Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 10, No. 4, November 2019), Pages: 97-106
www.jacr.iausari.ac.ir

A New Replica Consistency Maintenance Method Based On Dynamic Update Rate Calculation in the Data Grid

Rasmieh Parvaneh[1] ,Mahsa Beigrezaei ✉[2]
1) Department of Computer and Engineering, Gilan-E Gharb Branch, Islamic Azad University, Gilan-E Gharb, Iran
2) Department of Computer and Engineering., Yadegar -E- Imam Khomeini (RAH) Shahr-E-Rey Branch, Islamic Azad University, Tehran, Iran
rasmieh.parvaneh@gmail.com; m.beigrezai@gmail.com

Received: 2019/01/22; Accepted: 2019/10/15

Abstract

Data Grid provides convenient services for data management and uniform access to distributed high volume data sources. Data replication is a service to facilitate and accelerate data accessing in Data Grid. But in the large-scale grid environment, the need to maintain replica consistently and ensure identical content of the various replica of a file are inevitable. This paper presents an effective consistency method that maintains replicated data consistent. In this method, the node which contains replica is responsible for managing consistency. In each node, the replica update rate is calculated based on the rate of change in the original data file and the rate of the access request to the existing replica version. The proposed model reduces access delays. The model was simulated using Potosi developed by European Data Grid projects. The experimental results show that our proposed model has better performance in comparison with optimistic and pessimistic approaches in terms of job execution time, effective network usage, and the total number of updates.

Keywords: *Data Grid, Data Replication, Replica Consistency, Update Rate*

1. Introduction

Data Grid aims to provide massive storage space with proper facilities to manage high volume data that is not possible to be implemented in a centralized environment [1-6]. Grid data can be used to help the large volume of data be stored in various locations of the grid and then be retrieved. Since data is continuously transferred through different nodes in the grid, the efficiency of Data Grid is largely limited to the bandwidth among the nodes. Data replication methods have been presented to increase the efficiency and lower the impact of these restrictions; this means that several replicas of the original data files are stored in the Data GridGrid. In Data Grid, a file can have many replicas in different locations of the grid that each of them may change by user, so updating all replicas by an effective method is necessary .

Data replication [4] is an economical solution to improve and reform the access performance and reliability by means of duplicating original data in a distributed manner. A file on a grid system may have several duplicates called replicas in various

grid sites. In general, consistency and synchronization problems of the replica are not addressed in previous researches, as the file is regarded as read-only. Nevertheless, issues of replica coherence emerge if data are allowed to be modified by applications. All other replicas are obliged to update with the same up-to-date contents when the original file is modified. Keeping consistent content at all distributed replicas over several grid sites is an important subject in a wide-area computing environment. The replica consistency problem arises from the update synchronization of multiple replicas of a file [6-11]. In this paper, we proposed a new replica consistency maintenance method based on the dynamic update rate calculation in the Data Grid. In our method, the replica update rate is calculated based on the rate of change in the original data file and the rate of the access request to the existing replica version. The rest of this paper is organized as follows: Section 2 presents a brief introduction of related works. Section 3 presents proposed method In Section 4 the simulation results will be described. Finally Section 5 concludes our discussion and presents a few future works.

2. Related Works

Replica consistency methods have attracted much attention over the last decade. So far, many different methods have been proposed for maintaining replicas consistency in the Data Grid environment. The following, some of these methods are presented briefly. In the pessimistic approach, after making any changes, the contents of the original file will be transferred to all its replica to have all replica updated. This sometimes leads some versions to be updated that they are rarely or never used. This policy brings about together a waste of network resources and an increase in the time of work. Because after any changes, any other access to data is not allowed unless all replica is updated .

In the optimistic approach, the replica update will be delayed until it is needed [13-16]. This is done to prevent the unnecessary update. When a job needs to access to a replica of a specific data file, the replica is checked whether it's updated or not. If it is not updated, the access to it will be delayed until it is updated. In this case, in order to prevent waste of resources and unnecessary updates, access latency penalty will be given.

In [17], a consistency service called the grid consistency service has been introduced. This approach has provided various consistency levels (comprehensive to partial consistency) for different users. In this approach, only the original file can be changed, and the other replica is updated by grid consistency service according to the consistency level chosen by the user .

In [18], a combination of pessimistic and optimistic approaches has been used. In this method, nodes are topologically categorized into groups named network regions. Each of these regions has several secondary and a master file and at most. There is comprehensive consistency among the original and a few secondary versions that their weight access passes from a fixed threshold, and the partial consistency is established among the rest of the secondary versions .

In [20], a three-node topology with three categories, including supernode, a master node, and child node, has been presented. Only the data available on the supernode can

change in this method. Any changes are immediately applied to all replica available on master nodes and on some of the child nodes (based two factors: the access frequency and memory capacity of that node) – full consistency. The replica has partial consistency in the other child nodes.

A decision model of replica consistency has been proposed in [19]. This method uses Naive Baysian to decide which secondary replica, along with master versions, should be fully consistent and which of the secondary versions should have partial consistency .

In [20], nodes are divided into categories called virtual organizations. In each virtual organization, a tree with the minimum height is made out of the nodes containing replica to expedite the replica update. The root will be informed of any changes in any replica in each node of the tree, and the root will inform the other roots through peer to peer network that exists between the roots of trees; therefore the trees in each virtual organization begin to update their own replica in parallel. Some approaches have been presented in peer-to-peer systems that have avoided the problems of both optimistic and pessimistic approaches as much as possible so that they avoid transferring unnecessary data and simultaneously reduce the delay in access time. An approximation of the mentioned environments can be used in grid data due to the similarities between gird data and peer-to-peer systems. This paper has been given on Data Grid consistency regarding the similarities between gird data and peer to peer systems.

In [21], a new method for maintaining consistency of peer-to-peer file-sharing networks has been introduced; its main purpose is to make replica update rate in a node flexible considering the rate of change in the master file and the dynamic state of connection and disconnection of servers in peer-to-peer networks.

A comprehensive method in order to create and maintain replica consistency in peer-to-peer networks has been presented in [22]. The approaches used in this paper are inspired by [21]. In this approach, each customer begins to calculate each of its replica's update rate considering the change rate parameters and access to the replica. This will prevent unnecessary updates and reduce costs associated with the network. In most methods presented to maintain consistency in the grid, the manager or the one who is responsible for maintaining consistency sends the updated version to the location or locations in which the replica exists according to the policy type [22]. But in this paper, as the [22], the node itself is responsible for implementing the burden of maintaining consistency so that each node calculates the rate of its replica based on the two factors: the rate of change in the original data file and access to replica; these calculations are dynamic, and if any of the aforementioned factors change over time, the rate of replica update on the node will also change according to them. Therefore the updates are carried out neither like the optimistic approach when it is needed nor like the pessimistic approach with any changes occur. The update is done for every replica. But each node has its own criteria to decide when to update each of its replicas. The implementation of this policy avoids unnecessary updates and reduces the replica access time delay.

This paper divides into four parts: Part 1 gives a brief introduction of the related works, Part 2 introduces the proposed methods, Part 3 presents the results of the

simulation, and ultimately, Part 4 is devoted to conclusions and the description of IT works.

3. The Proposed Method

In this section, the proposed method is discussed. First, the architecture used is described; then, the proposed algorithm is described in detail.

3.1.The architecture of the proposed algorithm

The architecture used in this paper is the same as the architecture used in [13]. In this architecture, the number of sites that are geographically close to each other and high-bandwidth communication considered as a region. The connections have been established between regions with a low bandwidth network (usually by the Internet). In the architecture of this paper, each network region has a server named region server that the master files are stored on it. Each region should have an original copy of each file. The server is a node in which the capacity of its storage element is greater than the other nodes and has wider bandwidth with the other nodes in the region. Each region can have an unlimited number of replicas. Replicas can be accessed for both reading and writing. After any file changing, considered the original version of the file in the region server is informed and updated. The aforementioned regional server notifies other file serves from these updates and version number.

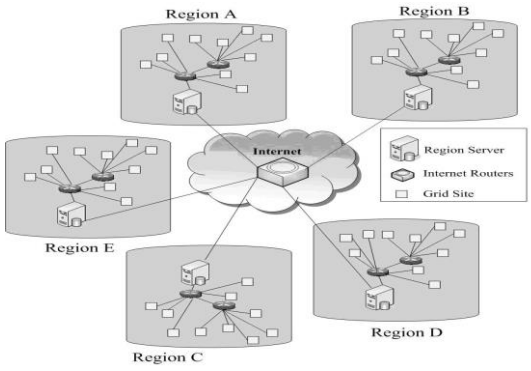

Figure 1. proposed architecture [18]

3.2. The proposed algorithm

As mentioned before, each node should decide when it updates its replicas. It tries to perform the minimum number of updates. Therefore, two questions should be answered first, "How to determine the fetching rate of the original data file to the node in order to update the replica periodically And the second, " How to reduce the rate of pre-fetching to decrease network costs? (to avoid unnecessary fetching)". This question is replied In the following.

3.2.1. Determining the fetching rate

It is assumed that the minimum time between two changes in the original file is T. indeed it means that the content of the original data file is changed once by the users in each T per time unit. Thus, if the node, which contains the replica, updates its replica

and original version with the rate of 1/T (fetching the content of the original data file into the node), it can make sure that its replica is never older than T per time unit. Since the speed of change is different over time, the nodes containing replica should be able to change the fetching rate to fit these changes. In fact, the node containing replica, by applying the linear increase algorithm and reducing the coefficient, will intelligently change the fetching rate so as to avoid unnecessary updates, yet it provides access to an updated version of the required data file. A value called the Time to update(TTU) is assigned to each version. TTU is the time interval between the present time and the next fetching time of the original data file in the node. TTU dynamically changes according to the results of any fetching. If the fetched data file has not changed between two consecutive fetchings (if the number of the fetching data files are the same), a constant value is added to TTU according to the formula (1). Where TTUOld is the latest TTU and I is the contest value.

$$TTU = TTU_{Old} + I \ , \ I > 0 \qquad (1)$$

When the version number of the current fetched file is greater than the previous fetched file, TTU value is reduced with a constant coefficient according to formula (2).). Where TTUOld is the latest TTU and D is the constant coefficient.

$$TTU = TTU_{Old} / D \ , \ D > 1 \qquad (2)$$

To TTU not to be very big to maintain its effectiveness and not to be very small to lose its optimal aspect, two thresholds, including TTU min and TTU max, are considered. Typically TTUmin or the value of T, is initialized with the shortest interval time between two changes in the original data file. Formula (3) shows the initialization for each replica.

$$TTU = TTU_{min} = T \qquad (3)$$

If, after the changes on TTU, its value is more than the higher threshold or is less than the lower threshold, then its value is calculated by the formula (4).

//(4) $$TTU = Max (TTU_{min} \ Min (TTU_{max}, TTU))$$

$$TTU = \begin{cases} TTU_{min} & TTU < TTU_{min} \\ & \\ TTU_{max} & TTU > TTU_{max} \end{cases} \qquad (4)$$

3-2-2 Fetching rate reduction

In calculating the fetching rate, in addition to the rate of changes of the original file, the rate of access to the replica of the file in the node is also an important factor for reducing the number of replica updates that should be noted. To calculate the fetching rate of the original file into the node (updates), a combination of access rate to the file and the rate of changes in the original file is used. If we consider TTUpoll as interval time between two file fetching, query the time between the two requests for access to

the replica in the node, and TTU the time between two modifications in the original data file then to calculate TTU poll we act as follow:

- If TTU <$T_{query,,}$ it means the rate of change in the original file is more than the access rate to the replica of the file, so there is no need to update the replica as same as the rate of change in the original file. Because if the replica is immediately updated after a change in the original version, but if there is no request over access to it, then the update would have been unnecessary. For example, suppose that the original data file changes once in a second while its replica in the node is accessed once in every two seconds, therefore the update of that replica once in every two seconds can also guarantee accuracy and consistency of that replica's content or the content of its original data file. So in cases where the rate of the access request to the replica of the data file is less than the rate of change in the original file, the fetching rate is initialized as equally as the rate of access request to the replica data file which is on that node.

- If TTU>T_{query}, it means the replica is accessed with a higher rate than the rate of changes in the original file, then the replica should be updated with a rate as equal as the rate of change in the original file. As a result, Troll should be calculated according to to the formula (5):

TTU <= $T_{query}T_{query}$

T_{poll} =T TTU >T_{quer}

$$TTU_{fetch}= \begin{cases} T_{request} & TTU <= T_{request} \\ T & TTU > T_{request} \end{cases} \qquad (5)$$

According to the above descriptions, it is seen that the number of fetching and overhead related to the replica update without causing an adverse effect on consistency, are reduced, and consistency can be guaranteed at a lower cost. It should be noted that this method does not guarantee that all access will be reached to the updated replica. Since the number of access to the old files is few, it's possible to neglect this disadvantage because this method results show improvement in the reduction of latency access, time of performance, and efficiency of network resources. Some advantages of the mentioned method are as the following:

- Each node can determine D, I parameters in order to determine the speed of increasing and decreasing after each fetching.

- It only needs to save the most recent TTU and T_{Query} and file version number along with the replica.

- This method is quite adaptable to dynamic replacement and placement algorithms. After producing a replica in a node, TTU will be initialized TTU_{min} for it.

4. Simulation Results

The "OptorSim" simulator has been used for simulation and evaluating the efficiency of the proposed method in this paper (OptorSim – A Replica Optimiser Simulation). OptorSim is a simulator written in Java [23-27]. The presented method has been compared with the optimistic and pessimistic methods in terms of the average job execution time and the number of updates. The test environment consists of four network regions. Each region has five sites. The resource specifications and job parameters are listed in Tables 1, respectively. The simulated grid used in our experiments has 20 sites, 18 of them have Storage Element (SE), and Computing Element (CE), and 2 of them have only SE. The storage element capacities of sites that have Storage Elements are 100 GB or 50 GB, and 100 jobs are applied to the system. The number of file access per job is 10. The total size of the data files is 50 GB. Available bandwidth between sites within the regions is 1000 Mb/s, and bandwidth between regions (between region servers) is 50 Mb/s.

Table 1. Simulation parameters.

Parameter	Value
Total number of the sites	**20**
T, T_{query}	**2**
I	**-/25**
D	**1.25**
T_{min}	**2**
T_{max}	**4**
Each Site Storage capacity (GB)	**100, 50**
Number of the jobs	**100**
Number of job types	**5**
size of each file (GB)	**1**
the total size of the files (GB)	**50**
Bandwidth between regions (Mbps)	**500**
bandwidth within the region (Mbps)	**1000**

As shown in Figure 2, the proposed method demonstrates better results in term of average job execution time rather than the optimistic and pessimistic approaches. The reason for this enhancement is the impact of decreasing network latency in the proposed method, which leads to a reduction in the average execution time of jobs. The number of updates is used as the criterion to compare methods together in terms of the network cost. In fig 3, The size of data files is identical. Fewer numbers of updates indicate the effectiveness of transformation and avoiding unnecessary transformation and updates.

As Figure 3 shows, the proposed method significantly reduces the number of unnecessary updates compared to the pessimistic approach; however, the optimistic approach is still best in terms of the number of updates. Therefore, the method proposed in this paper, compared with the optimistic and pessimistic approach, leads to a reduction of access latency and a reduction in network costs compared with the pessimistic approach.

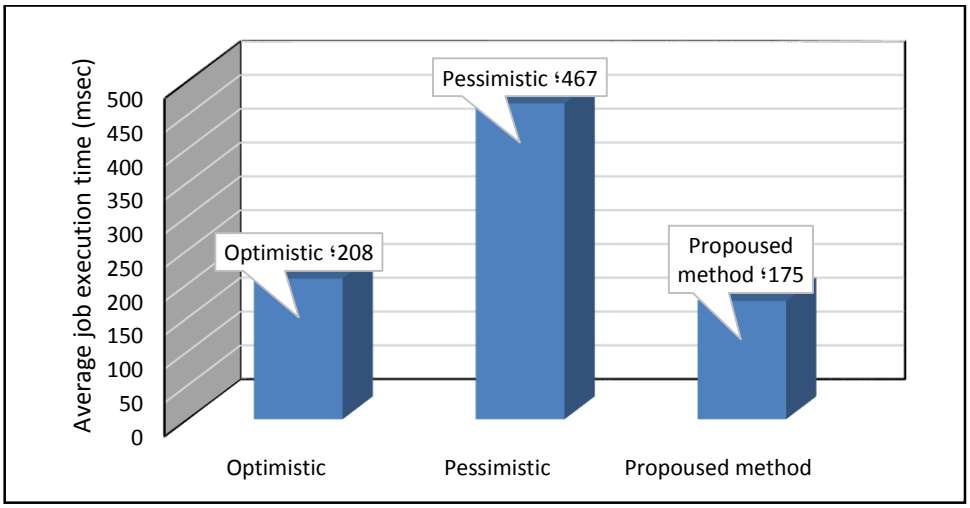

Figure 2. The average job execution time for various replica consistency methods job

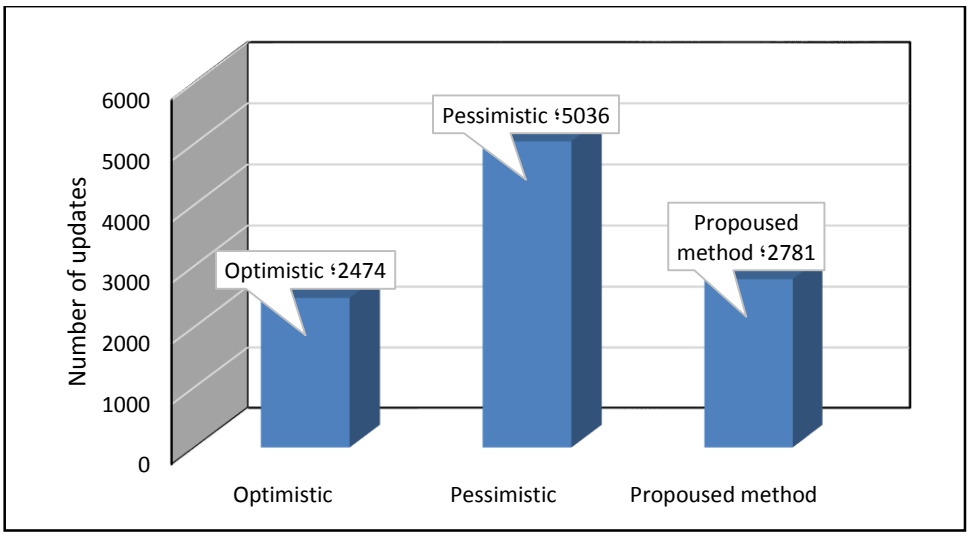

Figure 3. number of updates for various replica consistency methods.

5. Conclusions and Future Works

So far, Different methods have been proposed to maintain replica consistent in the Data Grids, and each of them has its own advantages and disadvantages. In this paper, we present a method that each node calculates the time of update by taking into account

parameters such as the rate of changes in the original data files and request accesses to replicas. These calculations are dynamic, and any changes in each of those two factors are considered. In other words, the node itself is responsible for maintaining its own replica consistency, and this means that each node updates its own replica considering the conditions of the original data file and its own requirements. So unnecessary updates are avoided. The proposed algorithm is implemented and evaluated in OptorSim, a Data Grid simulator developed by the EU Data Grid Project. Simulation results demonstrate that this method reduces access latency and network costs compared with the pessimistic approach. It should be noted that this method does not guarantee that all accesses will be reached to the updated replicas, but the number of accesses to the old replicas is few. So it is possible to neglect this disadvantage because our method results in an improvement in the reduction of latency access and efficiency of network resources. In the future, we plan to evaluate our algorithm in terms of more evaluation parameters in real Data Grid.

References

[1] Foster, "The grid: a new infrastructure for 21st", Physics Today,pp. 42–47, 2002.

[2] Rahman Rashedur M., "A Dynamic Replica Placement Strategy in Grid Environment", 2005.

[3] Foster, C. Kesselman, S. Tuecke, "The anatomy of the grid: enabling scalable virtual organizations", International Journal of Supercomputer Applications, pp. 200–222, 2001.

[4] Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke, "The Data Grid: towards an architecture for the distributed management and analysis of large scientific datasets", Journal of Network and Computer Applications, pp. 187–200, 2000

[5] Hasenburg, J., Grambow, M., & Bermbach, D. (2019). FBase: A Replication Service for Data-Intensive Fog Applications. arXiv preprint arXiv:1912.03107.

[6] Beigrezaei M, Haghighat AT, Meybodi MR, Runiassy M. PPRA: A new pre-fetching and prediction based replication algorithm in data grid. In: IEEE. ; 2016: 257–262..

[7] Beigrezaei M, Haghighat AT, Kanan HR. A new fuzzy based dynamic data replication algorithm in data grids. In: IEEE. ; 2013: 1–5..

[8] Beigrezaei, M., Toroghi Haghighat, A. and Leili Mirtaheri, S., Minimizing data access latency in data grids by neighborhood-based data replication and job scheduling. International Journal of Communication Systems,2020, p.e4552.

[9] Aldin, H. N. S., Deldari, H., Moattar, M. H., & Ghods, M. R. (2019). Consistency models in distributed systems: A survey on definitions, disciplines, challenges and applications. arXiv preprint arXiv:1902.03305

[10] Foster I., Kesselman C., The Grid: Blueprint for a New Computing Infrastructure,2nd Edition, Morgan Kaufmann, 1998.

[11] Brutschy, L., Dimitrov, D., Müller, P., & Vechev, M. (2018). Static Serializability Analysis for Causal Consistency (extended version). ETH Zurich.

[12] Hsu, T. Y., Kshemkalyani, A. D., & Shen, M. (2018). Causal consistency algorithms for partially replicated and fully replicated systems. Future Generation Computer Systems, 86, 1118-1133.

[13] Maozhen L, Mark B., *The Grid Core Technologies*, Wiley Publishing, 1005.

[14] Yuzhong Sun, Zhiwei Xu, "*Grid Replication Coherence Protocol*", The 22th International Parallel and Distributed Processing Symposium, Santa Fe, USA, pp.131-132, April 1004.

[15] Saito Y., Shapiro M., "*Optimistic replication"*, ACM Computing Survey, 33(2):41-22, 1005.

[16] Saito Y, Levy H.,"*Optimistic replication for internet data Services"*, In Proc. of the 24th Int. Conf. on Distributed Computing, London: Springer-Verlag, October 1000, pp.123-324.

[17] Düllmann D., Hoschek W., Jaen-Martinez J., et al., *"models for Replica Synchronisation and Consistency in a Data Grid"*, Proc. Of the 20th IEEE International Symposium on High Performance Distributed computing. IEEE Computer Society, pp. 63-35, San Francisco, 1002.

[18] Chang R.S., Chang J.S., *"Adaptable replica consistency service for Data Grids"*, Proc. Of the 3thv International Conference on Information Technology: New Generations (ITNG), 1006.

[19] Chang, J.S. and R.S. Chang. "*An Innovative Replica Consistency Model in Data Grids,* Parallel and Distributed Processing with Applications, ISPA, 1002.

[20] Yang Chao-tung, Tsai Wen-Chi, Chen Tsui-Ting, Hsu Ching-Hsien, "*A One-way File replica Consistency Model in Data Grids*", IEEE Asia-Pacific Services ComutingConfrence, pp. 364-332, 1003.

[21] Chang, J.S. and R.S. Chang. "*An Innovative Replica Consistency Model in Data Grids,* Parallel and Distributed Processing with Applications, ISPA, 1002.

[22] Ren X., Ru-chuan W., Qiang K., "*Efficient Model for Replica Consistency Maintenance in Data Grids"*, International Symposium on Computer Science and its Applications, pp. 252-261, 1002.

[23] OptorSim – A Replica Optimiser Simulation, http://grid-data-management.web.cern.ch/grid-data-management/optimization/optor.

[24] Bell W.H., Cameron D.G., Capozza L., Millar A.P., Stockinger K., Zini F., "Simulation of Dynamic Grid Replication Strategies in OptorSim", International Journal of High performance Computing Applications, pp. 46-57, 2002.

[25] Cameron D.G., Millar A.P., Nicholson C., Schiaffino R., Zini F., Stockinger K., "OptorSim: a simulation tool for scheduling and replica optimization in Data Grids", In proceedings of Computing in High Energy and Nuclear Physics (CHEP), 2004.

[26] Xun-yi R., Ru-chuan W., Qiang K., "Using optorsim to efficiently simulate replica placement strategies", The journal of China Universities of Posts and Telecommunications, 17(1), pp. 111-119, February 2010.

[27] Cameron D.G., Schiaffino R., Ferguson J., Millar A.P., Nicholson C., Stockinger K., Zini F., "OptorSim v2.1 Installation and User Guide", pp. 1-21, October 2006.