponentially distributed with parameter λ. The expected residency time is given by $\frac{1}{\lambda}$. Also, we assumed that the number of available cars sufficiently large, so that upon one car departing, a substitute could be found without delay.

Jobs durations are exponentially distributed random variables with parameter γ. In our simulation γ varied between $\frac{1}{3}$ and $\frac{1}{8}$.

Upon the car assigned to a job departing before job termination, the computation is rolled back to the last checkpoint and continued from there by a different car. Individual checkpoint duration is a function of the amount of data that needs to be stored and is taken to be exponentially distributed with parameter μ. This means that the average duration of a checkpoint is $\frac{1}{\mu}$. We experimented scenarios, namely average checkpoint duration of 10, 15 and 20 minutes.

This model was developed in MATLAB 7.14. In each of the experiments, a job with one of the parameters described above was submitted to the Vehicular Cloud and the duration of job executed recorded. The same experiment was then repeated 10^5 times and the average of these runs were plotted.

6.2.2 SIMULATION RESULTS

This section presents our simulation results performed in order to validate the analytical results obtained in Sections 6.1. We have performed several sets of experiments that we now describe. In order to evaluate the impact of the various problem parameters discussed in Subsection 6.2.1 on the expected duration of job execution we have conducted a number experiments corresponding to the scenarios that we detail next.

In this set of experiments, the model has been set up in such a way that system failures due to cars leaving during checkpointing will not occur.

- Scenario 1: In the first scenario we assumed that under ideal conditions (i.e. no checkpoints necessary), the average job duration is three hours. We further assumed an average checkpoint duration of 10 minutes and average staying time of 3 hours;
- Scenario 2: In the first scenario we assumed that under ideal conditions (i.e. no

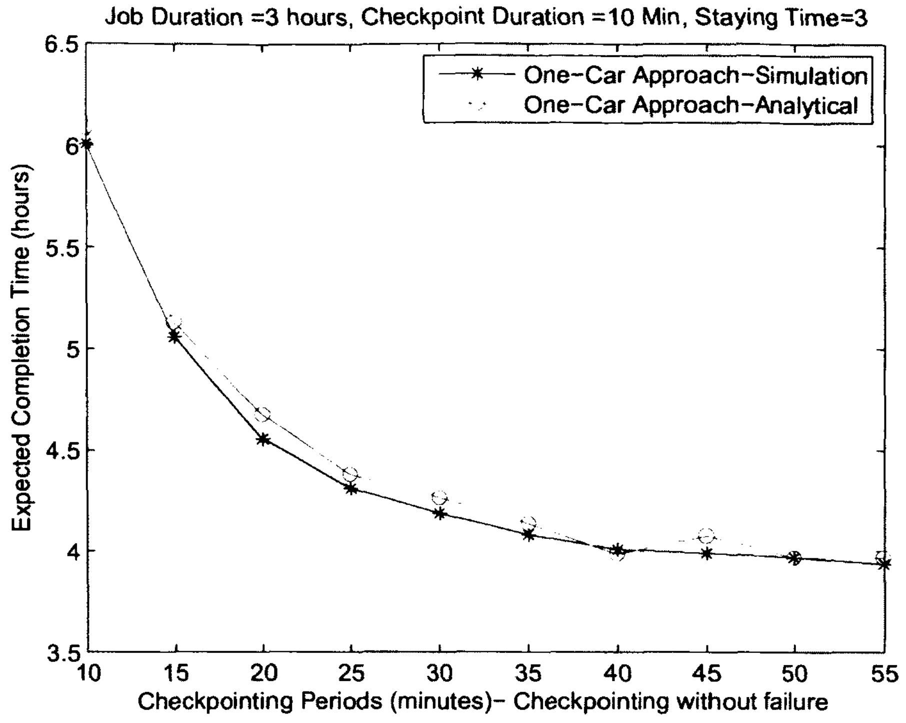


Fig. 38: Scenario 1. Illustrating Job Completion Time in J1 Strategy

checkpoints necessary), the average job duration is three hours. We further assumed an average checkpoint duration of 10 minutes and average staying time of 5 hours;

- Scenario 3: In the first scenario we assumed that under ideal conditions (i.e. no checkpoints necessary), the average job duration is three hours. We further assumed an average checkpoint duration of 10 minutes and average staying time of 8 hours;

Figure 38-40 are the comparison of simulation results and analytical results. Comparison shows that analytical approach and simulation results are consistent.

In the second set of experiments, we compare job completion time in J1 and J2 strategy. In order to compare the results, we consider nine scenarios in the second set of simulations:

- Scenario 1: In this scenario we assume that the average duration of the jobs are 8 hours and the average checkpoint duration is 10 minutes and average staying time of 3 hours.

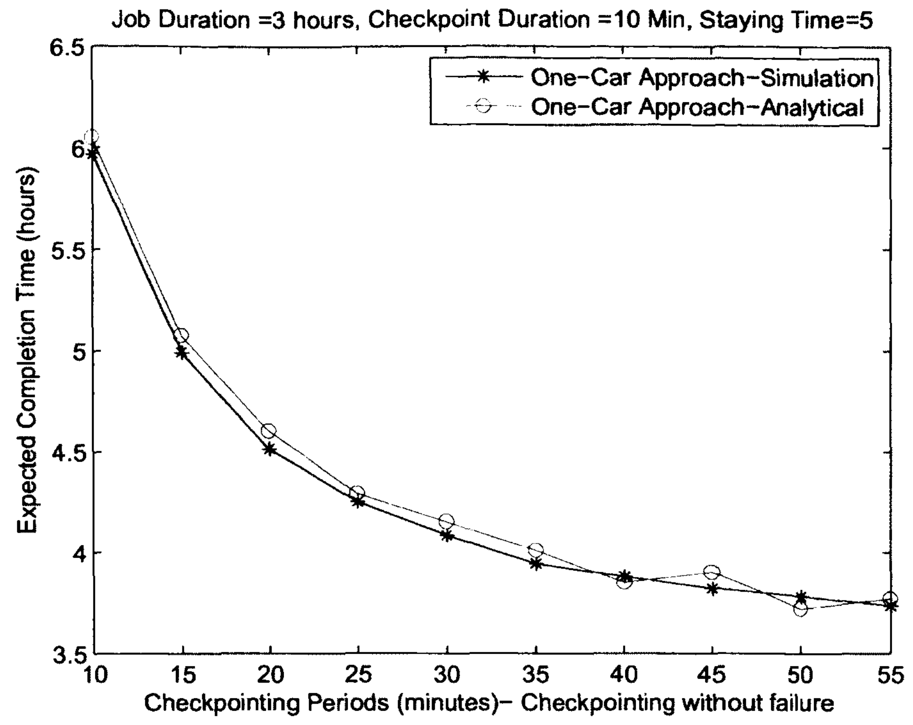


Fig. 39: Scenario 2. Illustrating Job Completion Time in J1 Strategy

- Scenario 2: In this scenario we assume that the average duration of the jobs are 8 hours and the average checkpoint duration is 10 minutes and average staying time of 8 hours.
- Scenario 3: In this scenario we assume that the average duration of the jobs are 3 hours and the average checkpoint duration is 10 minutes and average staying time of 3 hours.
- Scenario 4: In this scenario we assume that the average duration of the jobs are 3 hours and the average checkpoint duration is 10 minutes and average staying time of 4 hours.
- Scenario 5: In this scenario we assume that the average duration of the jobs are 3 hours and the average checkpoint duration is 10 minutes and average staying time of 5 hours.
- Scenario 6: In this scenario we assume that the average duration of the jobs are .5

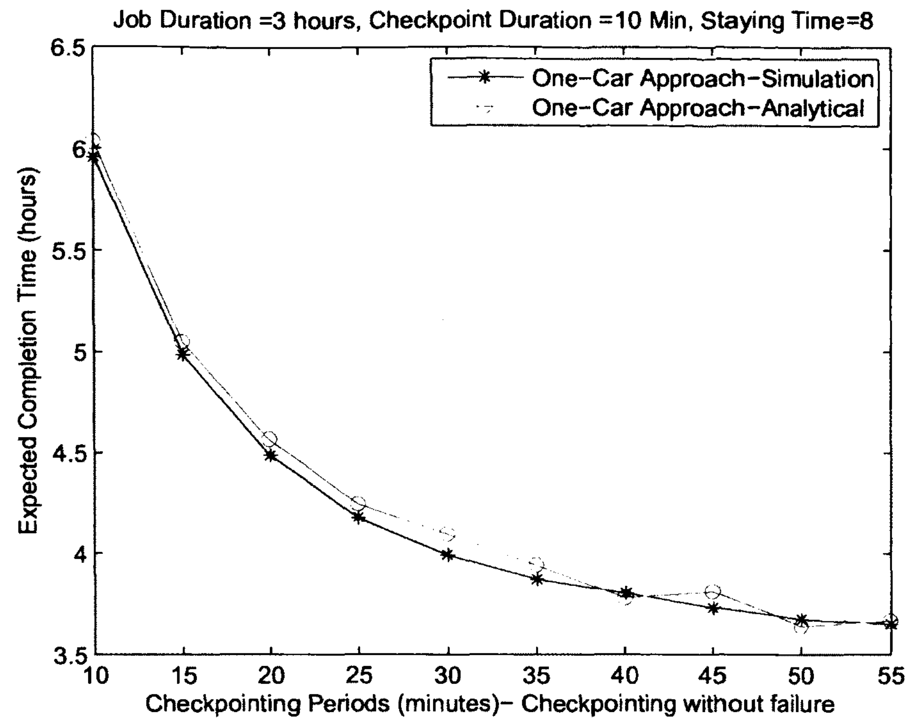


Fig. 40: Scenario 3. Illustrating Job Completion Time in J1 Strategy

hours and the average checkpoint duration is 20 minutes and average staying time of 5 hours.

- Scenario 7: In this scenario we assume that the average duration of the jobs are 3 hours and the average checkpoint duration is 20 minutes and average staying time of .5 hours.

- Scenario 8: In this scenario we assume that the average duration of the jobs are 3 hours and the average checkpoint duration is 10 minutes and average staying time of .5 hours.

- Scenario 9: In this scenario we assume that the average duration of the jobs are 7 hours and the average checkpoint duration is 10 minutes and average staying time of 3 hours.

Figure 41-49 are the comparison of J1 and J2 strategy. Comparison shows that J2 provides smaller job completion time.

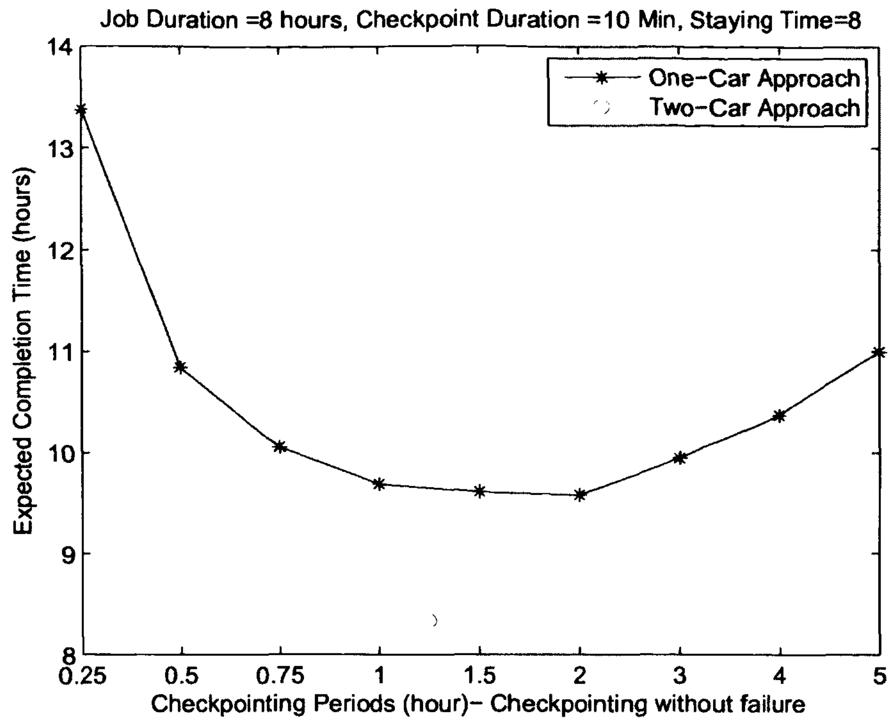


Fig. 41: Scenario 1. Illustrating Comparison between J1 and J2 Strategy.

6.3 COMPARING J1 AND J2 STRATEGIES

By analyzing the result from Figures 41-49 we can conclude the following. Job completion time in J2 strategy is always shorter than J1 strategy. The reason is in J1 strategy once the car leaves, if no checkpoint has been taken, the job is lost. In case there has been checkpoints the computation will roll back to the last checkpoint. In this case only part of the job is lost. The portion of the lost job depends on how often checkpoints are created. This will introduce a new challenge which is optimal period for checkpointing. In J2 strategy once the car leaves the other car will go under checkpointing and this will avoid computation loss.

By analyzing Figures 41-49, it is obvious that checkpointing period has important role in job completion time for strategy J1. Choosing non-optimal or sub-optimal checkpointing period will increase the job completion time drastically.

Results also show that when job duration is small compared with car residency time, J1 and J2 strategy will have same performance in terms of completion time. This is because most probably the job execution will be complete before any departure happens. In a

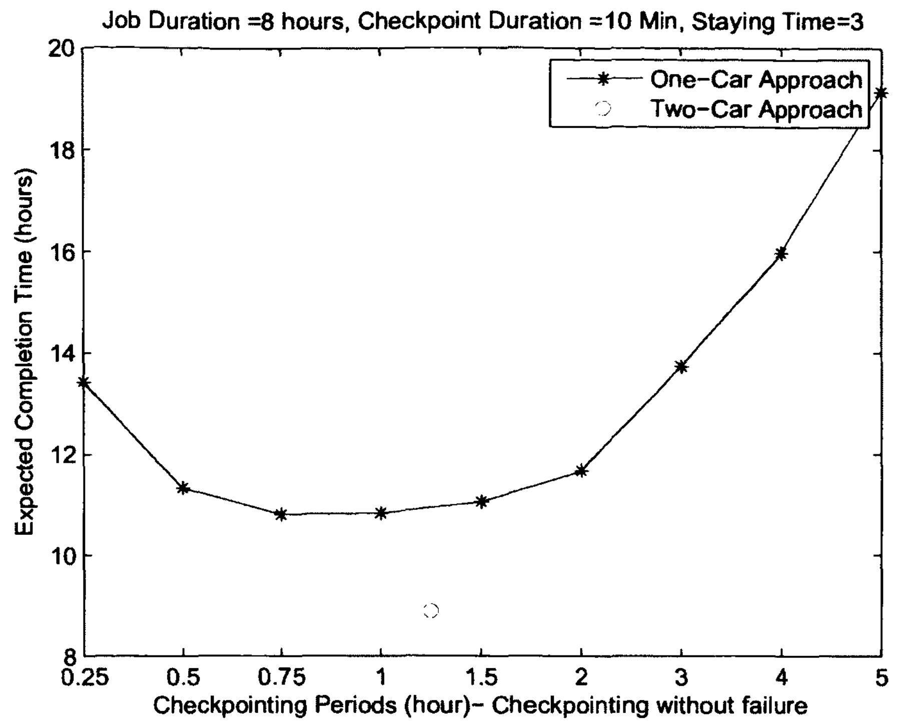


Fig. 42: Scenario 2. Illustrating Comparison between J1 and J2 Strategy.

scenario that no departure happens J1 and J2 will have some performance in terms of job completion time. Because there will be no checkpointing process.

Checkpoint duration also has direct effect in increasing job completion time. Figure 47 shows that by increasing the checkpoint duration J1 and J2 will have almost same performance because J2 by its nature creates more checkpoint compared with J1 and once the checkpoint duration is large the completion time will increase which brings J2 performance close to J1.

The other important factor in J2 performance is job duration. Figure 41 shows that by increasing job duration J2 performance increases compared with J1. Departure of car in J1 will cause computation loss. The computation loss will be longer for jobs with longer duration and this will decrease the J1 performance.

6.4 CONCLUDING REMARKS

In this chapter we have proposed a job assignment strategy using one car at a time in vehicular clouds and have analyzed its performance both analytically and by simulation.

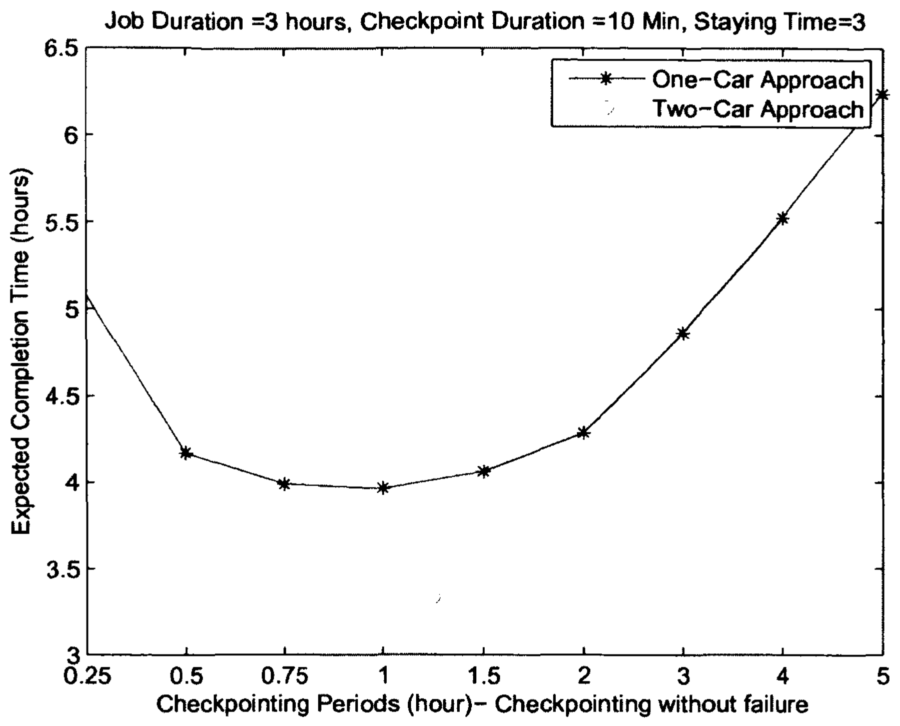


Fig. 43: Scenario 3. Illustrating Comparison between J1 and J2 Strategy.

We discussed advantages and disadvantages of J1 strategy compared with J2 strategy.

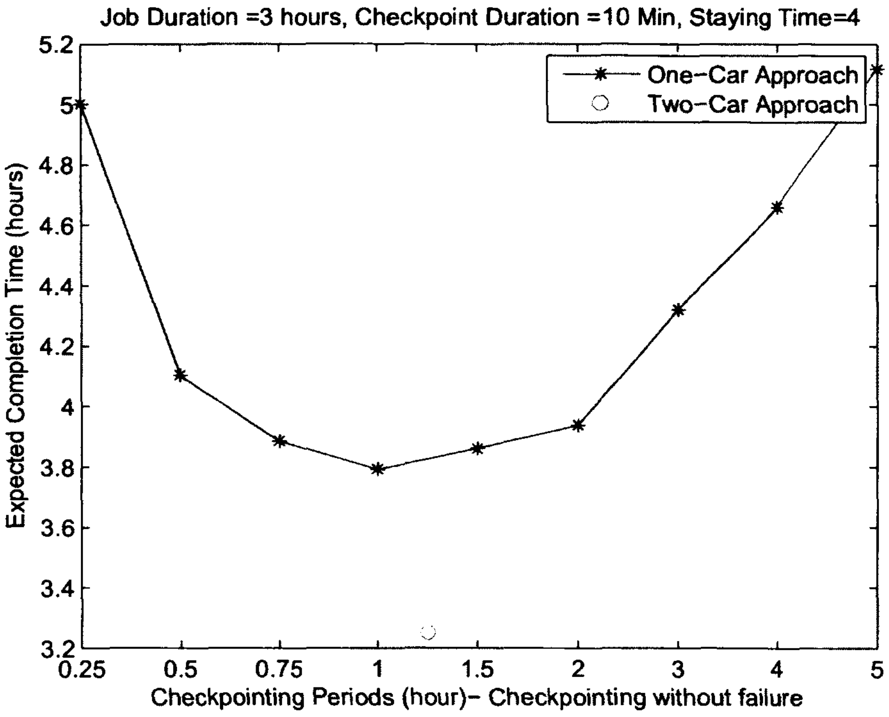


Fig. 44: Scenario 4. Illustrating Comparison between J1 and J2 Strategy.

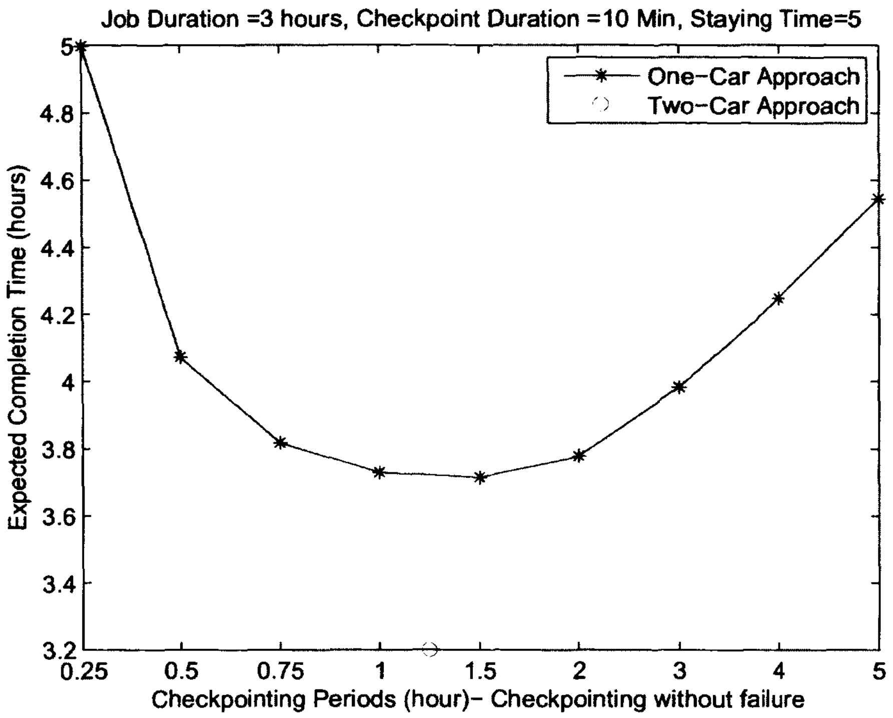


Fig. 45: Scenario 5. Illustrating Comparison between J1 and J2 Strategy.

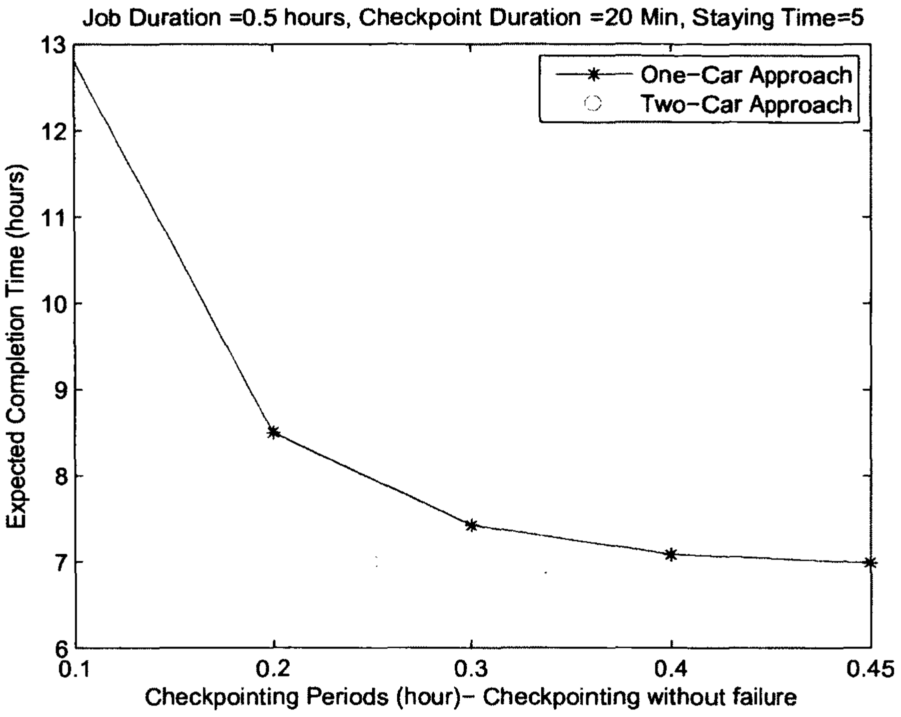


Fig. 46: Scenario 6. Illustrating Comparison between J1 and J2 Strategy.

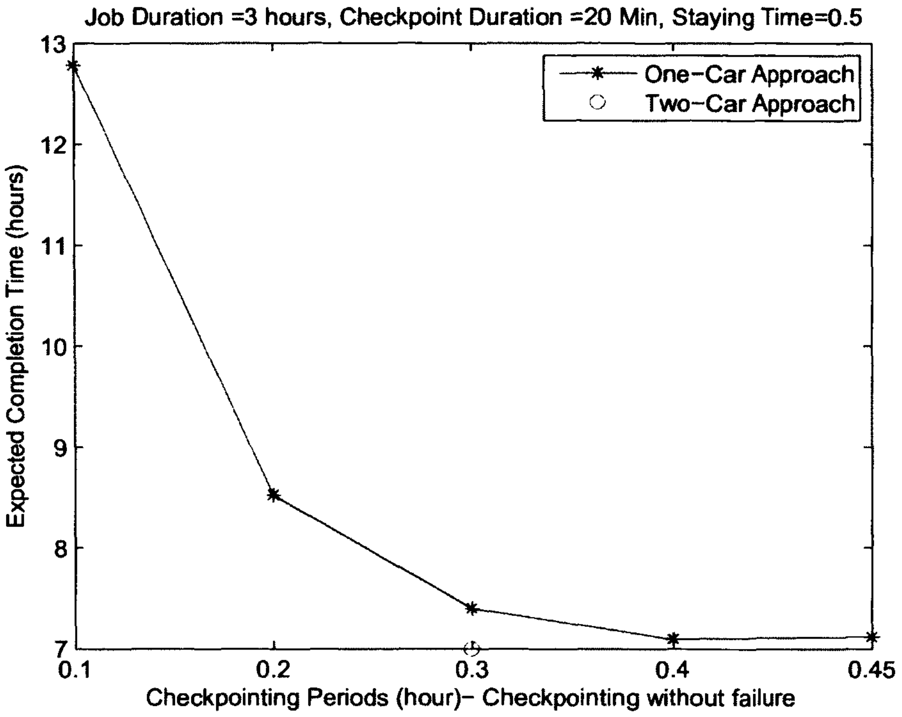


Fig. 47: Scenario 7. Illustrating Comparison between J1 and J2 Strategy.

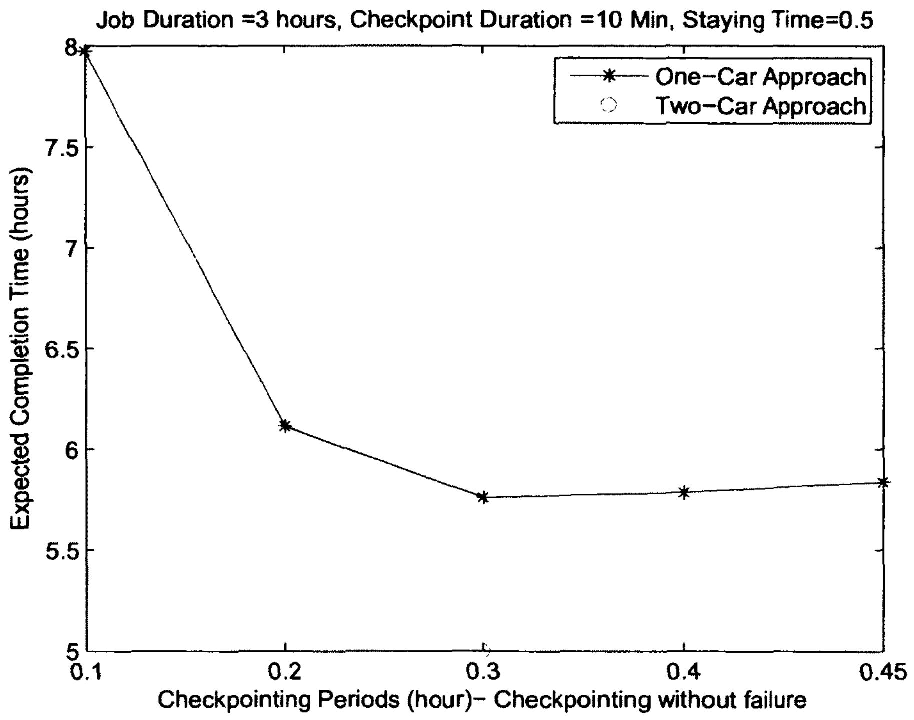


Fig. 48: Scenario 8. Illustrating Comparison between J1 and J2 Strategy.

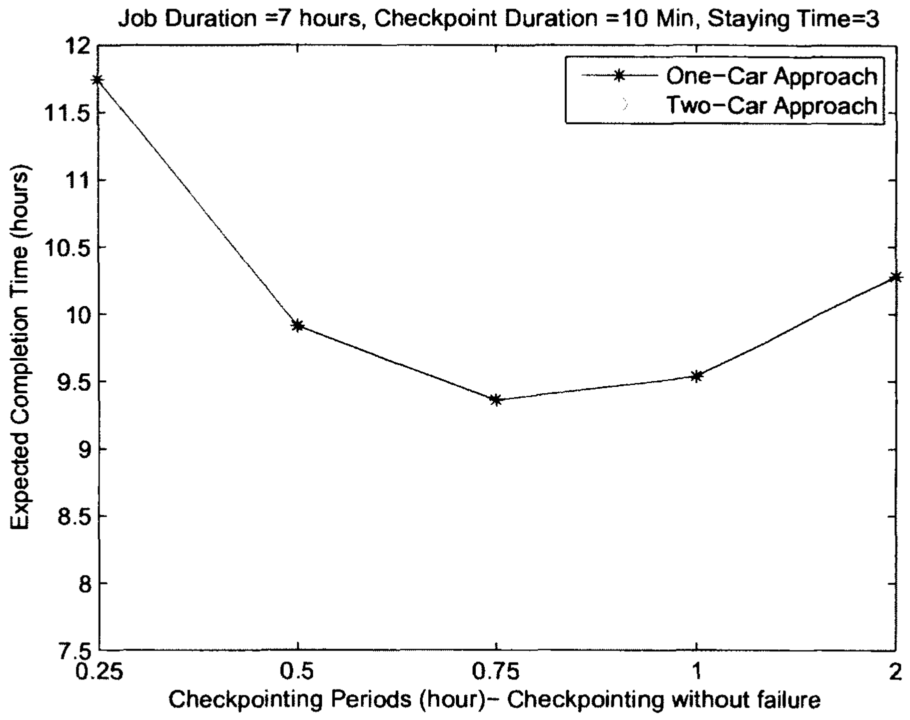


Fig. 49: Scenario 9. Illustrating Comparison between J1 and J2 Strategy.

CHAPTER 7

COMPARISON OF JOB ASSIGNMENT STRATEGIES

In this chapter we compare J2 and J3 strategies in terms of job completion time, by assuming that car does not leave during checkpointing. In Section 7.1 first we offer theoretical predictions of the job execution time in J3 strategy. Then we provide a comprehensive set of simulations that shows our theoretical predictions are accurate.

Checkpoint duration has important role in job completion time and system failure. In Section 7.4 we predict the probability of checkpoint failure by experimenting different scenarios. A comprehensive set of simulations that shows our theoretical predictions are accurate. In Section 7.5 we present number of checkpoints during job execution time. We provide simulation results for this section.

7.1 JOB EXECUTION TIME J3

Lemma 7.1.1. *Let X, Y and W be independent, exponentially distributed random variables with parameter λ. The random variable $Z = \min\{X,Y,W\}$ is exponentially distributed with parameter 3λ.*

Proof. For $z \geq 0$ we write

$$\begin{aligned}
\Pr[Z > z] &= \Pr[\min\{X,Y,W\} > z] \\
&= \Pr[\{Y > z\} \cap \{Y > z\} \cap \{W > z\}] \\
&= \Pr[X > z] \Pr[Y > z] \Pr[W > z] \\
&= e^{-\lambda z} e^{-\lambda z} e^{-\lambda z} = e^{-3\lambda z}
\end{aligned}$$

and the proof of the lemma is complete. □

Lemma 7.1.1 tells us that the inter-arrival time of checkpoints is exponentially distributed with parameter 3λ and so the the conditional distribution of the random variable N, given that $\{T = t\}$, is Poisson with parameter $3\lambda t$. In other words, for all natural number n

$$\Pr[N = n \mid T = t] = \frac{(3\lambda t)^n}{n!} e^{-3\lambda t}.$$

To get a handle on the probability mass function of the number N of checkpoints, we evaluate $\Pr[N=n]$ by conditioning on T:

$$\begin{aligned}\Pr[N=n] &= \int_{t=0}^{\infty}\Pr[N=n \mid T=t]\,\mathrm{d}G(t)\\ &= \int_0^{\infty}\frac{(3\lambda t)^n}{n!}e^{-3\lambda t}\,\mathrm{d}G(t).\end{aligned} \tag{51}$$

We are now in a position to reveal the relationship between N and T. We proceed as follows:

$$\begin{aligned}E[N] &= \sum_{n=0}^{\infty} n\Pr[N=n]\\ &= \sum_{n=0}^{\infty} n\int_0^{\infty}\frac{(3\lambda t)^n}{n!}e^{-3\lambda t}\,\mathrm{d}G(t)\\ &= \int_0^{\infty}\sum_{n=0}^{\infty} n\frac{(3\lambda t)^n}{n!}e^{-3\lambda t}\,\mathrm{d}G(t)\\ &= 3\lambda\int_0^{\infty} te^{-3\lambda t}\,\mathrm{d}G(t)\sum_{n-1}^{\infty}\frac{(3\lambda t)^{n-1}}{(n-1)!}\\ &= 3\lambda\int_0^{\infty} te^{-3\lambda t}e^{3\lambda t}\,\mathrm{d}G(t)\\ &= 3\lambda\int_0^{\infty} t\,\mathrm{d}G(t)\\ &= 3\lambda E[T].\end{aligned} \tag{52}$$

Next, we are interested in the expected total time spent taking checkpoints (and, of course, restarting the job after checkpoints). For this purpose, we need to evaluate the expectation $E[X_1+X_2+\cdots+X_N]$. By the law of total expectation we have

$$\begin{aligned}&E[X_1+X_2+\cdots+X_N]\\ =\ &\sum_{n=0}^{\infty}E[X_1+X_2+\cdots+X_N \mid N=n]\Pr[N=n]\\ =\ &E[X_1+X_2+\cdots+X_n]\Pr[N=n]\\ =\ &\sum_{n=0}^{\infty} nE[X]\Pr[N=n]\\ =\ &E[X]\sum_{n=0}^{\infty} n\Pr[N=n]\\ =\ &E[X]E[N]\end{aligned} \tag{53}$$

where, recall, $E[X]$ is the expected duration of a checkpoint/restart operation.

Finally, we can write

$$\begin{aligned} E[J] &= E[T+X_1+X_2+\cdots+X_N] \\ &= E[T]+E[X_1+X_2+\cdots+X_N] \\ &= E[T]+E[X]E[N] \quad \text{[by (53)]} \\ &= E[T]+3\lambda E[X]E[T] \quad \text{[by (52)]} \\ &= E[T](1+3\lambda E[X]). \end{aligned} \tag{54}$$

Observe that the distributions of T and X only occur in (54) through their first moment. This tells us that our result holds for a large number of distributions that share the same first moment.

7.1.1 SIMULATION RESULTS

In addition to the theoretical models developed in Section 7.1 we have evaluated the performance of our fault-tolerant job assignment strategy through an extensive set of simulations. The goal of this section is to report on our experimental results and to show that they closely match the theoretical predictions obtained previously. Specifically, Subsection 7.1.2 presents the details of our simulation model used to generate the various diagrams. Later, Subsection 7.1.3 summarizes our findings in the form of various diagrams.

7.1.2 SIMULATION MODEL

In this subsection, we present the detail of our simulation model. We assume a large parking lot, similar to a typical parking lot in the downtown area of a large city. Car residency time are assumed to be exponentially distributed with parameter λ. In our simulation λ varied between $\frac{1}{3}$ and $\frac{1}{8}$. Since the expected residency time is given by $\frac{1}{\lambda}$, our choice corresponds to cars spending, on the average, between three and eight hours in the parking lot. Also, we assumed that the number of available cars sufficiently large, so that upon one car departing, a substitute could be found without delay.

Jobs durations are exponentially distributed random variables with parameter γ. In our simulation γ varied between $\frac{1}{3}$ and $\frac{1}{5}$.

Upon one of the car assigned to a job departing before job termination, the remaining car enters a checkpoint. Individual checkpoint duration is a function of the amount of data that needs to be stored and is taken to be exponentially distributed with parameter μ. This means that the average duration of a checkpoint is $\frac{1}{\mu}$. We experimented scenarios, namely average checkpoint durations of 10, 15 and 20 minutes.

This model was developed in MATLAB 7.14. In each of the experiments, a job with one of the parameters described above was submitted to the Vehicular Cloud and the duration of job executed recorded. The same experiment was then repeated 10^5 times and the average of these runs were plotted.

7.1.3 SIMULATION RESULTS

This section presents our simulation results performed in order to validate the analytical results obtained in Sections 7.1. We have performed several sets of experiments that we now describe. In order to evaluate the impact of the various problem parameters discussed in Subsection 7.1.2 on the expected duration of job execution we have conducted a number experiments corresponding to the scenarios that we detail next.

In these set of experiments, the model has been set up in such a way that system failures due to cars leaving during checkpointing will not occur.

- Scenario 1: In the first scenario we assumed that under ideal conditions (i.e. no checkpoints necessary), the average job duration is three hours. We further assumed an average checkpoint duration of 10 minutes;
- Scenario 2: In this scenario we assume that the average duration of the jobs are 3 hours and the average checkpoint duration is 15 minutes.
- Scenario 3: In this scenario we assume that the average duration of the jobs are 3 hours and the average checkpoint duration is 20 minutes.
- Scenario 4: In order to see the impact of longer job duration, in the second scenario we assumed an average job duration of five hours and an average checkpoint duration of 10 minutes.
- Scenario 5: In this scenario we assume that the average duration of the jobs are 5 hours and the average checkpoint duration is 15 minutes.
- Scenario 6: In this scenario we assume that the average duration of the jobs are 5 hours and the average checkpoint duration is 20 minutes.

- Scenario 7: In order to see the impact of longer job duration, in the second scenario we assumed an average job duration of eight hours and an average checkpoint duration of 10 minutes.
- Scenario 8: In this scenario we assume that the average duration of the jobs are 8 hours and the average checkpoint duration is 15 minutes.
- Scenario 9: In this scenario we assume that the average duration of the jobs are 8 hours and the average checkpoint duration is 20 minutes.

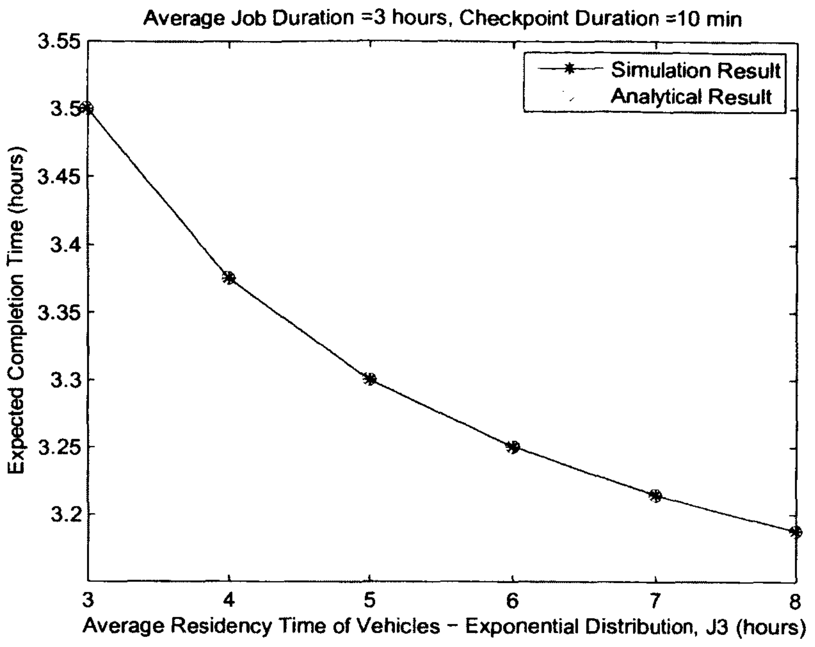


Fig. 50: Scenario 1. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed

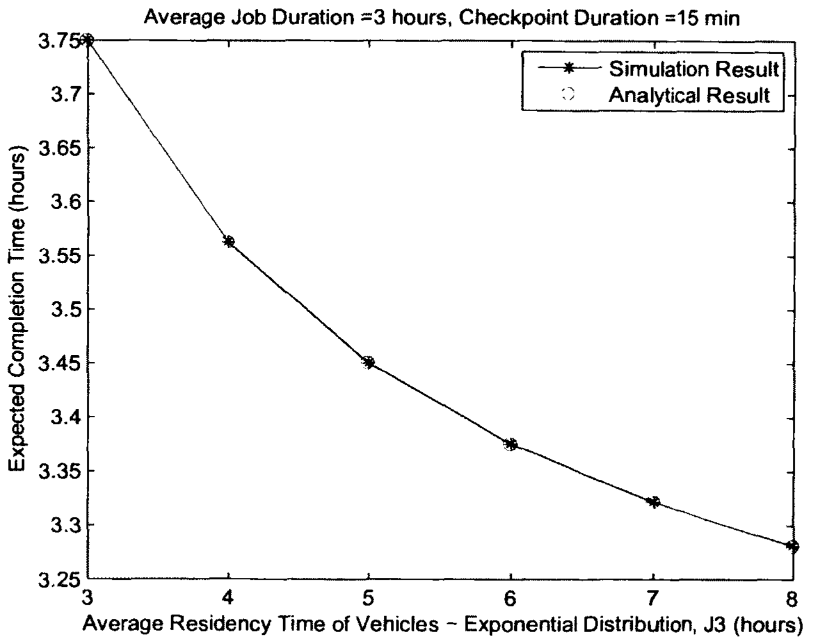


Fig. 51: Scenario 2. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed

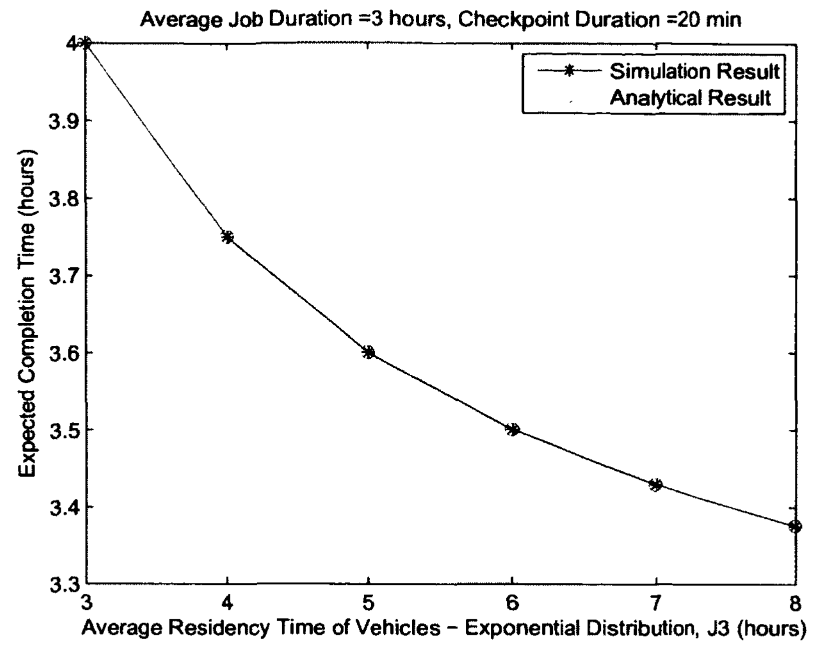


Fig. 52: Scenario 3. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed

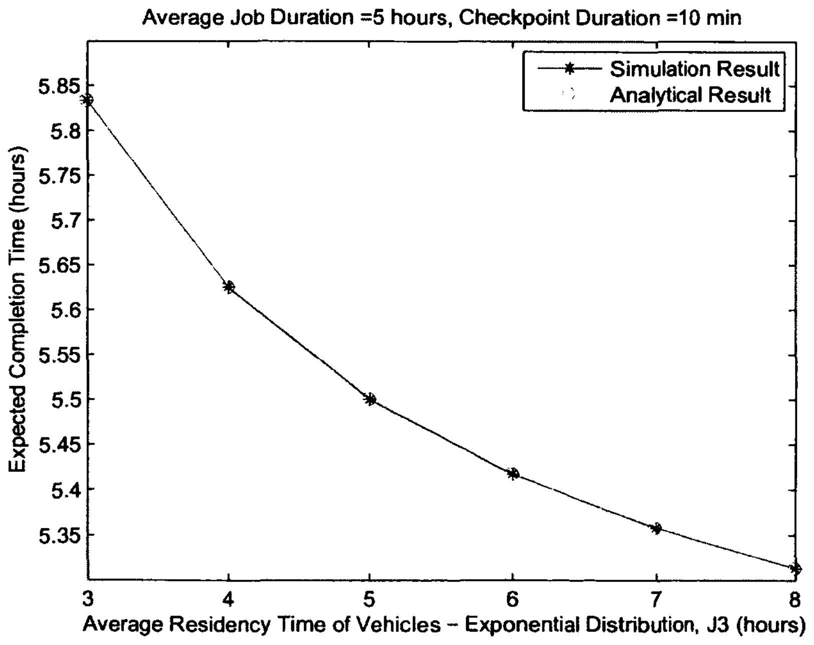


Fig. 53: Scenario 4. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed

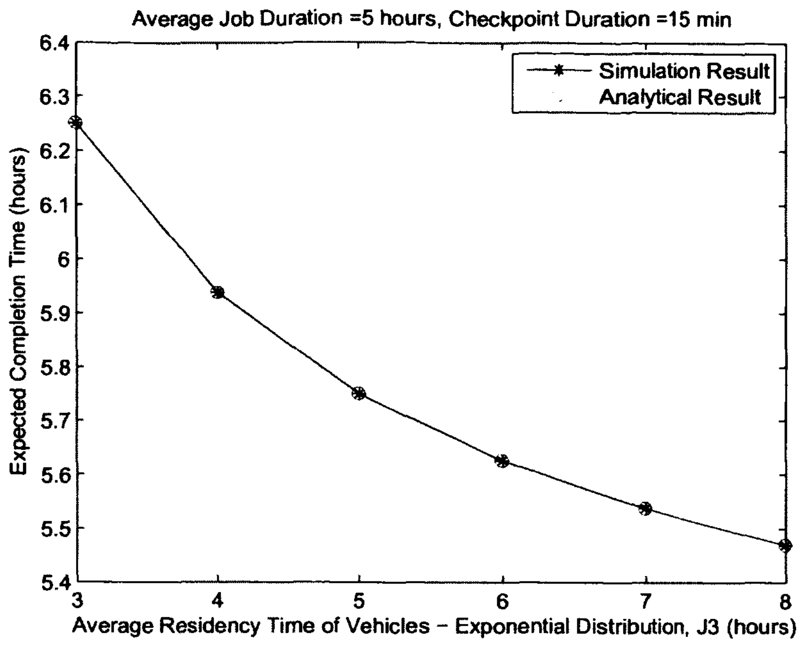


Fig. 54: Scenario 5. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed

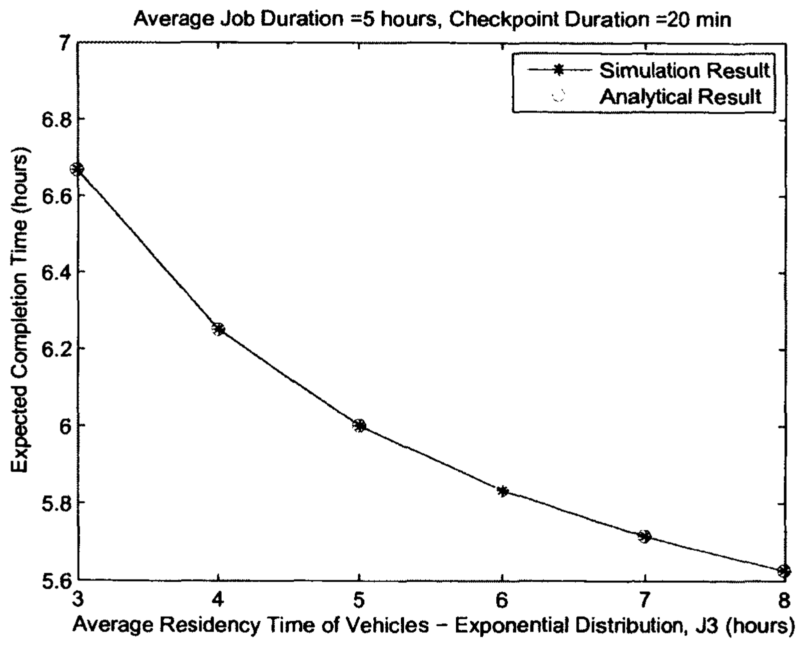


Fig. 55: Scenario 6. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed

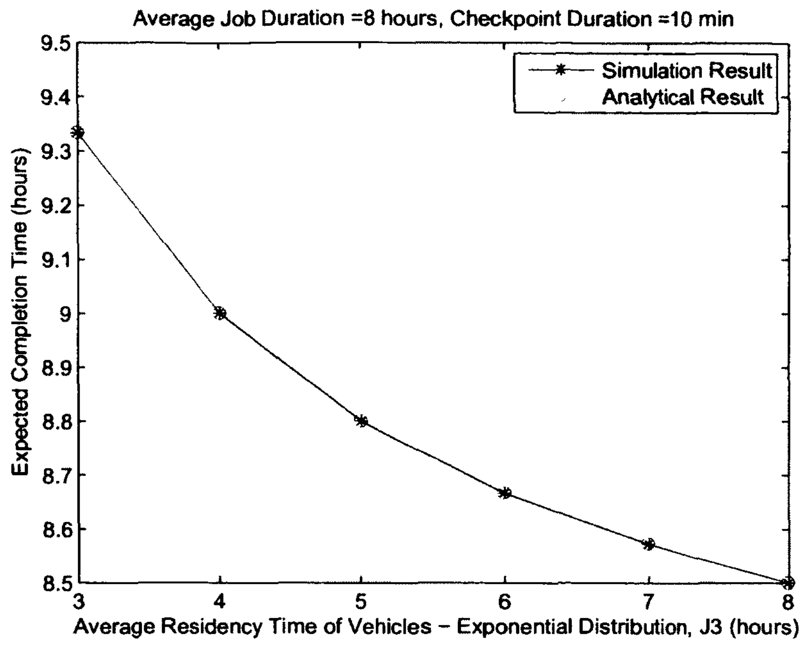


Fig. 56: Scenario 7. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed

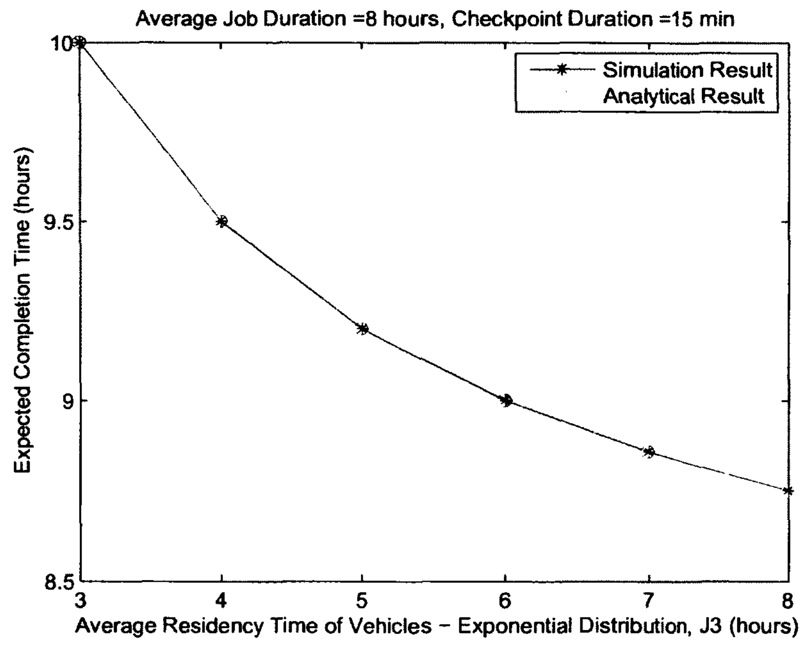


Fig. 57: Scenario 8. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed

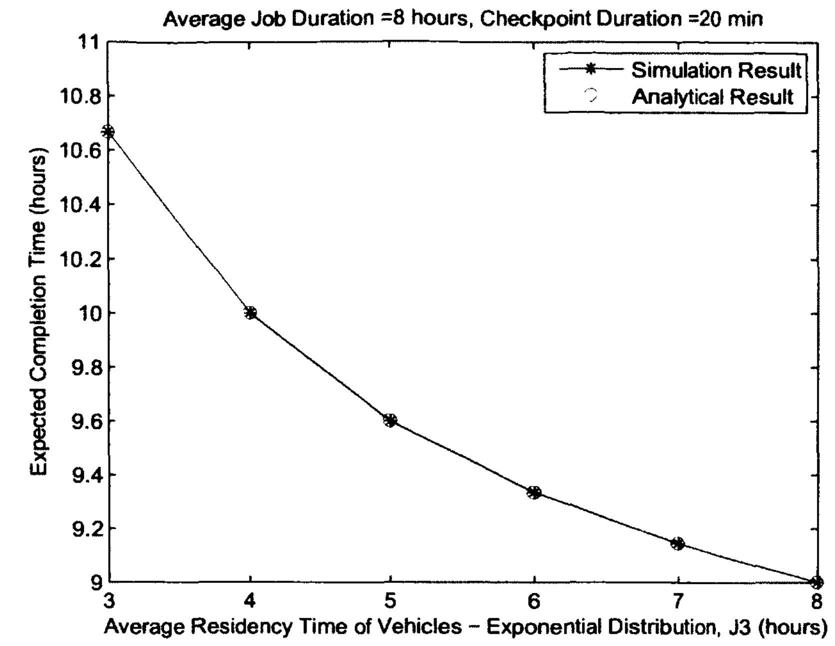


Fig. 58: Scenario 9. Illustrating Job Completion Time J3 Strategy Checkpoint Failure not Allowed

The results of these scenarios are featured in Figures 50-58. In all scenarios described, the simulation results confirmed the accuracy of the theoretical predictions, the results being, essentially, indistinguishable.

7.2 COMPARISON OF JOB COMPLETION TIME

In this section we compare *J*2 and *J*3 strategy in terms of job completion time.

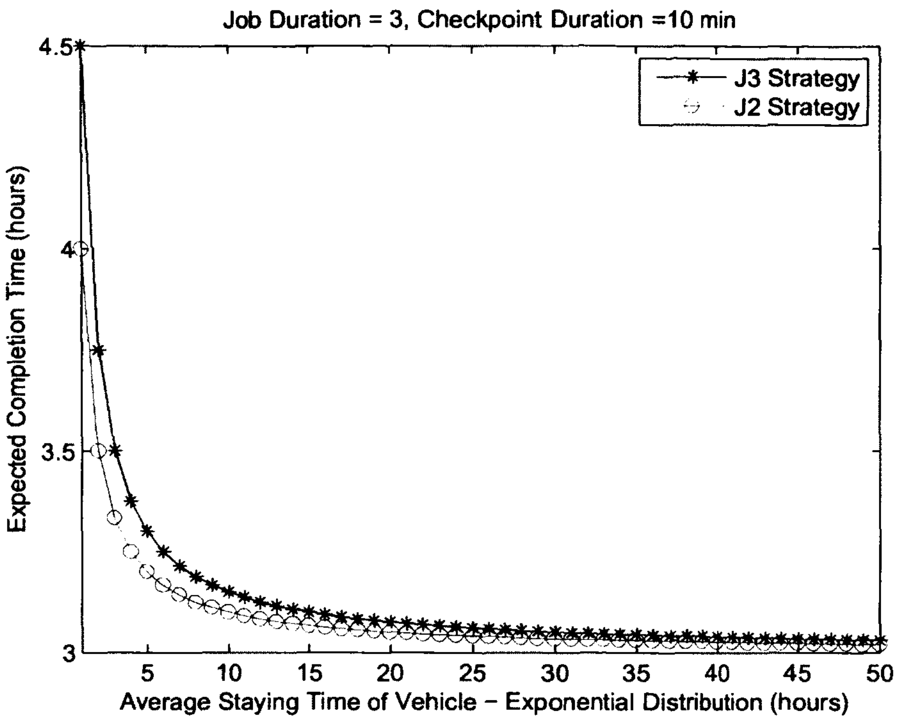


Fig. 59: Scenario 1. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy.

Results show that completion time in J2 strategy is smaller than J3 strategy approach. Having 3 cars will increase the probability of departure and each departure will lead to making checkpoint. Having more checkpoints in J3 strategy is the main reason that increases the completion time of jobs.

In the other hand by analysing the results from Section 5.8.2 it is clear that mean time to failure in J3 strategy approach is longer that J2 strategy approach.

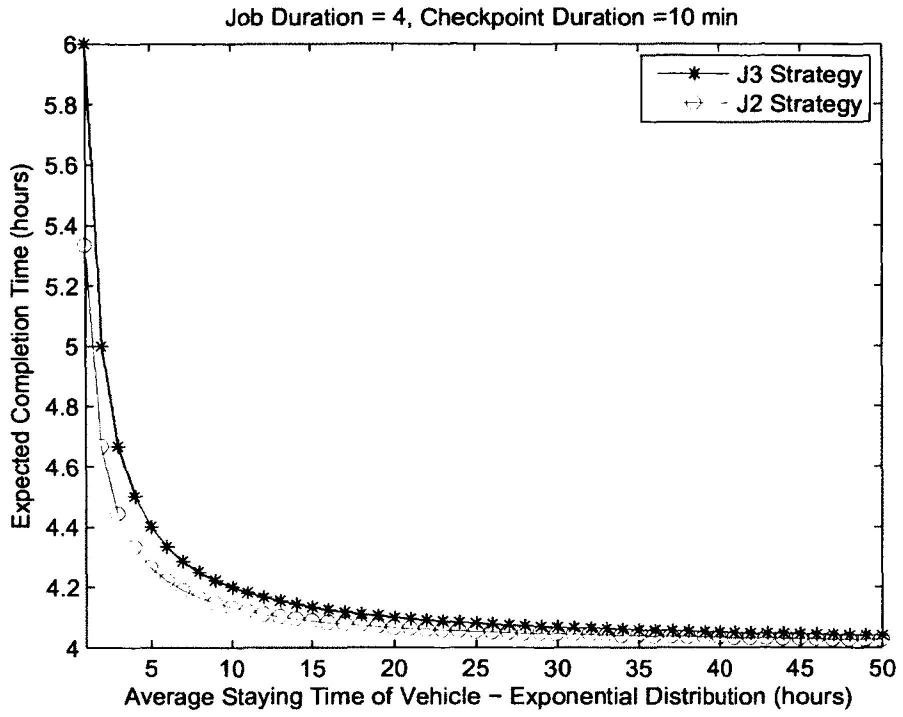


Fig. 60: Scenario 2. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy.

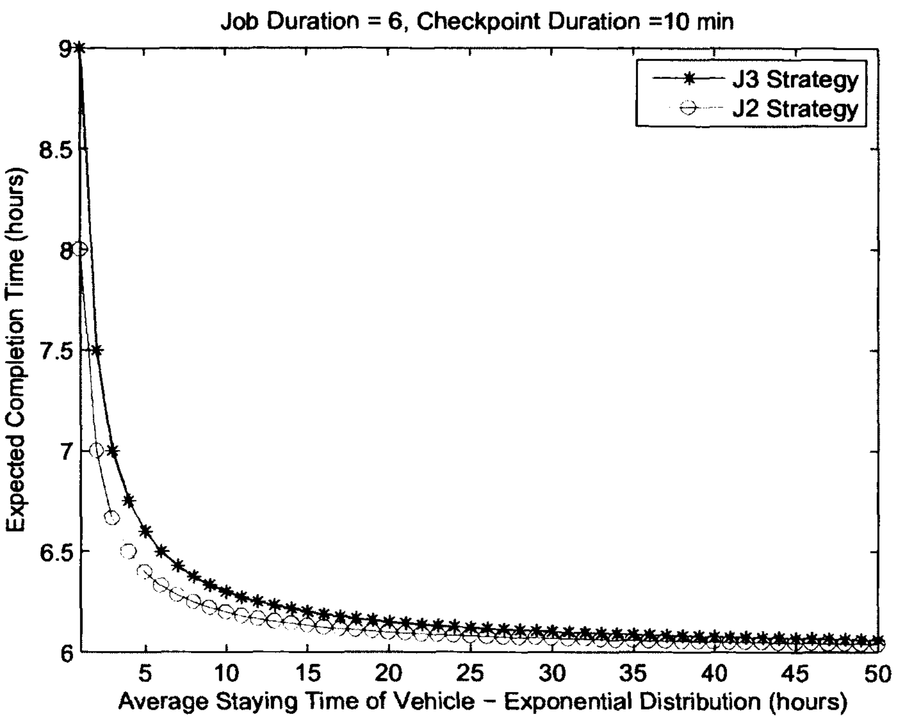


Fig. 61: Scenario 3. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy.

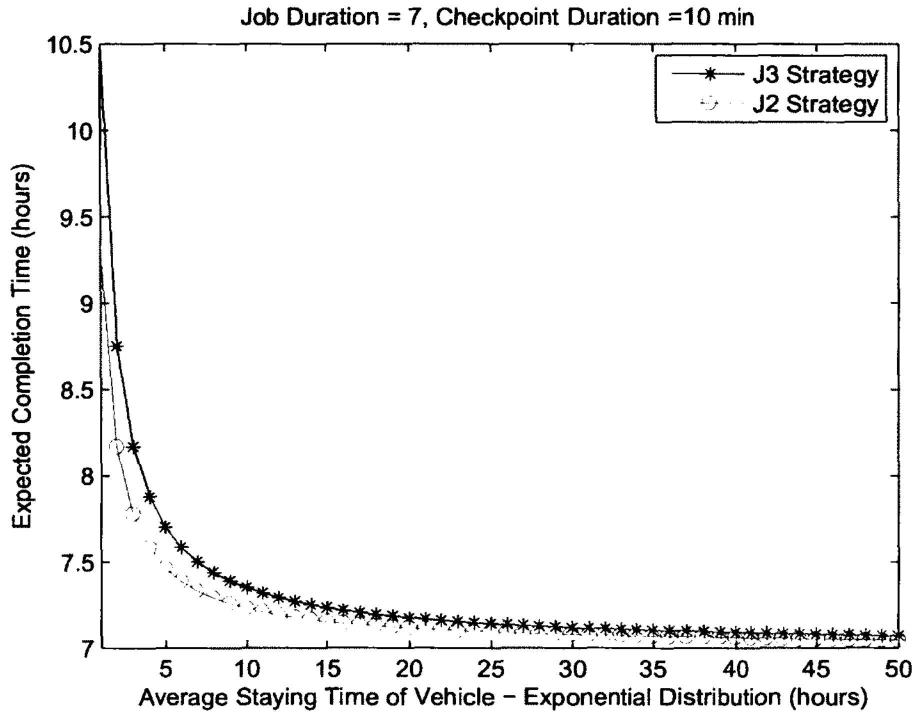


Fig. 62: Scenario 4. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy.

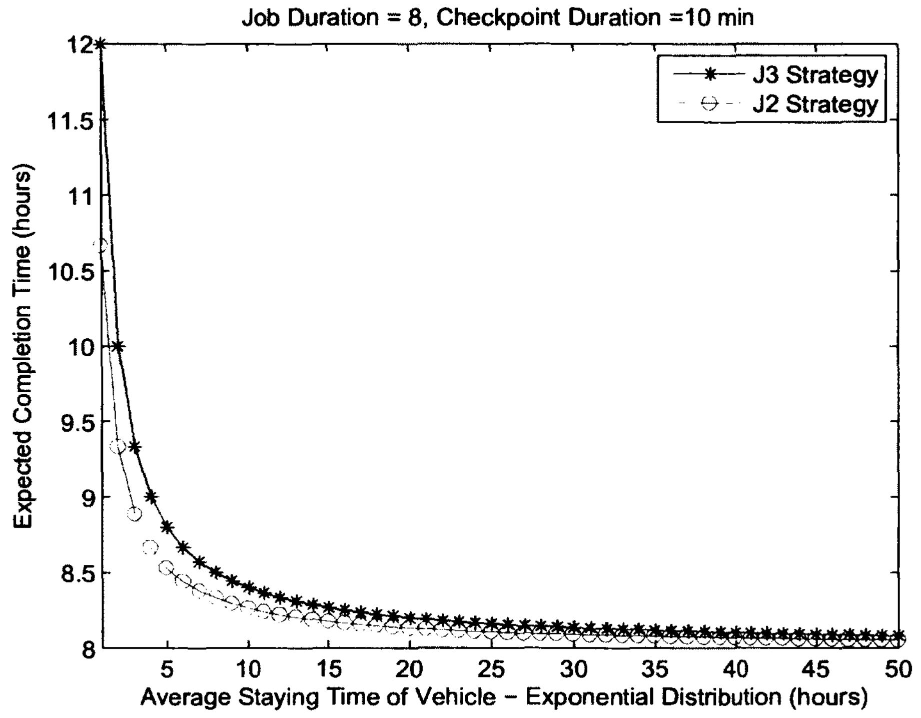


Fig. 63: Scenario 5. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy.

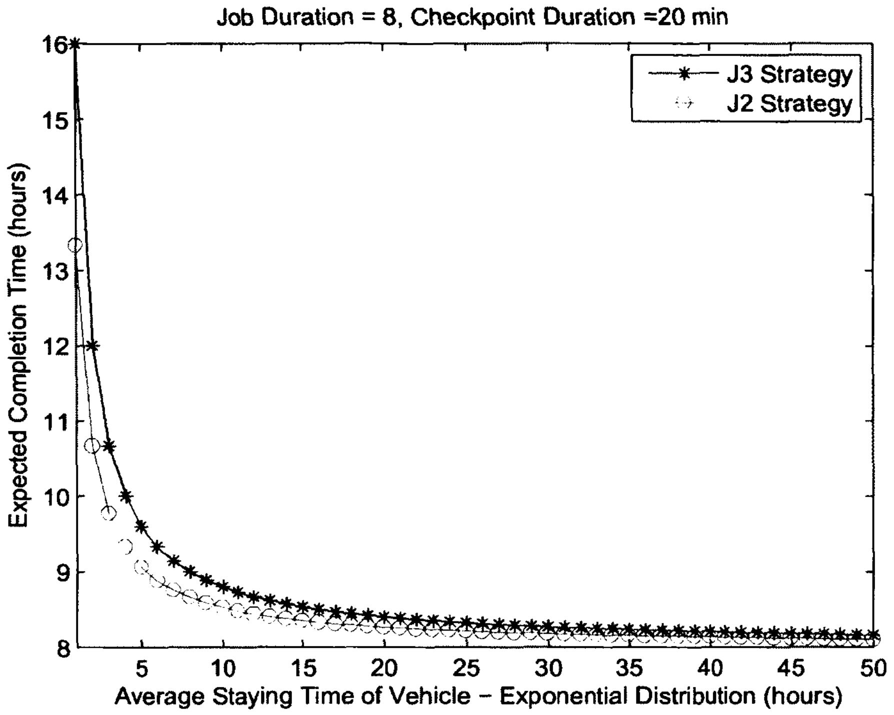


Fig. 64: Scenario 6. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy.

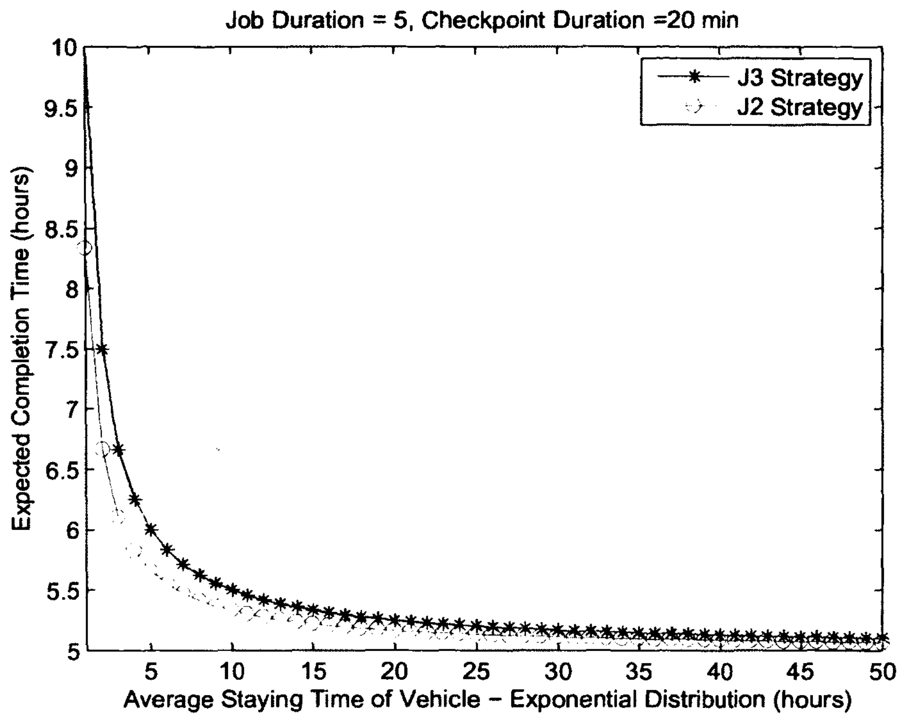


Fig. 65: Scenario 7. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy.

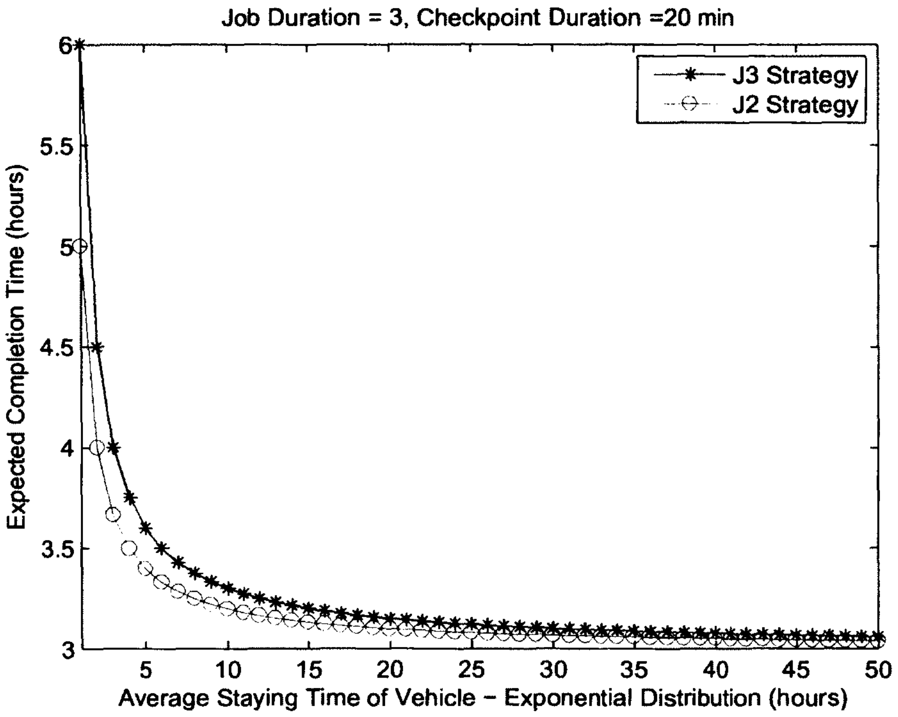


Fig. 66: Scenario 8. Illustrating the Comparison of Job Completion Time in J2 and J3 Strategy.

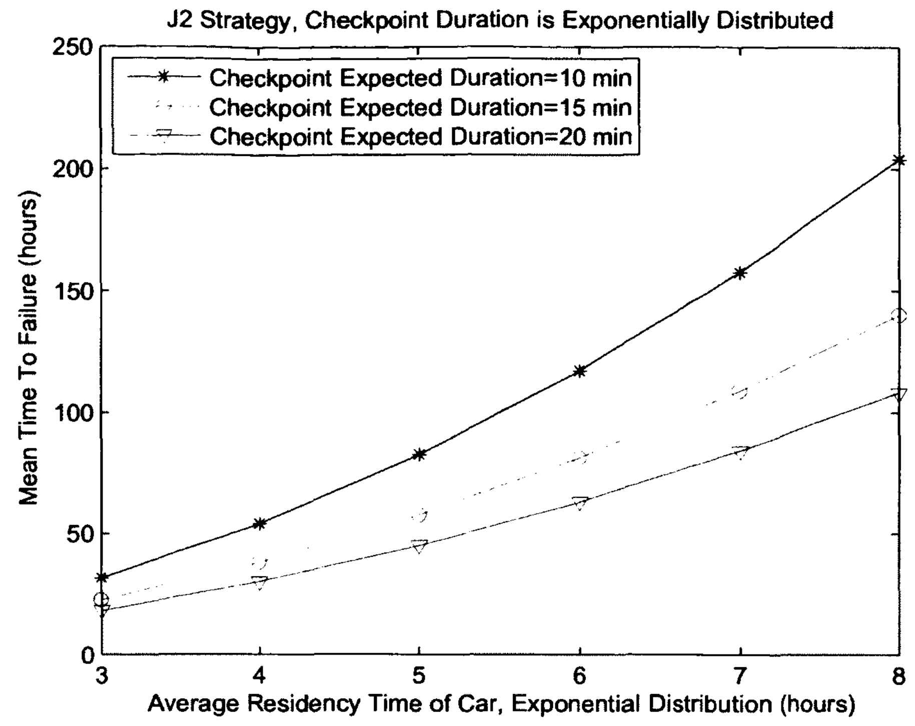


Fig. 67: Scenario 1. Illustrating the Comparison of Mean Time to Failure in J2 Strategy.

7.3 COMPARISON OF MEAN TIME TO FAILURE

In this section we analyze the impact of checkpoint duration in mean time to failure for J2 and J3 strategies. First we analyze the results for J2 strategy in Figure 67, then we explain the results in Figure 68 for J3 strategy.

Figure 67 shows the mean time to failure for J2 strategy. In this figure the residency time of vehicles varies between 3 and 8 hours and checkpoint duration can have following values: 10, 15 and 20 minutes. Results from Figure 67 show that by increasing the checkpoint duration, mean time to failure in J2 strategy decreases. The reason is because by increasing checkpoint duration, the probability that car leaves during checkpointing increases. Since departure of the car during checkpointing leads to failure, this will lead to shorter mean time to failure.

Figure 68 shows the mean time to failure for J3 strategy. In this figure the residency time of vehicles varies between 3 and 8 hours and checkpoint duration can have following values: 10, 15 and 20 minutes. Analyzing the results from Figure 68 shows that J3 has same behavior as J2. In J3 same as J2, by increasing the checkpoint duration, meant time to

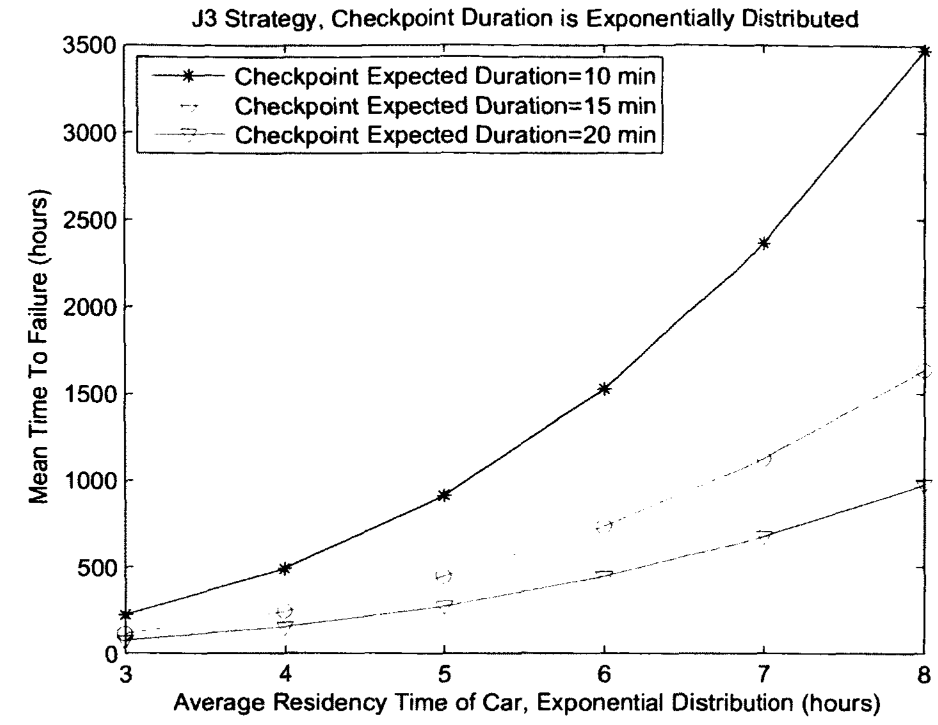


Fig. 68: Scenario 2. Illustrating the Comparison of Mean Time to Failure in J3 Strategy.

failure decreases. By increasing the checkpoint duration the probability that two remaining cars depart will increases which leads to shorter mean time to failure.

7.4 PROBABILITY OF CHECKPOINT FAILURE

In this section we present the analytical and simulation result for the probability of checkpoint failure in J2 strategy.

The goal of this section is to provide an outline for checkpointing process presented in J2 strategy. Having general idea about number of checkpoints and the probability of checkponiting success is important in determining system requirement and reliability. In other words based on checkpoint failure rate the administrator can consider backup strategy like using traditional servers in vehicular cloud.

Also determining the expected number of checkpoints can be used in monitoring the system. As an instance administrator can monitor the job execution process and compare the number of taken checkpoints with the number of expected checkpoints. The comparison can determine if there has been any malicious action and the reliability of the system can be defined.

From Lemma 4.6.1 we know that if car residency times are iid exponential random variables with parameter λ and checkpoint duration is a random variable with distribution function F and finite expectation μ, P the probability that checkpoint succeed is:

$$P = \int_{x=0}^{\infty} e^{-\lambda x} \mathrm{d}F(x) \tag{55}$$

7.4.1 SIMULATION RESULTS

This section presents our simulation results performed in order to validate the analytical results obtained in Sections 7.4. We have performed several sets of experiments that we now describe. In order to evaluate the impact of the various problem parameters on the probability of checkpoint failure we have conducted a number experiments corresponding to the scenarios that we detail next.

- Scenario 1: In the first scenario we assumed that the average checkpoint duration is 10 minutes;
- Scenario 2: In this scenario we assumed that the average checkpoint duration is 15 minutes;
- Scenario 3: In this scenario we assumed that the average checkpoint duration is 20 minutes;

The results of these scenarios are featured in Figures 69-71. In all scenarios described, the simulation results confirmed the accuracy of the theoretical predictions, the results being, essentially, indistinguishable.

7.5 NUMBER OF CHECKPOINTS

In this section we present the simulation results for the number of the checkpoints that are taken during executing a job in J2 strategy. We assume that car may leave during checkpointing that leads to failure. We provide the results for total checkpoints, successful checkpoints and failed checkpoints.

In order to evaluate the impact of the various problem parameters on the probability of checkpoint failure we have conducted a number experiments corresponding to the scenarios that we detail next.

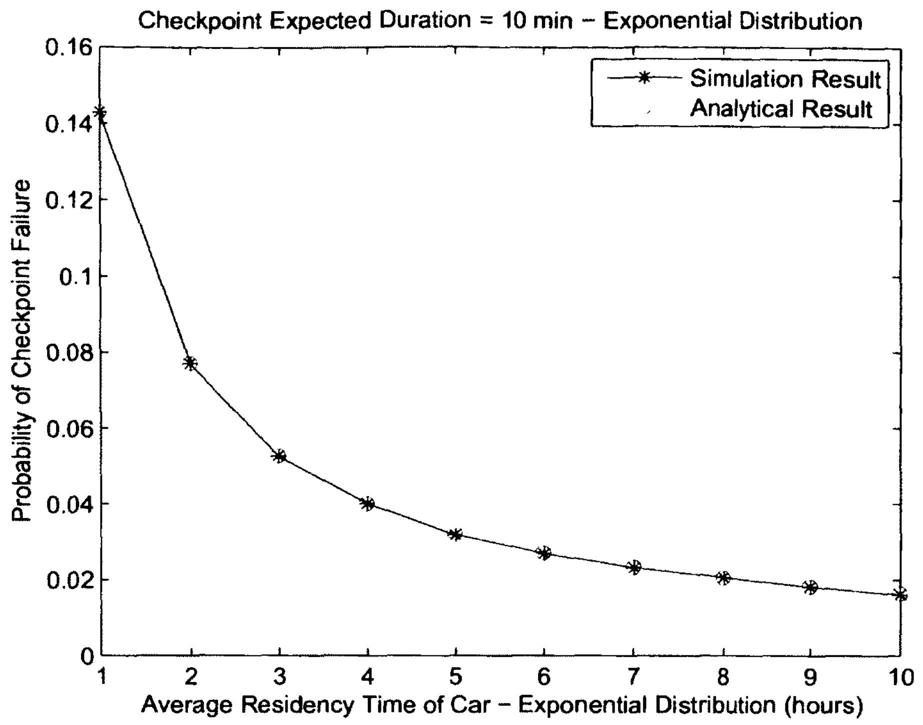


Fig. 69: Scenario 1. Illustrating the Probability of Checkpoint Failure.

- Scenario 1: In the first scenario we assumed that the average job duration is three hours. We further assumed an average checkpoint duration of 10 minutes;
- Scenario 2: In this scenario we assume that the average duration of the jobs are 3 hours and the average checkpoint duration is 15 minutes.
- Scenario 3: In this scenario we assume that the average duration of the jobs are 3 hours and the average checkpoint duration is 20 minutes.
- Scenario 4: In order to see the impact of longer job duration, in the second scenario we assumed an average job duration of five hours and an average checkpoint duration of 10 minutes.
- Scenario 5: In this scenario we assume that the average duration of the jobs are 5 hours and the average checkpoint duration is 15 minutes.
- Scenario 6: In this scenario we assume that the average duration of the jobs are 5 hours and the average checkpoint duration is 20 minutes.

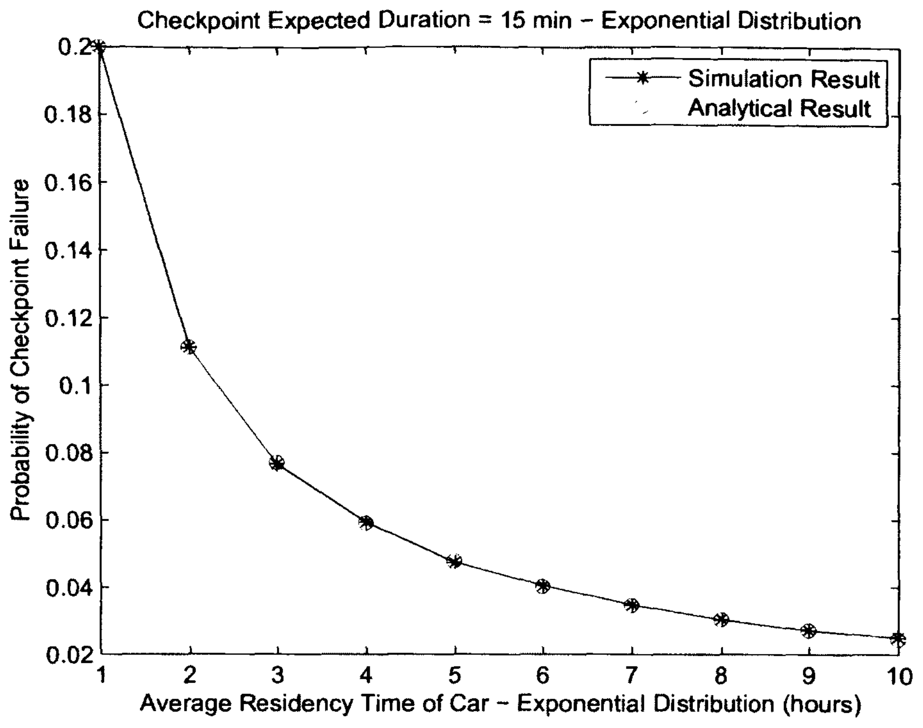


Fig. 70: Scenario 2. Illustrating the Probability of Checkpoint Failure.

- Scenario 7: In order to see the impact of longer job duration, in the second scenario we assumed an average job duration of eight hours and an average checkpoint duration of 10 minutes.
- Scenario 8: In this scenario we assume that the average duration of the jobs are 8 hours and the average checkpoint duration is 15 minutes.
- Scenario 9: In this scenario we assume that the average duration of the jobs are 8 hours and the average checkpoint duration is 20 minutes.

The results of these scenarios are featured in Figures 72-80.

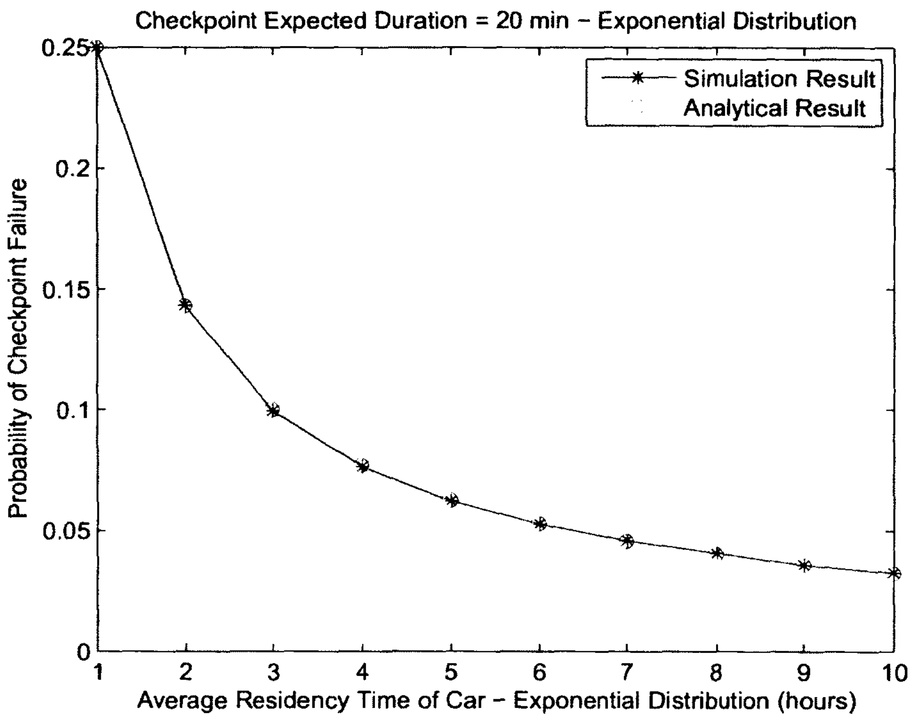


Fig. 71: Scenario 3. Illustrating the Probability of Checkpoint Failure.

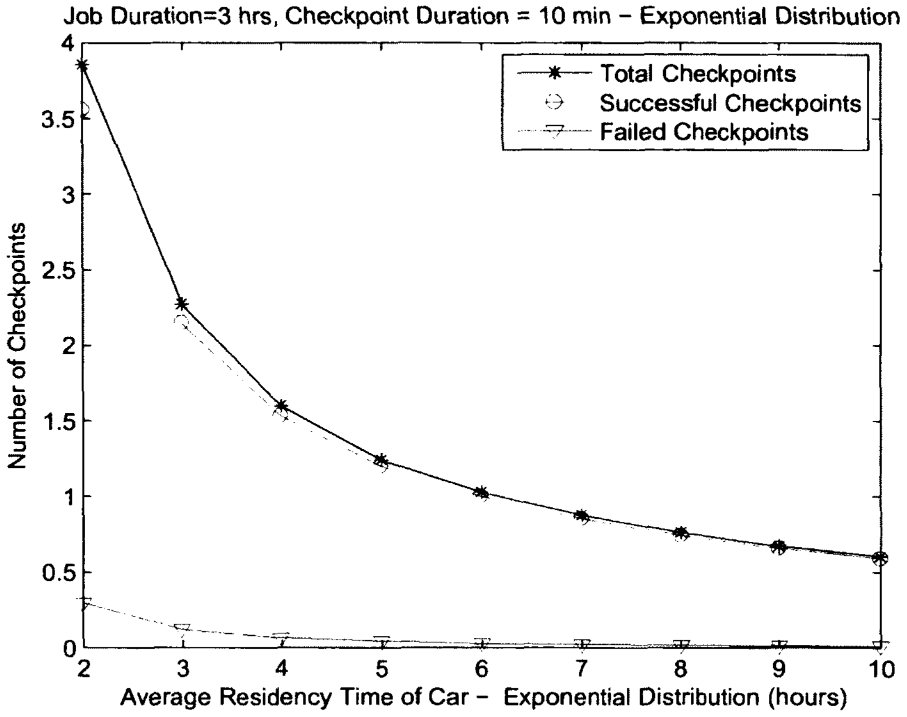


Fig. 72: Scenario 1. Illustrating the Number of checkpoints.

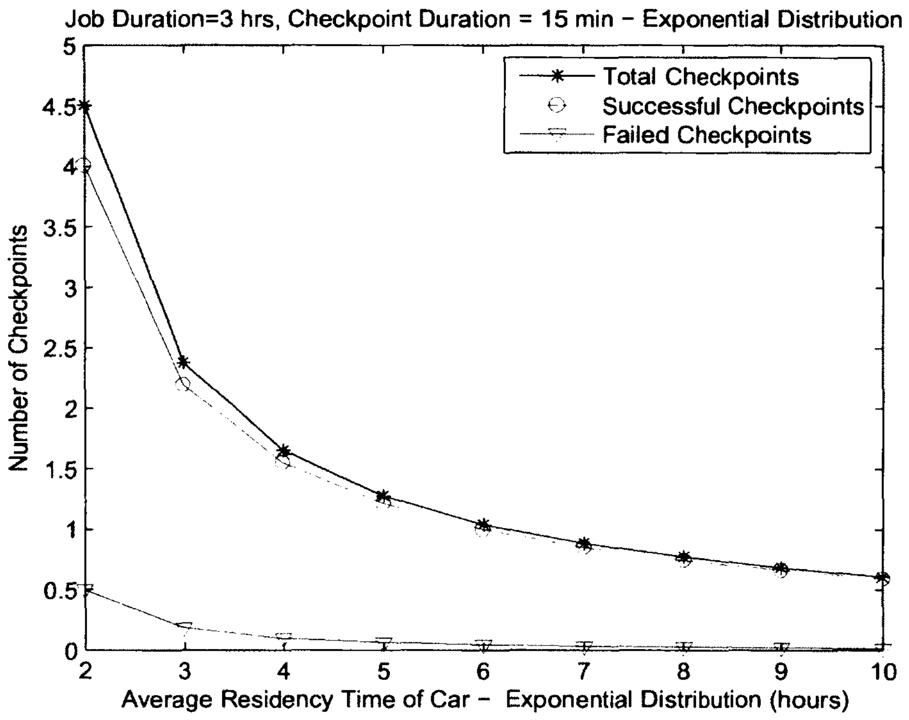


Fig. 73: Scenario 2. Illustrating the Number of checkpoints.

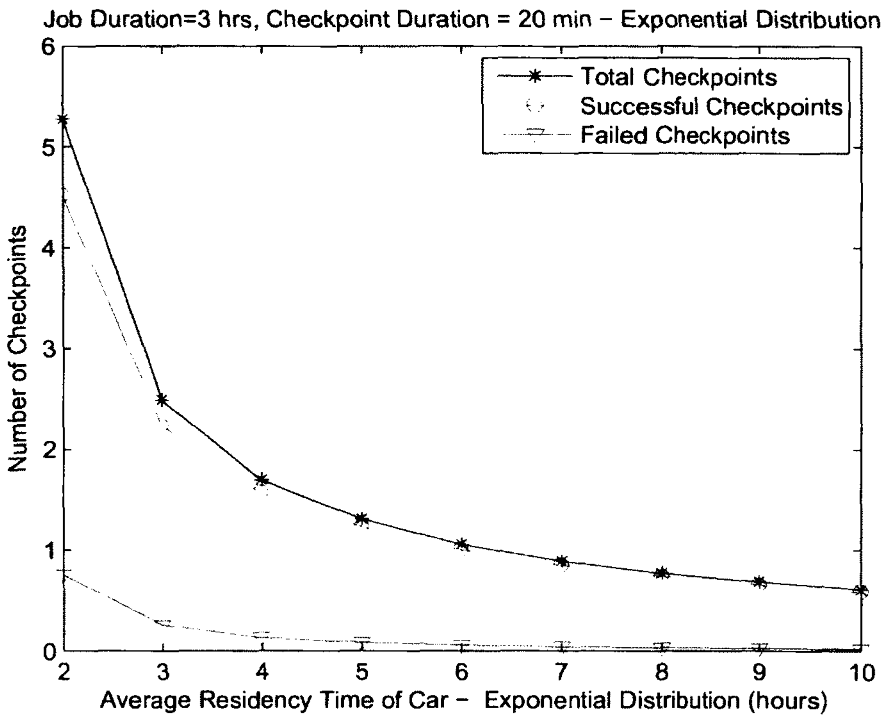


Fig. 74: Scenario 3. Illustrating the Number of checkpoints.

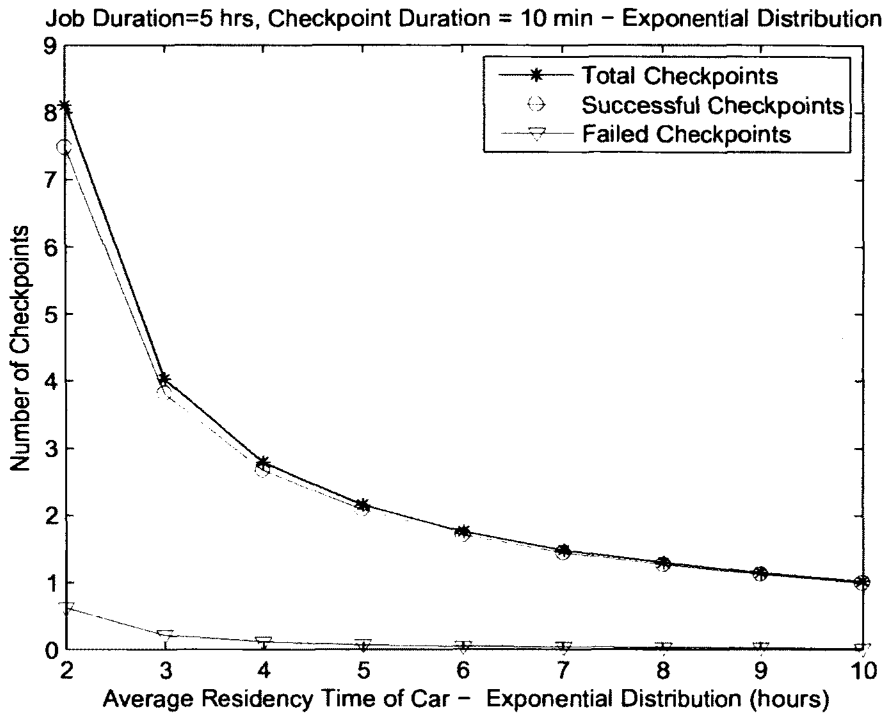


Fig. 75: Scenario 4. Illustrating the Number of checkpoints.

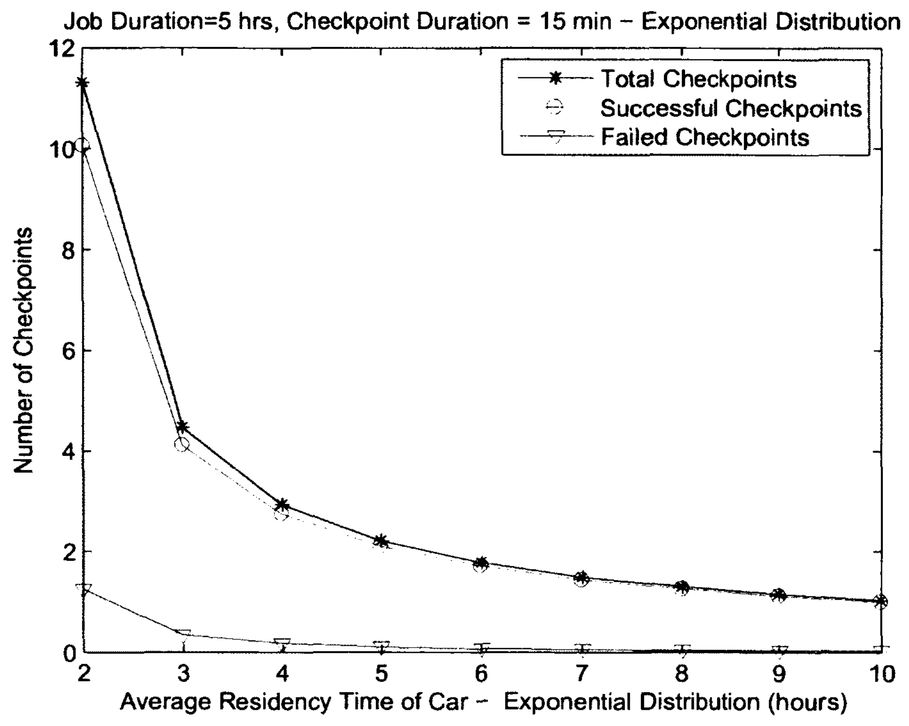


Fig. 76: Scenario 5. Illustrating the Number of checkpoints.

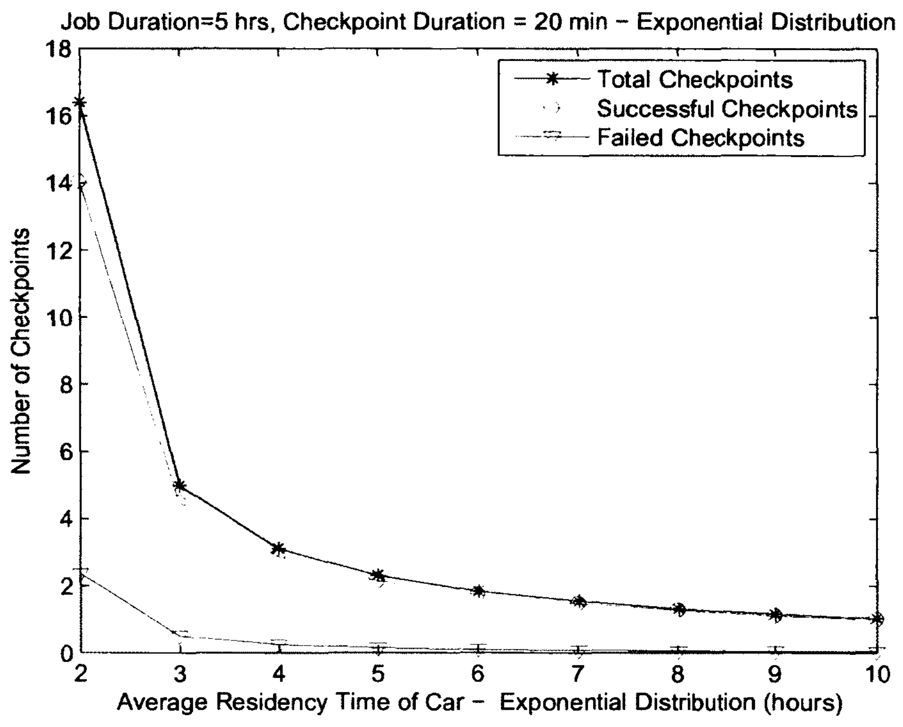


Fig. 77: Scenario 6. Illustrating the Number of checkpoints.

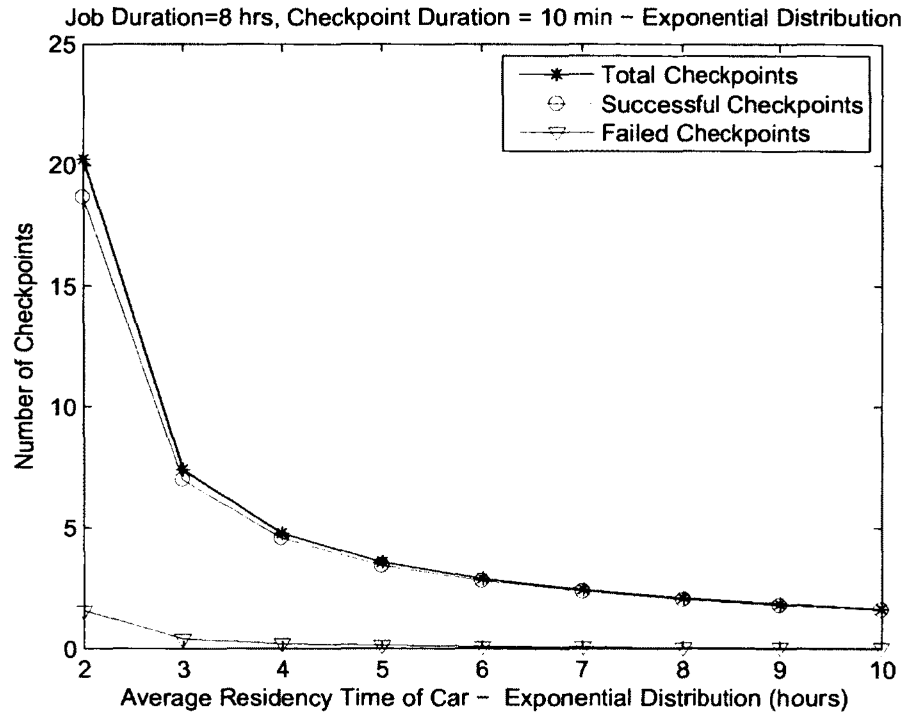


Fig. 78: Scenario 7. Illustrating the Number of checkpoints.

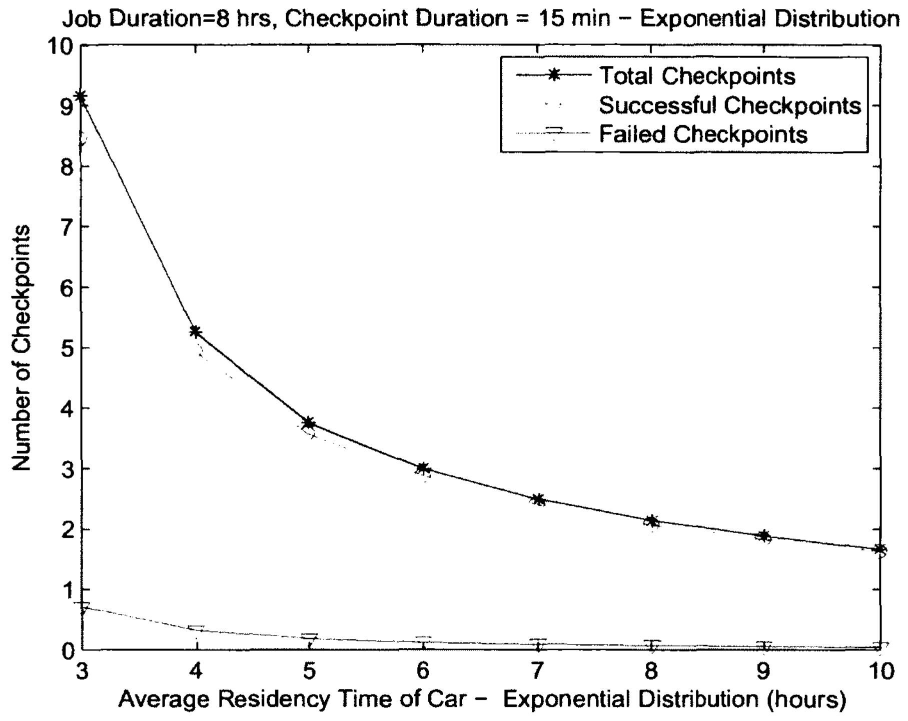


Fig. 79: Scenario 8. Illustrating the Number of checkpoints.

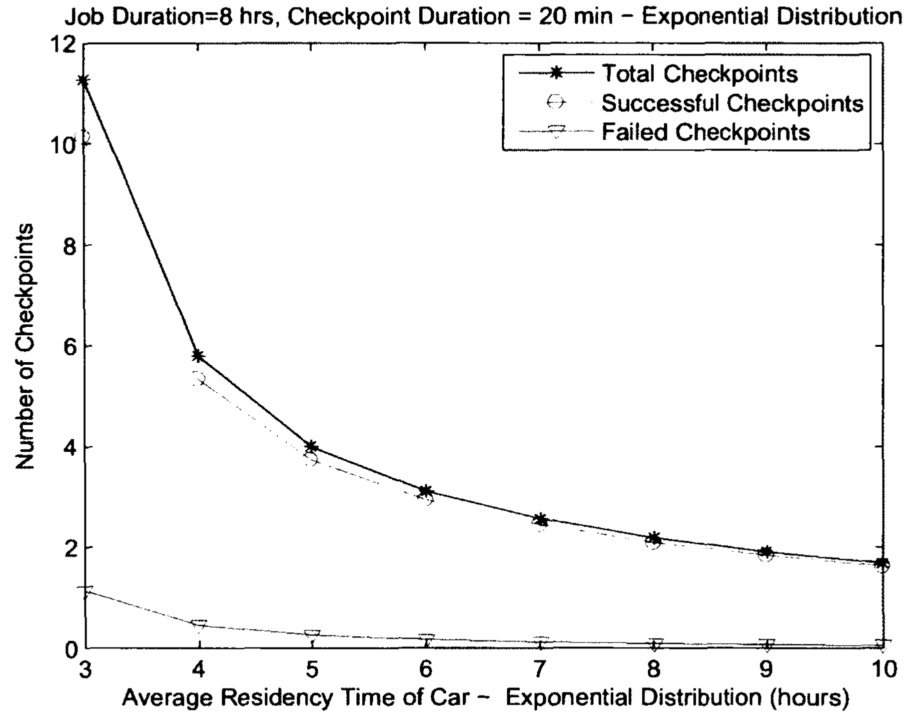


Fig. 80: Scenario 9. Illustrating the Number of checkpoints.

CHAPTER 8

CONCLUDING REMARKS AND DIRECTIONS FOR FUTURE WORK

The main goal of this chapter is to put the work done in this dissertation in perspective. We will present the problems that we have addressed and the solutions we have proposed. We then will point out features that will be implemented as part of future work. This chapter is organized as follows: Section 8.1 summarizes the results and the main conclusions of this dissertation. Future extensions and developments of this work are discussed in Section 8.2.

8.1 CONCLUDING REMARKS

Nowadays, vehicles are beyond transportation machines. In fact they are the machines with high computing powers. People can get different services (navigation, weather forecast, entertainment, etc) while they are in the vehicle as a driver or passenger. In the other hand advances of cloud computing have encouraged companies and users to move their computations, IT services or digital entertainments to the cloud. The main factor of cloud computing is computation power. Vehicles in parking lots could be potential resources for computation power [5,69,70].

Recently a new concept Vehicular Cloud Computing has been introduced to the literature. Vehicular Cloud is a network of vehicles on road or in parking lot that can provide computation services to users. In this model each vehicle is a computation node.

In this thesis, we addressed resource allocation in vehicular cloud computing. The main difference between vehicular cloud and traditional cloud is in resource availability. In traditional cloud, computation nodes are available permanently unless there is a hardware failure. Clouds are designed to deliver highly available computing services with hardwares that are highly reliable and their failure rate is very small. In other words computation nodes are available all the time. In the other hand vehicular cloud computation nodes are on-board devices that are installed in cars. Random arrival and departure of cars create a dynamic environment in terms of resource availability. This dynamic environment can cause failure in executing jobs because cars may randomly leave during running jobs.

Consequently the computation assigned to those car will be lost. In order to handle failure in this environment, we have presented different approaches based on checkpointing and redundancy and migration.

This work investigates job scheduling problem and resource allocation in vehicular cloud established in parking lots (e.g. mall, university and parking lots in airport). We believe it is only a matter of time before the huge vehicular fleets on our roadways, streets and parking lots will be recognized as an abundant and underutilized computational resource that can be tapped into for the purpose of providing third-party or community services. It is common knowledge that many of these vehicles spend hours each day in a parking garage, parking lot, or driveway. The computational and storage capabilities of these parked vehicles is a vast untapped resource that, at the moment is wasted. We conjecture that, given the right incentives, the owner of a vehicle may decide to rent out excess on-board capabilities, just as the owners of large computing or storage facilities find it economically appealing to rent out their excess capacity.

We considered two different environments for scheduling strategy: deterministic and stochastic. In deterministic environment the arrival and departure of cars are known. This scenario is for environment like universities where employees should be present at worked with known schedules and university rents out its employees cars as computation nodes to create to provide service as a vehicular cloud. We presented a scheduling model for vehicular cloud based on mixed integer linear programming. This work investigates a job scheduling problem involving non-preemptive tasks with known processing time where job migration is allowed. Assigning a job to resources is valid if job has been executed fully and continuously (no interruption). A job can not be executed in parallel. In our approach, the determination of an optimal job schedule can be formulated as maximizing the utilization of VC and minimizing the number of job migrations. Utilization can be calculated as a time period that vehicles have been used as computation resources.

For dynamic environment in terms of resource availability, we presented a stochastic model for job assignment. we proposed to make job assignment in VC fault tolerant by using a variant of the checkpointing strategy. Rather than saving the state of the computation at regular times, the state of the computation is only recorded as needed. Also, since we do not assume a central server that stores checkpointed images like conventional cloud providers do [43,44]. In our strategy checkpointing is performed by a car and the resulting image is stored by the car itself. Once the car leaves, the image is lost. We consider two scenarios: in the first one cars do not leave during checkpointing; in the second one cars may leave during checkpointing, leading to system failure. Our main contribution is to offer theoretical predictions of the job execution time in both scenarios mentioned above. A comprehensive set of simulations have shown that our theoretical predictions are accurate.

8.2 DIRECTION FOR FUTURE WORK

Vehicular cloud as a new concept can inspire and provide many opportunities for scientists in different research areas. In this section, we describe future research directions to extend the concept of vehicular cloud and bring it closer to real-world deployment.

8.2.1 VC PERFORMANCE FOR DIFFERENT APPLICATIONS

Jobs can be classified to different categories based on resources usage: CPU-intensive, Memory-intensive, Data-intensive, Network-intensive, etc. Analyzing vehicular cloud performance for different type of the jobs is another challenge that we consider as a future work. Result of this analysis will determine if there is a need to add specific elements to vehicular cloud for a specific type of jobs to have higher performance.

8.2.2 CLUSTERING IN VEHICULAR CLOUD

Clustering has been a solution in distributed environments for load balancing and recovering from failure. Number of resources in vehicular cloud are dependent on arrival and departure rate of vehicles. Therefore there could be shortage of resources in specific period of time in some VCs. Clustering will allow VC to solve this problem by using resources from other vehicular clouds in the cluster. As a future work we are interested in investigating the impact of clustering in vehicular clouds.

8.2.3 VEHICULAR CLOUD OF HYBRID CARS

Most of the car manufactures have introduced their own model of hybrid car to the market and in near future hybrid cars will be the ultimate choice for consumers [71]. Hybrid cars are equipped with high capacity batteries and they are able to save electricity for a long period of time. Therefore they are good candidates in providing long term electricity power for computation purposes. As a plan for future work, we are interested in considering the vehicular cloud based on hybrid cars and determine the advantage and disadvantages of this model.

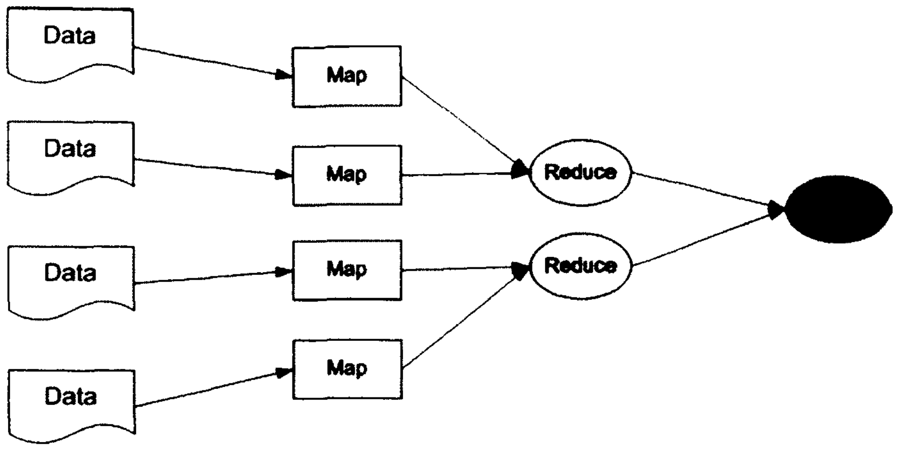


Fig. 81: Hadoop Architecture

8.2.4 IMPLEMENTATION OF HADOOP ON VC

Map-Reduce is a programming model that is used to process large datasets. This model parallelizes the data processing through distributed cluster of computation nodes [72, 73]. This model is based on defining two functions: Map function and Reduce function. Map-Reduce is based on functional programming, which the output of each map function is an input for reduce function. Map will sort the data based on keys that has been defined by user. The sorted data will be sent to reducers corresponding with sorting key.

Hadoop in open-source implementation of Map-Reduce [74]. Hadoop has two types of nodes: Master and Slave. Master divides the large dataset to small chunks. After reading each chunk, Master will send the data to corresponding Slave based on the key that has been defined by user. Slaves after processing the data will combine the result. Figure 81 illustrates the architecture of Hadoop.

Our goal is to adapt Hadoop framework to vehicular cloud computing. In our approach, we put slave nodes (reducers) in the stable number of vehicles. In Figure 82, we show a diagram that represents virtual data of a parking lot changing with time over one day. The maximum number of parking spots is 100. Measurement is done every hour. Sometimes, number of arriving vehicles is equal to leaving vehicles.

Although, number of vehicles in some cases is stable but there might be a Virtual Machine (VM) migration. Consider a scenario that the number of vehicles leaving at any hour is equal to number of vehicles arriving. In this case, we have the VM migration which takes place in short period of time [75].

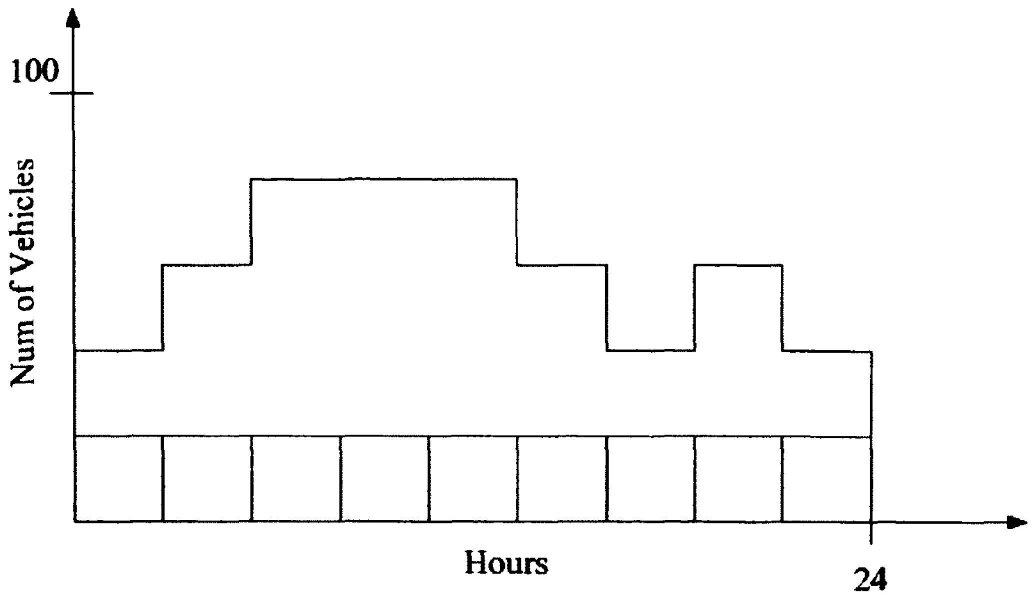
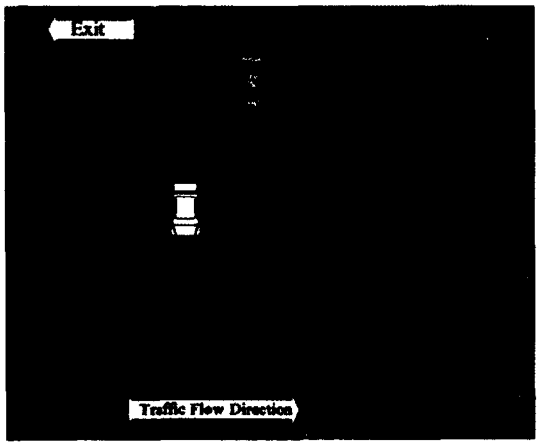


Fig. 82: Illustrating the migration of VMs in a parking lot.

In Figure 83, we explain the concept of migration of VM in vehicles located in a parking lot. It is noticed that the total number of vehicles in Figure 83- part A (left side) is equal to the total number of vehicles in Figure 83- part B (right side) but the VM which represents the vehicles has changed.

We assume that for each parking lot, a minimum number of vehicles are maintained. Our assumption is real in case of an airport parking lot. In order to implement Hadoop in vehicular cloud, we assign slave nodes to set of minimum number of vehicles (in Figure 82 the slave nodes have been recognized by shadow). The remaining number of vehicles represents the master nodes in our system (in Figure 82 the master has been shown without shadow). The moment that master is done with the task of reading the data, it gives the permission to slaves to start reading master. After finishing the reading process, master will start new task which is independent from the last task.

Based on the type of VC, number of master and slave nodes will be different. In other word, we can change the number of maps and reducers depending on actual number of vehicles in the parking lot. As a plan of work we will explore the optimum number of master and slave nodes for vehicular cloud.



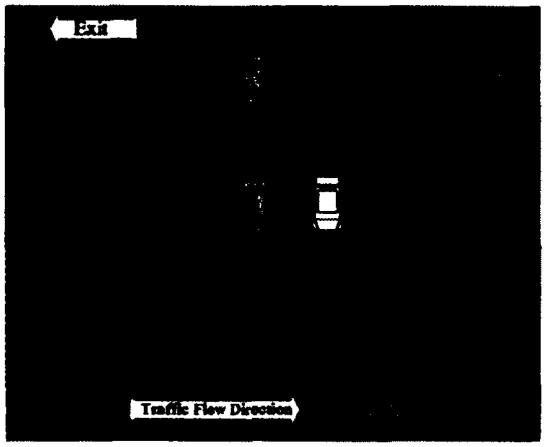


Fig. 83: Illustrating Virtual Data of Parking lot (Number of vehicles vs. Hours).

APPENDIX

In Section 5 we assumed that cars may leave during checkpointing. If cars leave during checkpointing failure will happen. In order to calculate the mean time to failure we need to calculate the distribution of the absolute value of difference between cars' residency time.

In the following derivation X and Y are standing for the residency times of the two cars currently executing the job. $Z = |X - Y|$ is the random variable that keeps track of the duration of the next checkpoint.

Consider independent exponential random variables X and Y with parameters λ and μ, respectively and write

$$Z = |X - Y|.$$

The main goal of this short note is to evaluate the distribution function F_Z of Z. For this purpose, given a non-negative real number z, we are interested in evaluating

$$F_Z(z) = \Pr[Z > z] = \Pr[|X - Y| \leq z].$$

As is usually the case, we will find it convenient to evaluate $P[|X - Y| > z]$ first. Let $f_{X,Y}(x,y)$ be the joint distribution function of X and Y. Since X and Y are independent we have

$$f_{X,Y}(x,y) = f_X(x) f_Y(y)$$

where $f_X(x)$ and $f_Y(y)$ are the marginal density functions. Moreover, since X and Y are exponential with parameters λ and μ we can write

$$f_{X,Y}(x,y) = \lambda e^{-\lambda x} \mu e^{-\mu y}.$$

Now consider the planar domain D defined as follows

$$D = \{(x,y) \mid x,y \geq 0;\ |x-y| > z\}.$$

Referring to Figure 84 it is easy to see that D decomposes into disjoint domains D_1 and D_2 such that

$$D_1 = \{(x,y) \mid x \geq y;\ x - y - z > 0\}$$

and

$$D_2 = \{(x,y) \mid y \geq x;\ y - x - z > 0\}.$$

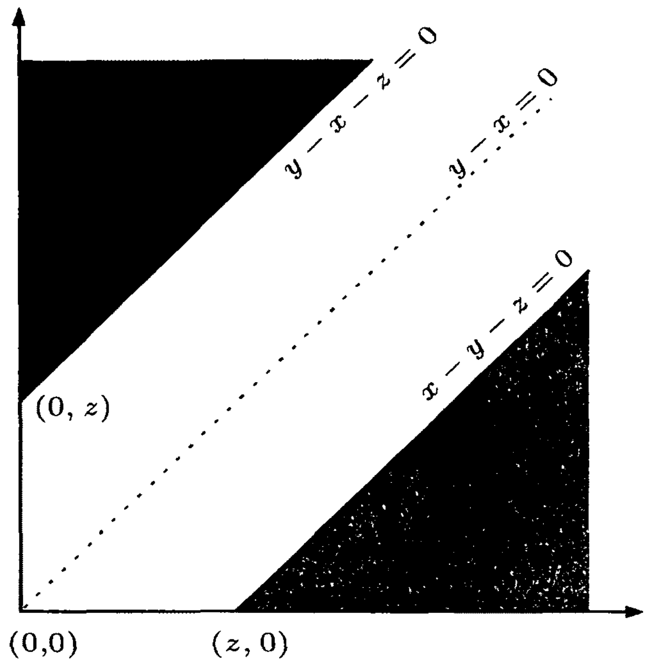


Fig. 84: Illustrating the sub-domains D_1 and D_2.

Since D_1 and D_2 are disjoint,

$$\iint\limits_{D} f_{X,Y}(x,y)\,\mathrm{d}x\,\mathrm{d}y = \iint\limits_{D_1} f_{X,Y}(x,y)\,\mathrm{d}x\,\mathrm{d}y + \iint\limits_{D_2} f_{X,Y}(x,y)\,\mathrm{d}x\,\mathrm{d}y.$$

With this notational preamble out of the way, we are ready to evaluate $P[|X-Y|>z]$.

$$\begin{aligned} P[|X-Y|>z] &= \iint\limits_{D} f_{X,Y}(x,y)\,\mathrm{d}x\,\mathrm{d}y \\ &= \iint\limits_{D_1} f_{X,Y}(x,y)\,\mathrm{d}x\,\mathrm{d}y + \iint\limits_{D_2} f_{X,Y}(x,y)\,\mathrm{d}x\,\mathrm{d}y. \end{aligned}$$

We now proceed by evaluating $I_1 = \iint_{D_1} f_{X,Y}(x,y)\,\mathrm{d}x\,\mathrm{d}y$ and $I_2 = \iint_{D_2} f_{X,Y}(x,y)\,\mathrm{d}x\,\mathrm{d}y$

separately. For this purpose we write

$$\begin{aligned} I_1 &= \int_{x=z}^{\infty} \lambda e^{-\lambda x} \int_{y=0}^{x-z} \mu e^{-\mu y} \mathrm{d}y \mathrm{d}x \\ &= \cdots \\ &= \frac{\mu}{\lambda+\mu} e^{-\lambda z} \end{aligned}$$

Similarly,

$$\begin{aligned} I_2 &= \int_{y=z}^{\infty} \mu e^{-\mu y} \int_{0=0}^{y-z} \lambda e^{-\lambda x} \mathrm{d}x \mathrm{d}y \\ &= \cdots \\ &= \frac{\lambda}{\lambda+\mu} e^{-\mu z} \end{aligned}$$

It follows that

$$P[|X-Y|>z] = \frac{\mu}{\lambda+\mu} e^{-\lambda z} + \frac{\lambda}{\lambda+\mu} e^{-\mu z}.$$

In the particular case where $\lambda = \mu$, i.e. X and Y are identically distributed, we write

$$P[|X-Y|>z] = \frac{1}{2} e^{-\lambda z} + \frac{1}{2} e^{-\lambda z} = e^{-\lambda z}$$

and, consequently,

$$P[|X-Y| \le z] = 1 - e^{-\lambda z}.$$

This result is only mildly surprising given the memoryless property of the exponential distribution.

Based on this derivation if the residency times are i.i.d., as we assume them to be, then the distribution of the absolute value of their difference is again exponentially distributed with the same parameter.

REFERENCES

[1] S. Olariu, I. Khalil, and M. Abuelela, "Taking vanet to the clouds," *International Journal of Pervasive Computing and Communication*, vol. 7, no. 1, pp. 7–21, 2011.

[2] S. Olariu, M. Eltoweissy, and M. Younis, "Towards autonomous vehicular clouds," *ICST Transactions on Mobile Communications and Computing*, vol. 11, no. 7-9, pp. 1–11, July-September 2011.

[3] S. Olariu, T. Hristov, and G. Yan, "The next paradigm shift: From vehicular networks to vehicular clouds," in *Stojmenovic I. and Basagni, S., (Eds), Mobile Computing, Wiley and Sons, New York*, 2014.

[4] S. Arif, S. Olariu, J. Wang, G. Yan, W. Yang, and I. Khalil, "Datacenter at the airport: Reasoning about time-dependent parking lot occupancy," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 11, pp. 2067–2080, 2012.

[5] M. Fontaine, *Traffic monitoring: in Vehicular Networks: From Theory to Practice*, S. Olariu and M. C. Weigle, Eds. Taylor and Francis, 2009.

[6] US Department of Transportation, "Catastrophic hurricane evacuation plan evaluation: A report to congress," http://www.fhwa.dot.gov/reports/hurricanevacuation/, June 2006.

[7] National Institute of Standards and Technology (NIST), "NIST definition of cloud computing," http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc, October 2009.

[8] J. Foley, "Private clouds take shape," *Information Week*, August 2008.

[9] Hodgson S., "What is cloud computing?" http://www.winextra.com/2008/05/02/what-is-cloud-computing.pdf, May 2, 2008.

[10] W. Kim, "Cloud computing: Today and tomorrow," *Journal of Object Technology*, vol. 8, no. 1, pp. 65–72, Jan. 2009, (column). [Online]. Available: http://www.jot.fm/contents/issue_2009_01/column4.html

[11] J. N. Hoover and R. Martin, "Demystifying the cloud," *InformationWeek Research and Reports*, pp. 30–37, June 2008.

[12] W. Kim, "Cloud computing: today and tomorrow," *Journal of Object Technology*, vol. 8, no. 1, pp. 65–72, January-February 2009.

[13] Woodford, Inc., "Cloud computing explained," http://www.explainthatstuff.com/cloud-computing-introduction.html, Feb. 5 2010.

[14] M. Rouse, "Software as a service (saas)," Aug 2010. [Online]. Available: http://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service

[15] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[16] J. Varia, "Amazon web services. cloud architectures," 2008, http://media.amazonwebservices.com/AWS_Cloud_Architectures.pdf/.

[17] "Beyond vm." [Online]. Available: http://www.beyondvm.com/what-is-virtualization/

[18] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, ser. NSDI'05. Berkeley, CA, USA: USENIX Association, 2005, pp. 273–286. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251203.1251223

[19] "Understanding clones." [Online]. Available: http://www.vmware.com/support/ws55/doc/ws_clone_overview.html

[20] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast transparent migration for virtual machines," in *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ser. ATEC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 25–25. [Online]. Available: http://dl.acm.org/citation.cfm?id=1247360.1247385

[21] "Siometer project." [Online]. Available: http://www.iometer.org/

[22] "Memtest86 - a stand-alone memory diagnostic." [Online]. Available: http://www.memtest86.com/

[23] "Microsoft sql server: Resource kit." [Online]. Available: http://www.microsoft.com/sql/techinfo/reskit

[24] M. Abuelela and S. Olariu, "Content delivery in zero-infrastructure VANET," in *Olariu S. and Weigle M.C., (Eds),* Vehicular Networks: From Theory to Practice, *Taylor and Francis, Boca Raton*, 2009, pp. 8.1 – 8.15.

[25] J. Eriksson, H. Balakrishnan, and S. Madden, "Cabernet: Vehicular content delivery using wifi," in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, ser. MobiCom '08. New York, NY, USA: ACM, 2008, pp. 199–210. [Online]. Available: http://doi.acm.org/10.1145/1409944.1409968

[26] L. Le, A. Festag, R. Baldesari, and W. Zhang, *Car-2-X Communications in Europe: in Vehicular Networks: From Theory to Practice*, in S. Olariu and M. C. Weigle, Eds. Taylor and Francis, 2009.

[27] S. Olariu and M. C. Weigle, *Vehicular Networks: From Theory to Practice*, 1st ed. Chapman & Hall/CRC, 2009.

[28] US Department of Transportation, Research and Innovative Technology Association, "National transportation statistics," http://www.bts.gov/publications/national_transportation_statistics/, 2011.

[29] NHTSA National Highway Traffic Safety Administration, "Traffic safety facts - preliminary 2009 report," http://www-nrd.nhtsa.dot.gov/Pubs/811255.pdf, March 2010.

[30] , "Tropos networks," http://www.tropos.com/pdf/solutions/Parking-Final.pdf, 2010.

[31] "National Motor Vehicle Crash Causationon Survey," US Department of Transportation, Intelligent Transportation Systems Field Operational Test Cross-Cutting Study, July 2008.

[32] US Department of Transportation, "Federal-aid highway program guidance on high occupancy vehicle (hov) lanes," http://ops.fhwa.dot.gov/freewaymgmt/hovguidance/index.htm, August 2008.

[33] Virginia Department of Transportation, "Commonwealth of virginia's strategic highway safety plan," http://virginiadot.org/info/resources/Strat_Hway_Safety_Plan_FREPT.pdf, 2006.

[34] US Department of Transporation, National Highway Traffic Safety Administration, "Event data recorders," www.nhtsa.gov/DOT/NHTSA/Rulemaking/Rules/Associated/%20Files/EDRFinalRule_Aug2006.pdf, Aug. 2006, [Docket No. NHTSA-2006-25666], 49 CFR Part 563.

[35] Y. Yang and R. Bagrodia, "Evaluation of vanet-based advanced intelligent transportation systems," in *Proceedings of the Sixth ACM International Workshop on VehiculAr InterNETworking*, ser. VANET '09. New York, NY, USA: ACM, 2009, pp. 3–12. [Online]. Available: http://doi.acm.org/10.1145/1614269.1614273

[36] J. Anda, J. LeBrun, D. Ghosal, C.-N. Chuah, and M. Zhang, "Vgrid: vehicular adhoc networking and computing grid for intelligent traffic control," in *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, vol. 5, May 2005, pp. 2905–2909 Vol. 5.

[37] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid information services for distributed resource sharing," in *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on*, 2001, pp. 181–194.

[38] V. J. Misener J.A., Dickey S. and S. R., *Vehicular Networks: From Theory to Practice to Practice*, V.-I. C. in Olariu S. and E. Weigle M.C., Eds. Taylor and Francis, CRC Press, Boca Raton, 2009.

[39] US Department of Transporation, Research and Innovative Technology Association, http://www.its.dot.gov/connected_vehicle/connected_vehicle_apps.htm#sthash.Fu4xwxqW.dpu.

[40] M. Rosenblum and T. Garfinkel, "Virtual machine monitors: Current technology and future trends," *Computer*, vol. 38, no. 5, pp. 39–47, 2005.

[41] J. Wei, X. Zhang, G. Ammons, V. Bala, and P. Ning, "Managing security of virtual machine images in a cloud environment," in *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, ser. CCSW '09. New York, NY, USA: ACM, 2009, pp. 91–96. [Online]. Available: http://doi.acm.org/10.1145/1655008.1655021

[42] J. S. Reuben, "A survey on virtual machine security," *Helsinki University of Technology*, 2007.

[43] Citrix, "Virtualization, networking and cloud computing," http://www.citrix.com, 2009.

[44] VMware, "Vmware virtualization software for desktops, servers & virtual machines for virtual and public cloud," http://www.vmware.com, 2009.

[45] P. Ghazizadeh, R. Mukkamala, and S. Olariu, "Data integrity evaluation in cloud database-as-a-service," in *Services (SERVICES), 2013 IEEE Ninth World Congress on*, June 2013, pp. 280–285.

[46] L. F. Sarmenta, "Sabotage-tolerance mechanisms for volunteer computing systems," *Future Generation Computer Systems*, vol. 18, no. 4, pp. 561 – 572, 2002, best papers from Symp. on Cluster Computing and the Grid (CCGrid2001). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X01000772

[47] Fred Dews, "Ninety percent of americans drive to work," http://www.http://www.brookings.edu/, 2013.

[48] I. Scarborough Research/Arbitron, "Teen mall shopping insights," A white paper, June 2009.

[49] J. Skolnik, R. Chami, and M. Walker, "Planned special events - economic role and congestion effects," Federal Highway Administration, U.S. Department of Transportation, Washington, D.C., Tech. Rep. FHWA-HOP-08-022, 2008.

[50] US Department of Transporation, "Intelligent transportation systems for planned special events: A cross-cutting study," Federal Highway Administration, Washington, D.C., Tech. Rep. FHWA-JPO-08-056, 2008.

[51] P. Brucker, J. Hurink, B. Jurisch, and B. Wstmann, "A branch and bound algorithm for the open-shop problem," *Discrete Applied Mathematics*, vol. 76, no. 13, pp. 43 – 59, 1997, ¡ce:title¿Second International Colloquium on Graphs and Optimization¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166218X96001163

[52] S.-C. Lin, E. D. Goodman, and W. F. Punch, "A genetic algorithm approach to dynamic job shop scheduling problems," in *Seventh International Conference on Genetic Algorithms*. Morgan Kaufmann, 1997, pp. 139–148.

/

[53] H. Zhou and Y. Feng, "The hybrid heuristic genetic algorithm for job shop scheduling," *Computers and Industrial Engineering*, vol. 40, no. 3, pp. 191 – 200, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360835201000171

[54] H. Zhou, Y. Feng, and L. Han, "The hybrid heuristic genetic algorithm for job shop scheduling," *Computers & Industrial Engineering*, vol. 40, no. 3, pp. 191–200, 2001.

[55] Z. Guo, W. Wong, S. Leung, J. Fan, and S. Chan, "A genetic-algorithm-based optimization model for scheduling flexible assembly lines," *The International Journal of Advanced Manufacturing Technology*, vol. 36, no. 1-2, pp. 156–168, 2008.

[56] M. Dell'Amico and M. Trubian, "Applying tabu search to the job-shop scheduling problem," *Annals of Operations Research*, vol. 41, no. 3, pp. 231–252, 1993.

[57] M. Kolonko, "Some new results on simulated annealing applied to the job shop scheduling problem," *European Journal of Operational Research*, vol. 113, no. 1, pp. 123–136, 1999.

[58] Y. Chen, Z. Guan, and X. Shao, "A comparative analysis of job scheduling algorithm," in *Management Science and Industrial Engineering (MSIE), 2011 International Conference on*, 2011, pp. 1091–1095.

[59] F. Hillier and G. Lieberman, *Introduction to Operations Research*, ser. Introduction to Operations Research. McGraw-Hill Higher Education, 2010. [Online]. Available: http://books.google.com/books?id=NvE5PgAACAAJ

[60] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2013. [Online]. Available: http://www.gurobi.com

[61] J. Deng, S.-H. Huang, Y. Han, and J. Deng, "Fault tolerant and reliable computation in cloud computing," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, Dec 2010, pp. 1601–1605.

[62] I. Egwutuoha, S. Chen, D. Levy, and B. Selic, "A fault tolerance framework for high performance computing in cloud," in *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, May 2012, pp. 709–710.

[63] S. Y. Ko, I. Hoque, B. Cho, and I. Gupta, "Making cloud intermediate data fault-tolerant," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 181–192.

[64] S. Malik and F. Huet, "Adaptive fault tolerance in real time cloud computing," in *Services (SERVICES), 2011 IEEE World Congress on*, July 2011, pp. 280–287.

[65] J. Park, H. Yu, K. Chung, and E. Lee, "Markov chain based monitoring service for fault tolerance in mobile cloud computing," in *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, March 2011, pp. 520–525.

[66] Y. A. Zuev, "On the estimation of efficiency of voting procedures," *Theory of Probability & Its Applications*, vol. 42, no. 1, pp. 73–81, 1998.

[67] Y. Zuev and S. Ivanov, "The voting as a way to increase the decision reliability," *Journal of the Franklin Institute*, vol. 336, no. 2, pp. 361 – 378, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0016003298000258

[68] K. Watanabe, M. Fukushi, and S. Horiguchi, "Expected-credibility-based job scheduling for reliable volunteer computing," *IEICE transactions on information and systems*, vol. 93, no. 2, pp. 306–314, 2010.

[69] S. Olariu and M. C. Weigle, *Vehicular Networks: From Theory to Practice*. CRC Press, Boca Raton, 2009.

[70] X. Wu, P. Michalopoulos, and H. X. Liu, "Stochasticity of freeway operational capacity and chance-constrained ramp metering," *Transportation Research Part C*, vol. 18, pp. 741–756, 2010.

[71] , "Sightline," http://www.sightline.org/research/energy/respubs/analysis-ghg-roads, 2009.

[72] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[73] H.-c. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker, "Map-reduce-merge: simplified relational data processing on large clusters," in *Proceedings of the 2007*

ACM SIGMOD international conference on Management of data, ser. SIGMOD '07. New York, NY, USA: ACM, 2007, pp. 1029–1040. [Online]. Available: http://doi.acm.org/10.1145/1247480.1247602

[74] "Hadoop." [Online]. Available: http://hadoop.apache.org/

[75] S. Kikuchi and Y. Matsumoto, "Performance modeling of concurrent live migration operations in cloud computing systems using prism probabilistic model checker," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, 2011, pp. 49–56.

VITA

Puya Ghazizadeh
Department of Computer Science
Old Dominion University
Norfolk, VA 23529

Puya Ghazizadeh received his Bachelors Degree in Civil Engineering with honors from University of Kurdistan in 2005. In 2009, he received the Master Degree in Computer Engineering from Science and Research Branch of Azad University, Tehran. His Master's thesis was focused on Modeling Consumer Behavior with Multi-Objective Genetic Fuzzy and PSO Fuzzy Systems for Market Prediction. After that he joined the PhD program in the Computer Science Department in Old Dominion University where he decided to pursue his dissertation under the supervision of Professor Ravi Mukkamala and Professor Stephan Olariu in Vehicular Cloud Computing.

Puya's main research interests include a wide variety of Cloud Computing topics such as Scheduling, Fault Tolerant Computing, Resource Management and Outsourced Database Security and Privacy.

In 2011, Puya's was selected as a recipient for the Outstanding Computer Science Teaching Assistant Award given by the Department of Computer Science. He also was awarded a Certificate of Participation in 2014 Graduate Research Achievement Day (GRAD) by Old Dominion University. Puya's research work and ideas were well received by the Cloud Computing, Security and Privacy and Modeling research communities, with over 8 publications in various IEEE and ACM, conferences and workshops at the time of writing this thesis.

Typeset using LaTeX.