Performance Metrics for Multiagent Systems

Aslam Muhammad[1], M. Junaid Arshad[2], Amjad Farooq[3], Rubab Fatima[4], Khair-un-Nisa[5]

[1,2,3,4,5]Department of CS & E, U. E. T. Lahore, Pakistan
[1,2,3]{maslam, junaidarshad, amjadfarooq}@uet.edu.pk, [4]rubabraza@gmail.com, [5]khairu_nisa87@hotmail.com

Abstract: Multi agent systems (MAS) bring about a change in the globe by making agents work together in a group achieving common goals and casting away monolithic paradigm. Proper understanding of the metrics that impact performance of MAS can help in employing the distinctive abilities of agents to its maximum. In this paper, we discuss various performance metrics that target an agents' role and environment and can help in accomplishing goals in optimum time. We also present an example that takes these performance metrics to account. We then move to case studies and draw attention to the best and worst cases for agents' performance against the metrics we gathered .[Aslam Muhammad, M. Junaid Arshad, Amjad Farooq, Rubab Fatima, Khair-un-Nisa. **Performance Metrics for Multiagent Systems.**Journal of American Science 2011;7(7):804-810].(ISSN: 1545-1003). http://www.americanscience.org.

Keywords: Multiagents, Metrics, Performance, Response time.

1. Introduction

Panoptic research in artificial intelligence has shifted the mode of independent study of fields to interdisciplinary studies like mixture of cognition, sociology, psychology, robotics, fuzzy logic, computation and several other interesting subjects. This results in better understanding and application of interdisciplinary theories. Multi agents system is a product of such rigorous studies. The shining hope in this arena is brought about by the fact that agents are autonomous in their nature. Since they encompass the ability to be decisive for attaining a goal, performance should not be a hindrance for their task accomplishment.

In the past apt attention was not given to defining metrics alone for multiagents. Either frameworks were suggested for a specific type of agent or environment or performance was neglected altogether. We introduce some performance metrics based on our study and knowledge that should be catered while construction of MAS.

The objective of this paper is to spot various performance metrics in accordance to the possible roles of an agent. Agent role is categorized based on its environment and functionality expected from it. An example of RPM is considered that takes the mentioned performance metrics to account. It shows how a system can be made efficient if performance metrics are considered properly. This paper will provide benefit to designers for designing a MAS system for any environment. They can determine which metric can be best applied for maximum efficiency.

First we will give you a walk through the related work done on this topic. We then define imperative performance metrics that we think are important to consider while designing a MAS. We then shift our focus to elaborating the performance metrics by means of an example that fulfils them properly. In the last we discuss some case studies for best, medium and worst case scenarios for the defined metrics and conclude our results.

2. Related Work

A metric is a quantitative measure of the degree to which a system possesses a given attribute. A performance metric is thought to be a function of the optimization of tasks an agent conducts to reach a goal. It assists an agent to be efficient and effective on time. Performance metrics are based on measurable attributes of an agent [1]. Since only measurable quantities can be controlled and directed, we can use them for our purpose to enhance performance and reliability. Agents also have the clairvoyant ability to improve performance over time by learning through examples and experiences [2].

A metric allows finding a pattern and trend in an object's behavior. In [12], various performance metrics are defined but they focus on single agents alone. The combined effect is not discussed. It also proposes different profiles by combining a set of single agent metrics to fulfill the need of a specific type of agent. This approach provides agility but not basis for a MAS system.

Various factors and variables determine performance metrics for meeting the performance requirements of MAS. Researchers have concentrated on domain specific metrics as discussed. These include time period for task completion, efficient interaction between agents, centralized or decentralized processing, how fast a MAS reacts when input is provided, relative performance of an agent with respect to its environment or other agents, the number of agents a system can bear, the number of tasks completed by agents etc [3][4]. These researchers have discussed only those metrics that concern or affect the particular system they have taken up as an example. What they do not define is the impact of these performance metrics in other dynamic environments. They also do not indicate which type of agent is affected by the performance metrics they are defining.

While designing MAS it is crucial to identify points where performance delays may occur and whether or not it is tolerable for the accomplishment for a global goal. Numerous conformance tests and performance evaluations may be conducted to verify these summits [5]. We present a handful of performance metrics that can be applied to any MAS system for evaluating its performance in advance.

3 Imperative Performance Metrics

The pattern followed for bringing to light distinct performance metrics in this paper is by discussing various factors that increase usability of agents in an assortment of scenarios. We then pin point which metric can be catered best for boosting performance of an agent in a particular environment or for a particular agent role.

A) Metrics for Single Agent

a. Number of Agents

The primary indicator for the size of MAS is the number of agents involved. This factor is directly proportional to the complexity of the overall MAS. The more complex a MAS, the more performance enhancement is required by it. It should be made sure that the performance enhancement does not create an extra over head decreasing the performance of overall MAS [`2].

In a huge MAS grouping of agents is usually done to suppress the overhead involved in communicating among groups of agents and within groups. By and large, the performance of communicating within a group is immensely increased.

b. Computational Time for Individual Agent MAS

An active state of an agent determines its computational time. The time period for which an agent conducts its tasks is its active state. In a MAS we need to know how many agents are active in what period of time. It is also calculated that how many agents have an overlapping active state time period. The number of overlapping computational time affects the performance of system. The higher it is, the lower the performance goes.

c. Independent Memory Consumed by Individual Agent MAS

An agent may require several resources for executing its tasks. The amount of memory required by an agent for creating its own objects should be known in advance to predict the performance of MAS. Memory is an inevitable constraint for many of the MAS. The more memory consumed by an agent, the low the performance of the MAS we get [12].

d. Agent Status

The total number of statuses an agent can have determines its status set. If the set is huge, it implies that an agent would go through transition at least once for all statuses. The transitions required would be a pure overhead for the system. The less the transition is required, the more performance we can get from MAS. The statuses an agent might commonly require are Active, Wait etc. The number of switches between these statuses must be less frequent to increase performance [12].

B) Metrics for Multiagent System

a. Agent Coordination in MAS

Message transport and communication protocols utilized directly determine the performance of MAS. These levels comprise of low level to abstract high level protocols like speech act theories for message content. A number of platforms are provided to support such protocols as Java Agent Development Environment (JADE) and Foundation for Intelligent Physical Agents Operating System (FIPA-OS). Coordination protocols employed can be either hierarchical or mesh structured.

In a homogeneous MAS a complex task is broken down and then delegated to agents. So this factor is more of an importance in such an environment. This is different for agents related to Mesh and hierarchical structures [12].

b. Fault Tolerance

Fault tolerance is one way to increase the dependability of agents or applications. It provides the recovery from service failure when a fault occurs [6]. Many solutions have been proposed for introducing fault tolerance in multi-agent systems, some are of curative nature while others are of preventive nature [7]. Fault tolerance can be achieved by cloning [8] or replication. Cloning has several disadvantages, like it cannot be applied to every multi-agents system, and agents are supposed to be stateless. Using replication, shortest recovery time is achieved and it scales much better.

This metric is particularly crucial in distributed environments, where failure of one system can cause failure or delay of many other systems.

c. Connection Metrics

While studying '*Structural performance evaluation of Multi- Agents systems*', Dariusz and Zelmozer introduced connection metrics to evaluate the Distributed Object Systems. Since, it is more difficult to evaluate performance of distributed applications because of various environments, architectures and implementations used in distributed object systems, so, in a real-time project, it is not possible to first try different architectures and designs and then decide the one with best performance. Therefore they introduced *connection metrics* as the cheapest analytical tools to decide the best design and implementation.

In this study, the most suitable metric they found is *Connection Cost Metric.* It estimates average distance between agents. This metric best measures the performance of distributed multi agent systems. Best and worst scenarios are discussed for this metric. These proposed metrics can be applied efficiently by using service oriented architecture (SOA).

d. Stability

One chief performance metric is the '*Stability'* of a Multi Agent System. It is hard to set a definition of stability in case of multi-agent system; it must be somewhat more concrete than the one used by software engineers, and a bit more flexible than that defined by control engineers. According to, a system is considered to be Stable if its state is converged to an equilibrium distribution. Hence, stability becomes a quantifiable metric to measure the performance of a multi-agent system. This metric is specifically applied to categorize multi-agents used in games, especially those that are close to real world. For that purpose, the experiments are performed on '*Agents Ecosystem'* in which agents are not fixed over a span of time. Agents can appear and disappear at anytime. This metric has also been tested and verified on simulation models that closely resemble the real-time systems.

e. Communication

Since MAS [9] is a system based on multiple agents that are interacting with each other and they can solve multiple problems that are difficult or even impossible to be solved by individual agents, so *communication* among the agents is a key factor of performance measure. Communication metrics include average message length, average number of messages to and from individual agents and average message communication delay [12].

There are a number of metrics proposed for designing the better communication among various agents. These metrics mainly targets the load balancing, for example, to check if a single agent is requested again and again, it will overload that agent, so based on the metrics, requests will be routed to some other agent who has less load.

The agents are divided into 5 categories depending upon their type of communication and load balancing.

i. *Over loaders*: these are the agents that send too many messages and overload the system
ii. *Overloaded*: these are the agents that receive too many messages and get overloaded.
iii. *Isolated*: these are the stand alone agents that neither send messages nor receive them.
iv. *Overloaded-Over loader*: these are the agents that are overloaded but they overload others too.
v. *Regular*: these are the ideal agents that exist in a balanced system. They send and receive a balanced amount of messages.

The metrics are derived to avoid the first four kinds of agents. Some of the metrics defined by are as follows:

a) Overloader system metric (BS): it measures the amount of sent messages by the agent as compared to total messages sent by the system.
b) Overloaded system metric (MS): it measures the amount of messages received by the agent as compared to total messages received by the system.
c) OverloaderRole Metric (BR): it measures the amount of messages sent by an agent as compared to amount of messages sent by agents playing the same role.
d) OverloadedRole Metric (MR): it measures the amount of messages received by an agent as compared to number of messages received by the agents playing the same role.

e. Agent Management

Agent management deals with keeping track of how many agents are in active state, how many are yet waiting to be invoked, which agents has accomplished its task, which is still executing, which agent require resources and which would be requiring in the future etc [12].

f. Dependencies between Agents

The mode of communication between agents may determine the degree of dependency an agent has on another agent. An agent may be operating in a synchronous mode with respect to another agent or in an asynchronous mode.

In a synchronous fashion an agent may be dependent on another agents' data when the other agent has completed execution. So the agent first waits for another agent to complete execution first and then continue with its own execution process.

In an asynchronous manner, an agent may use various algorithms to keep data synchronized among all agents. The protocols they might be following would comprise of WAW (Write after Write) or RAW (Read after Write) or WAR (Write after Read). The dependencies among agents can be computed by drawing dependency graphs.

g. The Optimization of Shared Computer Resource Usage

In a MAS environment, it might be very common for agents to have shared memory for communication of information. It is of prime importance that any action performed by an agent does not result in to memory outage. There may be various factors that will help to ensure that free memory is maintained and unnecessary data is removed from the system. There are two types of data in this regard. One comprises of end results that are useful for another agent at some time during the execution of its tasks. The other is used as helping data to reach an end result. Some tips for managing memory that are in practice may include:

- Gathering information as to when what data might be required by an agent.
- Making sure which data is no longer required after a particular agent has utilized it.
- Intermediate results are discarded as soon as they are utilized and no longer required.
- Avoiding dead lock states on shared resources.
- Synchronizing the availability of resources efficiently.
- Number of concurrent processes that will execute in a system must be known in advance to identify and then handle potential bottlenecks in the system.
- Dispatcher should be able to accommodate context switch gracefully without loss of time.

Since an agent is allowed to operate in limited memory, the careful analysis of which data is required at what time in which place and whether it is no longer required, before implementation of MAS, will lead to a chance to optimize the task execution of an agent. Memory leakages if not handled correctly might result in catastrophic performance on an agents' part.

While handing out resources to agents, the system should be sure that it is the proper time and the appropriate agent that actually needs the resource. It is of no use to an agent to confiscate a resource and then wait for another agent to provide it input for the utilization of the resource. Such states may either result in deadlocks or will force an agent to wait that will impact the performance profusely of the global goal of the system.

System itself should also minimize the response time when a resource is requested by an agent on urgent basis and make sure no other system resource or agent becomes a hindrance for its execution.

h. Ability of a system to manage the dynamics of the agent population size

A system should have the capability to administer several agents all together. The number of agents might be fixed before the execution of a process. During execution, some intermediary agents may appear. System should be well prepared to supervise these agents and to be hospitable enough to grant them resources whenever required.

When the agent population size increases, so does the demand of the system to be more responsive flourishes to grip the dynamics of all agents. Once a system is prepared to deal with the increase in population of agents, it should start providing room for storage of their result computations as well [9]. A good practice is to have an estimate to know till what extent memory requirements would exceed..

The implementation of increase in agent number can be facilitated by use of multithreading concept on software. However this approach should be employed only if one is well aware of its issues and is equipped of the expertise required to make the most of it.

i. Bandwidth and Latency

Communication efficiency in MAS is subject to improvement in overall system performance. In a fully networked environment it is observed that bandwidth and latency best define as performance metrics (Harchol, 2002). The network community lead proposes a study in which frequency of message exchanges between agents is shown to be directly proportional to the throughput of the system. The more recurrent the communication, the more frequent we get a response from the system.

A relationship between frequency, bandwidth and throughput is also discussed. It is deemed, through experimental evidence, that frequency of message passing cannot be more than the bandwidth of the system and hence throughput of the system gets an upper bound. It is quite notable that the frequency of message passing also tends to vault the maximum number of agents communicating at a time and the rate of exchange of information between them.

The latency is described as the time required for a message to be communicated from a source agent to a destination agent. This is a critical measure when agents have to operate in a real time environment. A small delay in latency may mean to be a difference between success and failure of entire MAS. The less the latency, the more effective a MAS can be. Latency can be highly dependent on the size of the message transmitted between agents. Longer messages are prone to high latency rates. Latency has been a subject of interest for network engineers because it is believed that it affects the quality of service of a network [10].

j. Load Balancing

When agents work in a large scale system, the overall performance is usually measured by calculating the performance of the locally active agents. On a huge portal like World Wide Web it is not easy to maintain global information available at all times. So people rely on local agents that become active and supply local information to the users.

Agents gather knowledge from partner agents locally when global information cannot be supplied instantly. For this purpose appropriate partner selection is important for leaning, which is a hectic task. Recent studies reveal that this factor can be overcome by load balancing across agents when workloads are high and concentration on high performance agents when workloads are less [11].

Load balancing may comprise of shifting data computation to some partner agents and dedicating others to gather information locally. Recent research also uncovers the fact that learning parameters for local strategies to select partner agents influences the total performance of MAS. Experiments prove that statistic values of known resources can make agents more adaptive. This has a drawback of declining performance in new environments. Since environments evolve around, relationship between speed of change and adaptation starts getting worse [11].

k. Performance of Cultural Evolution

Learning is a vital aspect of multi agent system. When agents work in a culture there is a set of agents that initiate the learning process and another set that conjure up the content conveyed. The more autonomous the agent, the more an agents' performance is increased. One of the most effective metric to increase the learning process is the addition of noise in the content communicated between agents [11].

Noise was added through cultural mutation. A neural network layer was used for communicating information. The receiver of the information imitates what data is communicated by the initiator. This imitation is based on back propagation of the received content. The receiver has a hint of what possibilities an initiator has to communicate. A receiver also has an idea to how many receivers an initiator can communicate with. These factors help while interpreting after back communication.

It is observed that overall fitness is increased by introducing noise. The ratio of noise to content communicated is set by keeping in view the number of initiators and the number of receivers for a specific content.

3 Case Studies

4.1 Framework for Natural Disaster Management

Our first case study is that of a framework introduced for interaction of MAS during natural disaster (FFNDM) [13]. It consists of Sensor agents for collecting data, a platform for collaboration between agents, a decision support system for deciding what action should the response agents take.

The framework considers some onsite agents and other remote agents. This gives a distributed touch to the application. The framework fulfills most of our performance metrics and hence is the best case overall taken by us. The only metric it lacks is the ability to learn from previous records and refine decision support system accordingly. We have highlighted the ratings of performance metrics for it in Table 1**.**

4.2 Evaluating Urban Traffic

The second case study is based on traffic modeling, evaluating urban traffic (EUT) and effects of unexpected events or uncertain factors [14]. Real time data for busiest intersections is used for this case study. The traffic is modeled first with the help of Bayesian Networks, and then causal networks are used to measure effective factors. Experiments have proved that this model is cheap and less time consuming as compared to other models in not only modeling but also predicting the future trends/patterns in the traffic.

Attention is given to percentage saturation value, it is the basic parameter for modeling. One drawback is that modeling is done assuming long time constraints only, so the sudden rush or anomalies in traffic for short time periods cannot be modeled.

Table 1: Comparison of case studies

Performance Metrics Elements	FMSIND	EUT	TSM
Number of Agents			X
Computational Time for Individual Agent MAS	X		
Independent Memory Consumed by Multiagents	X		
Agent Status	X		
Agent Coordination in MAS	X		
Fault Tolerance		X	
Connection Metrics		X	
Stability			X
Communication			X
Agent Management	X		
Dependencies between agents	X		
Optimization of shared computer resource usage	X		
Ability of system to manage the dynamics of agent population size	X		
Bandwidth and Latency	X		
Load Balancing	X		
Performance of cultural Evolution		X	

Stability is very well handled in this case study, since there's a special factor called 'traffic heavy'. The performance is evaluated on the basis of maximum and minimum traffic, and the case study has shown good results for all.

Another drawback is that not many factors could be considered in this case study for evaluation purposes, since the additional factors added to the complexity of the overall system, and thus degrading the performance.
A comprehensive list of where this case study lies in our performance metrics is given in table 1.

4.3 Trading Simulation Model

This case study specifically focuses on stability of multi agent systems. As a case study, a trading simulation model (TSM) is introduced, there are M traders with their discounting prices, their capital and available resources. This scenario requires tasks that are generated by agents. A task utilizes some resources, and creates some other ones, to be used by other agents. Each trader produces some tasks, and then advertises it among other agents. All the agents bid to perform that ask, and the bid with minimum demand is accept usually. Finally, each trader re-calculates its worth, by taking into account the discounting percentage, and if its offer was accepted or not. New agents are generated any old ones are destructed based upon requirements.

The stability condition is defined as the system reaches a stationary distribution. Stability would be if the system is left to execute for a while, and all the factors including number of traders (agents), wealth per trader etc reach to their stationary distribution.

The model can be run with initial constraints to check out the best and worst cases, especially with respect to stability.

7. Conclusion and Future Work

Performance metrics of MAS is dependent on the role an agent has to perform and the environment it works in. Performance metrics that impact MAS need to be considered in amalgamation with the metrics that impact an individual agents' performance. There are tradeoffs between performance metrics and MAS global traffic. Increasing size of MAS may require better management of agents resulting in low response time. The user needs to decide which division of a MAS can compromise on performance and which are mission critical.

The categorization of performance metrics have been done on the basis of the role of an agent. Performance metrics can be evaluated based on several other criteria and impact of those scenarios can be studied on real time MAS systems. This could assist in analyzing performance metrics from diverse angles.

References

1. Russel, S. and Norvik P., “Artificial Intelligence: A Modern Approach,” Edition. 3.
2. Linda Rosen, 1993. "MIT Media Lab presents the interface agents symposium: Intelligent agents in your computer?" Information Today, vol. 10, no. 3, p. 10.
3. Goldberg D. and Mataric M.J. Interference as a Tool for Designing and Evaluating Multi-Robot Controllers, In Proceedings of 4'h AAAI Conference in Artijicial Intelligence.1997.
4. Tumer PJ. and Jennings N.R. 2000. Improving the scalability of Multi-agent System. In Proceedings of 4'h International Conference on Autonomous Agents. 2000.
5. R. Ewald, D Chen, B. Logan, 2006. Performance Analysis of Shared Data Access Algorithms for Distributed Simulation of Multi-Agent Systems
6. Algirdas Avi Zienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–23, January–March 2004.
7. Christophe Dony, Jørgen Lindskov Knudsen, Alexander B. Romanovsky, and Anand Tripathi, editors. *Advances Topics in Exception Handling Techniques*, volume 4119 of *Lecture Notes in Computer Science*. Springer, 2006.
8. K. Decker, K. Sycara, and M. Williamson. Cloning for intelligent adaptive information agents. In *ATAL’97*, LNAI, pages 63–75. Springer Verlag, 1997.
9. E. Piquet, R. Coudier Enhancing a multi agent systems’ performance M. Wooldridge, *An Introduction to Multi Agent Systems*, John Wiley & Sons Ltd., 2002.
10. N. Cardwell, S. Savage, and T. Anderson, .Modeling tcp latency, In Proceedings of IEEE INFOCOM, March 2000.
11. T. Sugawara, S. Kurihara, T. Hirotsu, K. Fukuda, S. Sato, and O. Akashi. Total Performance by Local Agent Selection Strategies in Multi-Agent Systems. In *Proc. of AAMAS200,* May 2006.
12. N.K. Nagwani: Performance measurement analysis for Multi Agent System 978-1-4244-4711-4/09/$25.00 ©2009 IEEE.
13. Aslam Muhammad, Tariq Pervez Muhammad, Mushtaq Seemal, S. Shah Muhammad, Martinez Enriquez A. M., “FMSIND: A Framework of Multi-Agent Systems Interaction during Natural Disaster”, In. Journal of American Science, Published by Marsland Press, Lansing, USA, MI 48909, ISSN: 1545-1003 vol. 6(5), pp. 217-224, May, 2010.
14. Reyhaneh maarefdoust, Saeid Rahati “Traffic Modeling with Multi Agent Bayesian and Causal Networks and Performance Prediction for Changed Setting System”. Published in 2010 Second International Conference on Machine Learning and Computing.

6/12/2011