Iraqi Journal of Statistical Sciences

http://stats.uomosul.edu.iq

Message Passing Applications: A Review

Halemaa E. Solayman and Ashraf A. Abdulmajeed G. M. Aldabagh

Department of Computer Science, College of Computer Science and Mathematic, University of Mosul, Mosul, Iraq

Article information

Article history:
Received March 4, 2021
Accepted April 25, 2021
Available online December 1, 2021

Keywords: Message Passing Shared Memory process synchronous Asynchronous network protocol.

Correspondence:
H. E. Solayman
haleema_essa@uomosul.edu.iq

Abstract

It is known that message passing has become one of the most popular parallel programming paradigms because of its ease of use, so it was necessary to know or study the applications that were adopt message passing in their work.

Programming models are for the most part classified by how memory is utilized. In the shared memory model, each cycle gets to a shared location space, yet in the message passing model, an application runs as an assortment of self-ruling cycles, each with its own local memory. The principle preferences of setting up a message-passing standard are convey ability and convenience. In a circulated memory correspondence climate in which the more significant level schedules as well as reflections are based upon lower level message-passing schedules the benefits of normalization are especially evident. Moreover, the usage of a message passing, for example, that proposed here, gives sellers a plainly dined base arrangement of schedules that they can actualize in days of yore, or at times for which they can give equipment uphold, consequently improving adaptability.

DOI: 10.33899/IQJOSS.2021.169963

1. INTRODUCTION

Since 1994, computer manufacturers and many academic and government laboratories have developed standard specifications for interfaces using a library of message-passing procedures. This standard is known as MPI 1.0 (Message Scroll Interface Standard). After this initial release, several releases were produced, including: 1.1 (June 1995), 1.2 (July 1997), 1.3 (May 2008), 2.0 (July 1997), 2.1 (July 2008) and 2.2 (September 2009). Versions 1.1 and 1.2 work to correct errors in MPI 1.0. MPI-2 (MPI-2: add extensions to the message scrolling interface, add new functionality to MPI 1.2. These standards can all be obtained in HTML format [1]. The calling program delivers a message based on the object to determine which commands are acceptable. Message-passing architectures are clear systems, each unit will be dependent on each other, in addition to the use of a regular mechanism to pass data through system units [2]. As a form of passing messages, method calls can be considered one of them, but this is not considered practical, because it is considered a problem. The reason is that if we have a specialized method for scanning with a category, and another section calls for these methods, then a compilation of these sections must be done. A message-passing system often allows objects to be attached at run time, and does not allow messages to be routed to one or more objects frequently [3]. As a result, if there is some code that sends an "updated x data" message to all loaded objects, so that every action can take action with this information [4]. In this article, a detailed description about message passing and message forwarding applications is explained. Finally, a comprehensive summary is provided.

2. Related Works

An overview of the most important work related to this article review, that deal with Message Passing applications, which will be listed in the following lines. The author in [5] present some of message passing application examples, such as imaging applications like Radar and Medical applications, but the work focus on mathematical concepts only. In [6] and [7] frameworks are presented using algorithm for automatic generation selection, to select the message passing system components used in message passing applications. The work in [8] presents a paradigm to track many tools based on message passing, which has many features like it provides an efficient and improvable solution for probabilistic data attachment, which considered an issue in multiple tools field, the system use data captured by two radar stations for monitoring multiple goals from time to time, the work focused on non-linear paradigms only. The research in [9] use a data

structure based on graph model, this pattern is used for gaining adaptability with Neural Network, based on dynamic undirected graphic data which represent data correlation issue in different time intervals, the Neural Network for this structured graphic based on message passing gain the desired probability for each connection in this graph, but the work use binary box location to initiate descriptor for each object instance to adapt with Neural Network.

3- Message Passing

In schemes that based on an object communication such as CORBA[10] [11] and RMI [12] available in the Java [13][14]environment, messages that passed should be resorted, a restrictive and plain communication method, is needed to transfer specific information through entities. During passing the message is robust and simple like an objects distribution process, it's comparatively easy to perform by adapting the package (java.io). The two techniques have very varied and different goals[15]. Distributed objects extend the program over the network through modeling its objects that look to be prevalence across the virtual machine hosts. The Message Passing work look a much simpler task, using an undeveloped networking protocol to transmit the data. [16]. Using I/O streams for Passing messages avoids the overhead of communication happened when using most essential distributed schemes of an object, without requiring any special protocol of networks. For this reasons message passing is a beneficial tool, particularly for these situations as flowing:

- The required Communication are relatively and naturally not complicated
- The throughput of transaction is serious
- The restriction of the domain of the system, so that the speedy achievement outweighs the influence on the design and its modification
- Avoiding special protocols of the network (e.g., the required part which worked behind a firewall)
- The protocols of the remote object (e.g., a small browser program that does not support CORBA or RMI) that cannot be accessed.
- During agent development that is responsible for messages swapping, the estimation of message integration processing is important with the remainder of the objects that is responsible for constructing the agent [17] [18]. In principle, it is suitable to do the following:
- Separate the details of communications from those of application.
- This gives freedom to perform the bulk of the app-building categories about app issues, not app issues related to the connection system happened and using. Also, the subsystem of communications is updated and implemented separately, based on the connections required for each system.
- Supply association structured based way to connect messages with method calls on objects of an application.

A cleared way is required to access messages to run an object method calls for an application, and for object methods to create messages to remote objects of the requested service.

These may seem like inconsistent requirements, but they may be established to one degree or else when using a single method message-processing [19].

1. Asynchronous vs. Synchronous Message Handling

a. Synchronous message passing

Systems based on synchronous message passing require that the sender and receiver wait for one to another while message transferring [19] [20]. In asynchronous communication message passing the receiver and sender will not wait each other and run their own calculations during message transferring unlit message transferring complete [14]. So, whether the message passing systems uses synchronous or asynchronous message passing this will be the most important differences between message passing systems. Synchronous based message passing happens between the simultaneously running objects. In asynchronous based message passing, it is possible that the receiving object to be in the operation mode or not when the requested object is send the transmitting message [21] [22]. Usually, synchronous based message passing may be considered as an (OOP) Object-Oriented Programming languages such as Smalltalk and Java. While Asynchronous based message passing demands extra capacity to save and forward data to systems that may not be executed at the same time. The main characteristic of synchronous based message passing is that it's less complex as compared with asynchronous based message passing, Synchronous based message passing is similar to a function call that the sender of the message is the caller to a function and the message receiver is considered as the function called of the message. function call may be looks familiar and it is not complicated. Even if the caller of the function is not run until the termination of the called function, the process of sending stops working until the receiving process terminates. Who manufactures a synchronous message does not work for some types of applications. As an example, when synchronous based message passing is used in large scale and distributed large systems, this type may not usable and cannot perform good enough in these types of systems. For these cases which need to operating continuously despite some of their subsystems cease; subsystems may not need to work for maintenance of several kinds, or while they are closed (not open) for receiving inputs that are come from other different systems" [23]. If there is a busy business office with one hindered desktop computers send emails messages one to another using synchronous based message passing for satisfying communications. So long as the office does not depend on using asynchronous based message passing, then when any worker shuts down his destop_computers this will causing that the other 99 destop_computers to freeze or pause until this individual person turns on his computer to process (receive) an email messages [24].

b. Asynchronous message passing

In asynchronous message passing, the sender will not wait for a response from the other system. waiting the function call analogy, asynchronous message passing will be a function call returning at the same time, without suspension for the desired function to run. This function call will only move the parameters, to the specific function being called, report that function to run, then back again to continue running it. Asynchronous message passing delivers the message to message bus in a simple way. The message bus holds the message until the recipient needs these messages and sends them to him. When the receiving operation reaches the result, it advances the required result to the message bus, then the message bus saves the message until the sender operation takes the required messages from the message bus [25]. An important question in building a message processing system is whether it requires it to be synchronous or asynchronous. For example, the agents player is able to simultaneously process messages, and he is certain that he will communicate through the game all the time. One player transfers a move (using a message) to another player; the second player matches the move to the copy of the "playing board", measures its options, and transfers his "countermove" to the first player. In this example, there is nothing the player has to do except choose his move or wait for the second player to send his move, so there is no need to be able to synchronously receive and process messages[26]. This is not usually the case. Ideally, the messages create some important processing by the agent, which can be performed while waiting for any other messages. Application agents will be overall better off when they can send and receive messages in an asynchronous manner than any other action required. This "other business" might be to respond to these messages, or to act independently of passing on the message that will continue. "When implementing a network game more complex than this simple chess system, each player agent may have a lot of work to do in addition to sending and receiving messages." There may be an entrance to work with, many players for remote syncing, graphical screens may be required to update, and complex internal modules to keep the route right. To save every item in the proxy operating in the ideal way, asynchronous message I / O may be necessary so message-passing scheme will support this method[26][27].

2. Comparison Between Message Passing and Calling

Asynchronous and distributed message Passing has little burden correlated with it to be structured in an easier way just by a procedure call [26]. In conventional procedure call, arguments are typically passed to the recipient through one or more what are known as common purpose registers or may be passed as a list of parameters including the arguments addresses. This approach of communication varies from passing based messages in the following important criteria's:

- memory usage.
- locality.
- the amount of time required for data transfer.

Message passing include, moving every process argument, involves copying each argument to a part of the new message in its creation state. This applies to the argument size, and in some scenarios the arguments size may have data with megabytes. They must be completely duplicated and carried over to the object that will receive it. In contrast, in a procedure call, only a few bits which represent an address of an arguments must be passed to a common purpose registry that does not require additional storage space and zero time for transferring. This is not suitable for systems that are distributed into large scale, because the address space of the caller which contain the absolute address is commonly not understood by the remote program (the absolute address may really be used if the recipient has in advance the same copy of the memory of the sender). There are many examples of processes that interact by messages based passing, for examples: Web servers and web browsers. URLs are the samples of a resource referencing methods that is not based on revealing the internal process elements. The calling method or sub procedure call don't end until the named account is terminated. Message passing of type Asynchronous, by way of opposition, may trigger a reaction after sending of the requested message [28]. The handler of the message generally processes messages received from set of transmitters, indicating that the messages states can be changed according to the reasons that are unrelated to the single transmitter or client behavior. Which is differs from the object common behavior to which methods are called: calling methods are expected to keep the same state between processes that adopt method calls. So, the message handler interacts with an unsteady object. [26] [29].

3. Message passing applications

It is necessary to submit a research or study to cover all current studies, books and other resources related to the topic of the research presented. There are many applications related to this topic that motivated writing a brief summary of the most important of these methods and applications. In this article, a number of applications will be discussed. The presented research will help and provide the reader with important background information on the topic, the most important applications on this topic will be summarized in the following sub-paragraphs [30].

a. Communication between agent

Agents existing in the same environment can interact via sending messages to each other. When a developing agent does exchange messages, it is imperative to think about how message handling can be combined with the other objects

components that included in the agent. The steps for following are: Isolate the details of the connections from the details of an app, this gives freedom to collectively struct the categories that make the app around the determined application issues, not that associated with the system of communication used. And identically, the communication subsystem can be independently updated and designed, relied on the requirements of a communication of the entire system [28]. The structured method must be provided for linking messages with a method calls on the objects of an application. The distinct method is needed for receiving messages to run method calls on objects of an application, also for object methods to construct messages required by the remote agents for specific service requests [29]. These may like to be incompatible requirements; both can be met to one degree or another in a single message processing method [30].

b. Bank account example

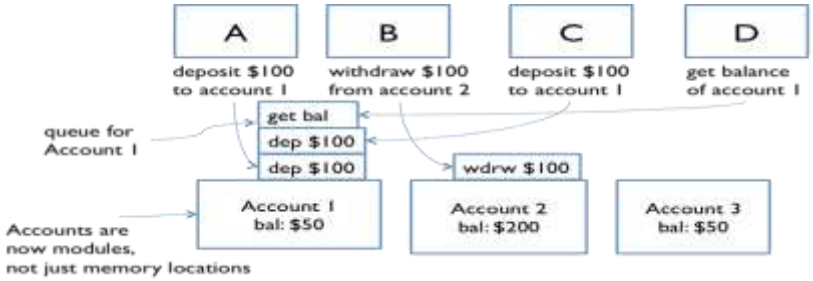

Figure 1. Bank Account Operations

Each account and every cash machine has its own unit, and the modules communicate by sending messages to each other. The incoming messages arrive on a queue. Messages are designed to obtain balance and draw, and each cash machine inspects the balance of the account before drawing to prevent an overdraft:

get-balance
if balance >= 1 then withdraw 1

But the messages could still be interleaved from two cash devices, and thus be fooled into believing that they could massively pull the last dollar out of an account with only $ 1. A better atomic process should be chosen: withdrawing funds if sufficient, then this is a better process than mere withdrawing [30] [31].

c. Network protocol

For dissipated machines, the only way one machine talks to another is by passing messages over the network [32].
Each network protocol and standard can be reduced to a form of message based passing. SSL, HTTP, low-level network protocols such as TCP / IP are protocols that are built around a form of message based passing. The message that is explained here is passed explicitly by the application programmer. While other protocols are organized around a type of message-passing protocol, this protocol level is unobserved from the developer by an API of some kind. As example, SSL is used by an application programmer by a class library, as method calls on SSL-related objects are automatically split into SSL-submissive messages. Likewise, incoming SSL Messages are assigned and processed in new method calls and data objects on SSL objects. This is what makes these complex yet powerful and useful protocols: the application programmer does not need familiarity with the details of the protocol at the lower level [33].
The web is the strange example. HTTP is a message passing system, and the command verb and "data packet" are passed to the server process [21]. HTTP is a synchronous remote procedure call. It is the appropriate version of the Syn protocol [34]. It usually has the same timeline diagram with the Syn protocol, and neither the user's browser nor the web server has anything to do with the data sent or where it was sent[35]. The server will place it on the code that will collect another "packet" of data and send it back to the user's browser. The component in this system does not know anything about the work of others or what they are doing, they only know the protocol used for communicating messages [32][36].The basis for all types of distributed account models is made using three types of protocols [33].

d. Message Handling Flow

Network communication takes the form of demand and response communications. The program that plays save requests is called a server. Symmetrically, the program that redirects requests to the server is called the client. The server or client is used to denote the role played by the program. The program can work as a client and server at the same time [37].

Synchronous Protocols

The synchronous protocol is associated with a normal synchronous function call, except that the call takes place at the boundary of the device [38]. It is commonly used to implement a concurrent server-side function and wait for the server-side to respond in a blocking fashion as shown in the time sequence diagram shown below.

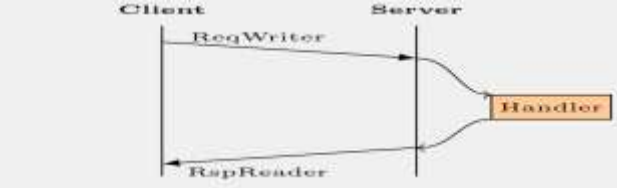

Figure 2. Synchronous Protocol

Asynchronous Protocols

In asynchronous protocols, the server responds with an acknowledgment of receipt to the client as soon as possible after receiving the message. Then a thread is chosen from the thread pool to handle the received message as shown in the following sequence diagram [39].

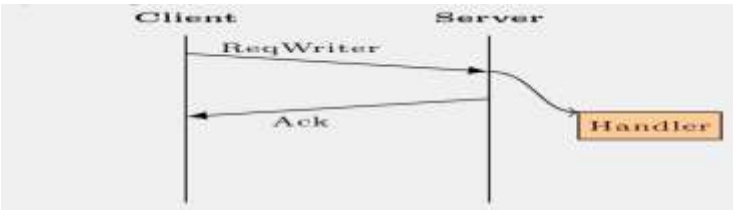

Figure 3. Synchronous Protocol

e. Message passing in multi-threading systems

In addition to the ability to satisfy inter-process message passing in communication between clients and servers via network sockets. Message passing between threads can also be performed in the same process, and this design is often preferred to design shared memory with affinity [40]. A synchronous queue is used to pass messages between threads. Queue performs the same function of a stored network communication channel for passing client / server messages. Java provides an interface to Blocking Queue for queues with blocking operations: On a regular queue:

add(e) adds element e to the ending of the queue. remove() deletes and returns the item at the top of the queue, or throws an exception if the queue is empty. Synchronous queue conforms to Java API documentation:

In addition, it helps processes that wait for the queue to be not empty when an item is retrieved, and wait until space is ready in the queue when an item is stored. put(e) blocks so that you can add item e to the end of the queue (if the queue does not have a restrictive size, it will not be blocked) [41].

take () blocks so you can delete the item and return it to the queue head, waiting for the queue to be not empty.

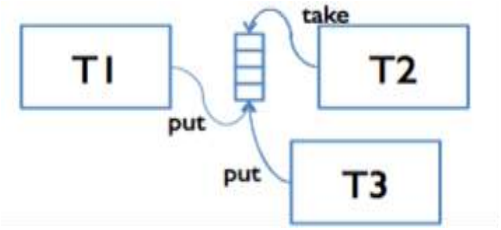

Figure 4. Queue Operations

Unlike sent and received byte streams using sockets, synchronous queues (such as the popular batch classes in Java) can carry random-type objects. As an alternative to designing a wired protocol, you must define or design a type of message in the queue. And just as it happened with operations on messages or ADT secure thread in a wired protocol, messages here must be designed to prevent race conditions and allow clients to continue the atomic operations they need. Similar to the client / server method of message passing through a network is the producer - consumer design pattern of passing the message between threads. Consumer and producer threads participate in a synchronized queue. Producers put data or orders on hold, and consumers obtain and process it [42]. The bounded-buffer [43] [1][14] (also known as problem producer–consumer issue) is a traditional sample of a multi-process synchronization problem. The issue defines two processes, the producer and the consumer, which share a general, buffer of limited -size used as a queue. The business of the producer is look like a data producing, buffering it and starting over at the same time as the consumer consumes the received data. The problem is how to ensure that the producer will not attempt to produce data to the buffer when it is full, also the consumer will not attempt to fetch data from an buffer when it empty [15] [44]. To solving the problem for the producer is to either reject the data if the buffer is full or go to sleep. Later the consumer will remove an item from the buffer, and informs the producer that it responsible for filling the buffer. In the same style, the consumer can go into pause

mode if the buffer was empty. Later the producer places data into buffer, it triggered the pausing consumer. The solution can be satisfied using inter-process communication method [45]. In this solution, each message has two elements: A data component being passed from the producer to the consumer and an empty/full flag. At first, the consumer transmits N messages marked as empty to the producer. The producer gets an empty message, blocking till one is available, fills it, and transmits it to the consumer. The consumer gets a filled message, blocking if needed, works on the data it contains, and returns the empty message to the producer [21]

More than one producer and one or more consumers may add and remove items from the same queue. This queue must be secure for synchronization. Java provides two representations of Blocking Queue:
Array Blocking Queue is a finite size queue that uses an array implementation. Putting a new item on the queue will block it if the queue is full. [44] LinkedBlockingQueue is not a limited queue with a linked list application. If the maximum size is not specified, the queue will not be full, so the mode will not be blocked. message passing in java can be represented in following ways...

- o Class variable and/or static variable of a class, can be used for all object.
- o Object passing through parameters can be used in a program.
- o Serialized object for message passing can be used in same/across jvm.
- o Properties can be used between two application system/server in one jvm, like in Tomcat sever Application Context is used to move object from one application to other [46].

Thread safety arguments with message passing

The message thread security argument with message passage may be based on:
The data types of the existing thread are secure for the synchronous queue. This queue is completely shared and completely changeable, so check if it's safe is necessary for the synchronization. [47] Stability of data and messages that may be accessible to more than one subject at the same time [48]. Limiting data to private producer/consumer threads. The local variables used by one producer or consumer are hidden in other threads, which only interact with each other using messages in the queue. Locked up data or changeable messages that are sent through a queue but can only be accessed through one message thread at a time. This argument must be applied and articulated with caution. But if one unit emits all signals to some mutable data sooner, it puts it on a queue to be sent to another thread, then only one thread will be able to access that data at a time, preventing simultaneous access [49] [50].

4. CONCLUSIONS

Different applications were explained to support the studied of message passing in this article, in which some studies for each of these applications were selected and clarified. Regarding the message passing topic, a collection of applications such as, agents, network protocols, producer and consumer, multi-threaded systems have been adopting message passing principle in work.

5. ACKNOWLEDGMENT

The authors would like to thank the University of Mosul / College of Computer Science and Mathematics for their facilities, which have helped to enhance the quality of this work.

6. REFERENCES

1. Ed Anderson, Scott Berryman, “MPI: A Message-Passing Interface Standard Version 3.0”, National Science Foundation Science and Technology Center Cooperative Agreement No. CCR-8809615, September 21, 2012.
2. Hesham El-Rewini Mostafa Abd-El-Barr,”Message Passing Architecture”,January 2005, https://doi.org/10.1002/0471478385.ch5.
3. Robert H. Halstead, Jr.Stephen A. Ward The MuNet,” A scalable decentralized architecture for parallel computatio ,The ACM Digital Library is published by the Association for Computing Machinery”. Copyright © 2019 ACM, Inc.
4. Péter Kacsuk, Gabriele Kotsis, “Distributed and Parallel Systems: From Instruction Parallelism to Cluster”, Springer Science & Business Media.LLC Copyright2012.
5. Phil Schniter, Approximate Message Passing: Applications to Communications Receivers, (With support from NSF grant CCF-1018368, NSF grant CCF-1218754, and DARPA/ONR grant N66001-10-1-4090), TrellisWare, Feb. 2014.
6. Coto A., Guanciale R., Tuosto E. (2020) On Testing Message-Passing Components. In: Margaria T., Steffen B. (eds) Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles. ISoLA 2020. Lecture Notes in Computer Science, vol 12476. Springer, Cham. https://doi.org/10.1007/978-3-030-61362-4_2.

7. F. Meyer et al., "Message Passing Algorithms for Scalable Multitarget Tracking," in Proceedings of the IEEE, vol. 106, no. 2, pp. 221-259, Feb. 2018, doi: 10.1109/JPROC.2018.2789427.
8. Akshay Rangesh, Pranav Maheshwari, Mez Gebre, Siddhesh Mhatre, Vahid Ramezani, Mohan M. Trivedi, TrackMPNN: A Message Passing Graph Neural Architecture for Multi-Object Tracking, [Submitted on 11 Jan 2021 (v1), last revised 28 Jan 2021 (this version, v3)]
9. P. MarendicEmail authorJ. LemeireD. VucinicP. Schelkens, "A novel MPI reduction algorithm resilient to imbalances in process arrival times" , An International Journal of High-Performance Computer Design, Analysis, and Use, May 2016, Volume 72.
10. William Cheng-Chung Chu ; Chih-Hung Chang, "Applying Software Defined Methodologies to Software Computing", 2016 Third International Conference on Trustworthy Systems and their Applications (TSA).
11. Elisa Gonzalez BoixEmail authorKevin De Porre," A Mobile Cross-Platform Actor Library for Multi-Networked Mobile Applications", Lecture Notes in Computer Science book series (LNCS, volume 10789), First Online: 07 September 2018.
12. FrédéricMagoulèsGuillaumeGbikpi-Benissan JACK2," An MPI-based communication library with non-blocking synchronization for asynchronous iterations", 2018 Elsevier Ltd.
13. OscarRodriguez,PrietoaFrancisco,OrtinaDonnaO'Sheab, "Information and Software Technology", Volume 100, August 2018, Pages 73-86, © 2018 Elsevier B.V.
14. Michel Raynal, "Fault-Tolerant Message-Passing Distributed Systems: An Algorithmic Approach",springer 2018.
15. Brandon Barker Workshop, "High Performance Computing on Stampede" ,January 14, 2015.
16. Kumar, Prashant, Zach, Richard Wang, "Implementation of Message Passing Language", May2018-02-22T17:22:09Z.
17. John M. BentSorin FaibishJames M. Pedone, Jr., " Message passing among interdependent parallel processes using a shared memory", US Patent 9,959,238, 2018.
18. Md. Mosaddek KhanLong Tran-Thanh, " A Near-Optimal Node-to-Agent Mapping Heuristic for GDL-Based DCOP Algorithms in Multi-Agent Systems", Proc. of the 17th International Conference
19. M. Dastani, G. Sukthankar, E. André, S. Koenig, "Autonomous Agents and Multiagent Systems(AAMAS)" (eds.), July 10–15, 2018,Stockholm, Sweden.
20. Long, Y. et al. On Ordering, "Problems in Message Passing Software", 15th International Conference on Modularity ,Mar. 2016.
21. Sean Pennefather ; Karen Bradshaw ; Barry Irwin, "Exploration and Design of a Synchronous Message Passing Framework for a CPU-NPU Heterogeneous Architecture", 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW).
22. Christopher J. ERDELYI, Ballwin, MO (US); Gerard Joseph ODONNELL, St. Peters, MO (US); Jensen James E. PASTRANA, O'Fallon, MO (US), "Method and system for converting asynchronous to synchronous transactions", Aug. 27, 2015, US 2015/0242847 A1.
23. Bernhard Kragl, Shaz Qadeer, "Synchronizing the Asynchronous", 29th International Conference on Concurrency Theory CONCUR 2018.
24. DavidPeleg, MicheleScquizzato,"Message lower bounds via efficient network synchronization November", 2018, Volume 758, febreuary 2019, issn 0304-3975.
25. Yu Huang, "An Analyzer for Message Passing Programs", Yu Huang, Copyright c 2016, Doctor of Philosophy ,Brigham Young University.
26. Shlomi Dolev, Thomas Petig, Elad Michael Schiller, "Self-Stabilizing and Private Distributed Shared Atomic Memory in Seldomly Fair Message Passing Networks", 9 Jun 2018 16:08:54 UTC.
27. Simone Pellegrini, "Simplifying and Optimizing Message Passing Programs: A Compiler and Runtime-Based Approach, Doctoral", November 2011, UNIVERSITY OF INNSBRUCK.
28. Vadim Kimlaychuk, "Simulations in Multi-Agent Communication System:, 2012 ISSN 1406-4731 ph.thesis.
29. Moises Lejter and Thomas Dean,"A Framework for the Development of Multi-Agent Architectures", e Advanced Research Projects Agency of the Department of Defense under grant No. F30602-95-1-0020, 1995.
30. T. Gonzaga, C. Bentes, R. Farias, M. C. S. de Castro and A. C. Garcia, "Using Distributed-Shared Memory Mechanisms for Agents Communication in a Distributed System," Seventh International Conference on

Intelligent Systems Design and Applications (ISDA 2007), Rio de Janeiro, 2007, pp. 39-46, doi: 10.1109/ISDA.2007.122.

31. Asli Celikyilmaz ,Antoine Bosselut ,Yejin Choi Microsoft Research Paul G. Allen, "Deep Communicating Agents for Abstractive Summarization School of Computer Science & Engineering", University of Washington, New Orleans, Louisiana, June 1 - 6, 2018. c 2018 Association for Computational Linguistics.
32. Pradnya Meshram, Kiran Gaware,"Bank Management system", International Research Journal of Engineering and Technology (IRJET) Volume: 05 Issue: 03 | Mar-2018.
33. Prachi Pathak, "AN ASSESSMENT OF BANK CREDIT LITERACY,ACCESSIBILITY AND SERVICE QUALITY AMONG WOMEN SELF HELP GROUPS", Doon University, Academy of Entrepreneurship Journal Volume 24, Issue 1, 2018.
34. Keichi Takahashi UnisonFlow, "A Software-Defined Coordination Mechanism for Message-Passing Communication and Computation", IEEE Access Published Volume: 6 : 23 April 2018, ISSN: 2169-3536.
35. Walsh et al, "Methods and Systems for Exception Detection and Reporting in a Distributed Network System", Pub . No . : US 2018 / 0062942 A1Mar . 1, 2018.
36. Gus SalamounMatthew,Parker Willingham Baggage, "messaging handling systems and methods", 2018-12-20US20180367454A1.
37. S. Lavanya and S. Prakasam, "Energy Efficient and Reliable MAC Protocol for Intra-cluster Communication in Application Specific Wireless Sensor Networks", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 1 (2018).
38. Muhammad Na'im Fikri Jamaluddin, Azlan Ismai, "Performance comparison of java based parallel programming models", Universiti Teknologi Mara ,Vol 16, No 3 ,December 2019
39. Riesen, Rolf & Maccabe, Arthur."RMPP: the reliable message passing protocol",658 - 666. 10.1109/LCN.2002.1181843,2002.
40. Anthony Rowe Karthik Lakshmanan,"SAMPL: A Simple Aggregation and Message Passing Layer for Sensor Networks", ACM ICST 978-963-9799-36-3,2008.
41. Graha mD. Hopkins Timothy J. Quigly And other, "Enhanced security when sending asynchronous messages", Apr . 10 ,2018, US20160294782A1, US 9 ,942 ,203 B2.
42. Dixon et al, "Dtermining where Bottlenecks Occur in Multi_Threaded Multi_Path Computing Systems" ,928 ,153 B2, Mar . 27 ,2018.
43. Grochowski et al, "Method and System to Provide User_Level Multithreading", Intel Corporation ,Santa Clara ,CA (US) Feb . 20 ,2018.
44. Ahmed Bouajjani, Constantin Enea, Kailiang Ji, Shaz Qadeer,"The Completeness of Verifying Message Passing Programs under Bounded Asynchrony", 2020 research and innovation programmer.
45. Julien Lange, Nicholas Ng, "A Static Verification Framework for Message Passing in Go using Behavioural Types", ,2018 ACM/IEEE 40th International Conference on Software Engineering.
46. S. Park, J. Lee and S. Hariri, "A multithreaded message-passing system for high performance distributed computing applications," Proceedings. 18th International Conference on Distributed Computing Systems (Cat. No.98CB36183), Amsterdam, Netherlands, 1998, pp. 258-265, doi: 10.1109/ICDCS.1998.679521.
47. James WhitneyChandler GiffordMaria Pantoja, "Distributed execution of communicating sequential process-style concurrency", Golang case study, An International Journal of High-Performance Computer Design, Analysis, and Use, 17 October 2018.
48. Mahmoud Husseini Orabi, Ahmed Husseini Orabi and Timothy C. Lethbridge, "Concurrent Prog

تطبيقات تمرير الرسائل: مراجعة

حليمة عسى سليمان أشرف عبدالمنعم عبدالمجيد غدا محمد الدباغ

قسم علوم الحاسوب ، كلية علوم الحاسوب والرياضيات، جامعة الموصل، موصل، العراق.

الخلاصة :من المعروف أن تمرير الرسائل أصبح من أشهر نماذج البرمجة الموازية لسهولة استخدامه، لذلك كان من الضروري معرفة أو دراسة التطبيقات التي كانت تعتمد على تمرير الرسالة في عملها والاطلاع عليها. يتم تصنيف نماذج البرمجة في الغالب حسب كيفية استخدام الذاكرة. في نموذج الذاكرة المشتركة، تصل كل دورة إلى مساحة موقع مشتركة، ولكن في نموذج تمرير الرسائل، يعمل التطبيق كمجموعة متنوعة من دورات الحكم الذاتي.

ولكل منها ذاكرتها المحلية الخاصة. التفضيلات الأساسية لإعداد معيار لتمرير الرسائل هي قابلية النقل والراحة. في مناخ مراسلات الذاكرة المتداولة حيث تستند جداول المستوى الأكثر أهمية وكذلك الانعكاسات إلى جداول تمرير الرسائل ذات المستوى الأدنى، تكون فوائد التطبيع واضحة بشكل خاص. علاوة على ذلك، فإن استخدام تمرير الرسالة، على سبيل المثال، المقترح هنا، يمنح البائعين ترتيبًا أساسيًا واضحًا للجداول الزمنية التي يمكنهم تنفيذها في الأيام الماضية، أو في الأوقات التي يمكنهم فيها دعم المعدات، وبالتالي تحسين القدرة على التكيف.

الكلمات المفتاحية: رسالة تمرير ،عملية الذاكرة المشتركة، متزامن بروتوكول، الشبكة غير المتزامن.