THE MAIN DIFFERENCES BETWEEN MPI AND THREADING TECHNOLOGIES

Ibragimova Muyassar Nazarali qizi

Ikhrorva Surayyo Isroiljon qizi

Tashkent State Medical University

Annotation. *In general, if one has the choice between an MPI parallel and Open MP parallel implementation, what the optimal choice, performance-wise? Programmer assuming it depends on what parts of the package are most often used. Yet, we do not more explain what difference them. This topic according the main unlike of MPI and Threading,*

Key Words: *Message Passing Interface, Open MP, Threading,*

Аннотация. *В общем, если у вас есть выбор между параллельной реализацией MPI и Open MP, какой оптимальный выбор, производительность? Программист предполагает, что это зависит от того, какие части пакета наиболее часто используются. Тем не менее, мы не более объясняем, в чем их отличие. Эта тема по сути отличается от MPI и Threading.*

Кулучевой слова: *Интерфейс передачи сообщений, Открыть MP, Threading,*

Annotatsiya. *Umuman olganda parallel MPI va Open MP parallel tizimlardan birini tanlashda optimal usulni tanlash qiyin masala. Dasturchilar odatda ishlab chiqilish qismidan foydalangan holda tanlovni amalga oshiradi. Halichacha biz bu kutubxonalarning farqlarini to'liq muhokama qila olmaymiz. Bu maqolada, MPI va Threading texnologiyalarinig o'xshash bo'lmagan jihatlari yoritilgan.*

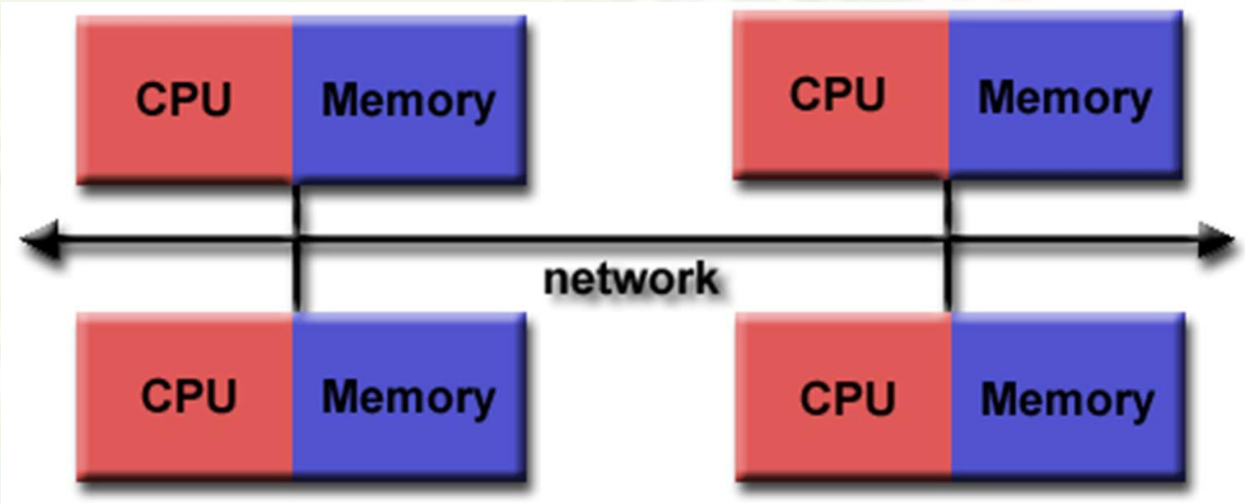
Kalit so'zlar: *Xabar yuborish interfeysi, Open MP, Threading,*

Introduction. The Message Passing Interface Standard (MPI) is a message passing library standard based on the consensus of the MPI Forum, which has over 40 participating organizations, including vendors, researchers, software library

developers, and users. The goal of the Message Passing Interface is to establish a portable, efficient, and flexible standard for message passing that will be widely used for writing message passing programs. As such, MPI is the first standardized, vendor independent, message-passing library. The advantages of developing message-passing software using MPI closely match the design goals of portability, efficiency, and flexibility. MPI is not an IEEE or ISO standard, but has in fact, become the "industry standard" for writing message passing programs on HPC platforms.

Threads are one of several technologies that make it possible to execute multiple code paths concurrently inside a single application. A thread in computer science is short for a thread of execution. Threads are a way for a program to divide (termed "split") itself into two or more simultaneously (or pseudo-simultaneously) running tasks. Threads and processes differ from one operating system to another but, in general, a thread is contained inside a process and different threads in the same process share same resources while different processes in the same multitasking operating system do not. Threads are lightweight, in terms of the system resources they consume, as compared with processes.

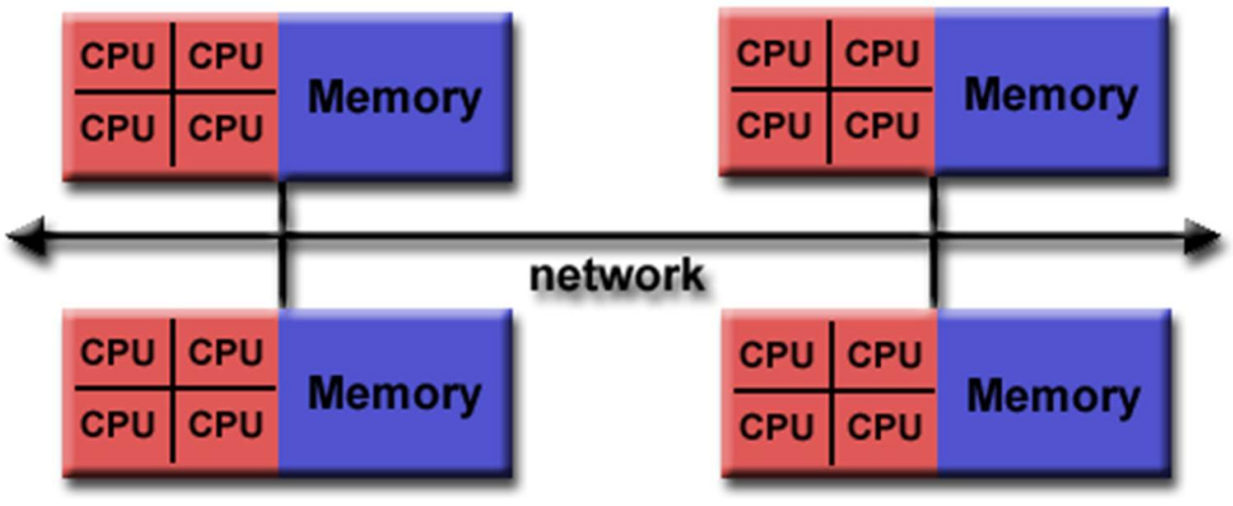
Message Passing Interface (MPI) Programming Model:Originally, MPI was designed for distributed memory architectures, which were becoming increasingly popular at that time (1980s - early 1990s).



Picture 1. Distributed memory architectures

1. As architecture trends changed, shared memory SMPs were combined over networks creating hybrid distributed memory / shared memory systems.

2. MPI implementers adapted their libraries to handle both types of underlying memory architectures seamlessly. They also adapted/developed ways of handling different interconnects and protocols.



Picture 2. Underlying memory architectures

1. Today, MPI runs on virtually any hardware platform:

✓ Distributed Memory

✓ Shared Memory

✓ Hybrid

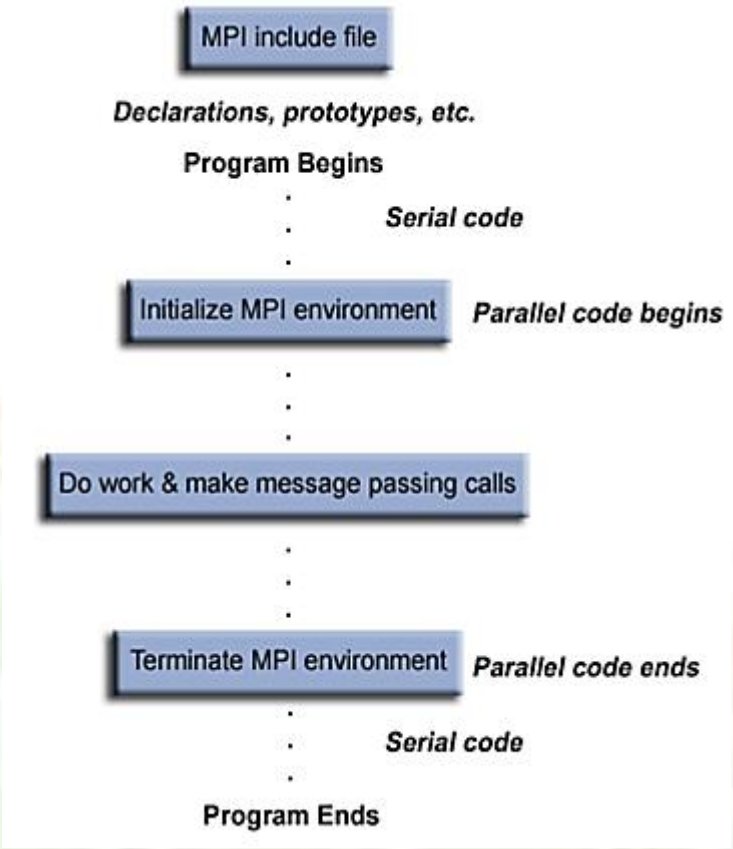
Reasons for Using MPI: Standardization - MPI is the only message-passing library, which can be considered a standard. It is supported on virtually all HPC platforms. Practically, it has replaced all previous message-passing libraries.

Portability - There is little or no need to modify your source code when you port your application to a different platform that supports (and is compliant with) the MPI standard.

Performance Opportunities - Vendor implementations should be able to exploit native hardware features to optimize performance. Any implementation is free to develop optimized algorithms.

Functionality - There are over 430 routines defined in MPI-3, which includes the majority of those in MPI-2 and MPI-1. Availability - A variety of

implementations are available, both vendor and public domain. General MPI Program Structure.



Picture 3. General MPI program structure.

Threading programming model and technologies

Three reasons for creating threads:

Therefore, it is clear why we do not want a whole bunch of threads. Similarly, there are a limited number of reasons forever-creating threads in the first place: To take advantage of multiple processors. This is the classic reason. If your software will typically be running on a multiprocessor or multicore machine, and you really do have more work than one processor can handle in a timely fashion, it makes sense to split that processing up into multiple independent threads of execution.

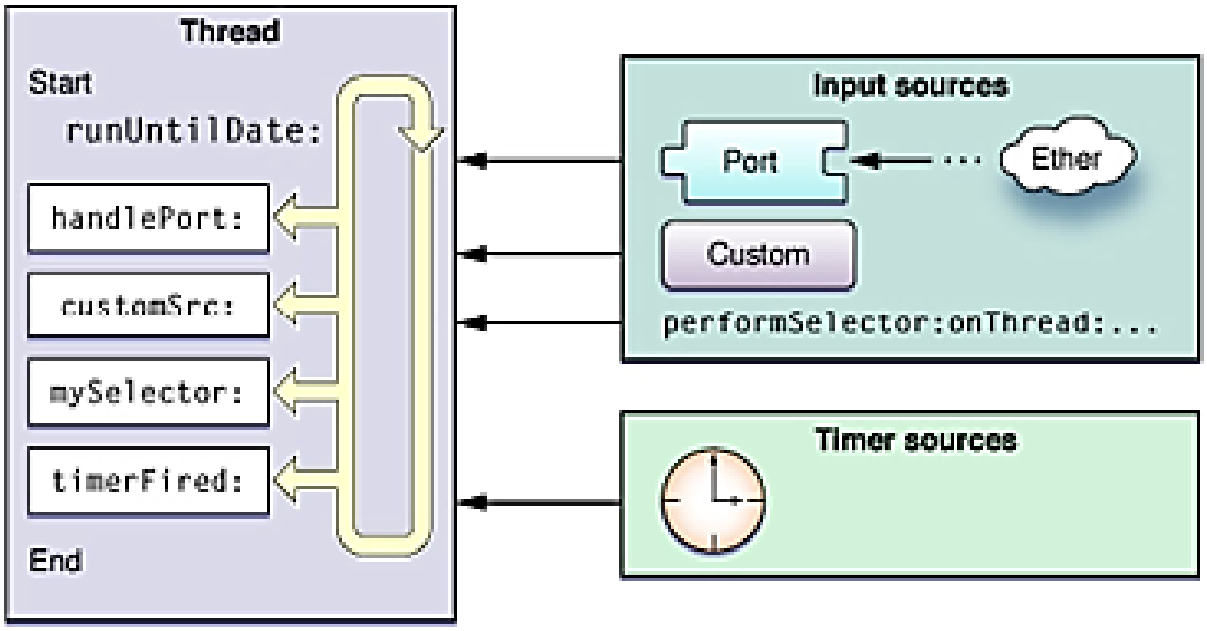
To move specific processing off the UI thread. This is a much more pragmatic reason. In the normal Windows world, you typically create threads so that a given background process does not choke your UI. In the Silverlight world where I have been living lately, this is less of a problem: Silverlight forces every sort of IO into an asynchronous pattern, and that tends to keep your UI thread from blocking.

However, the opposite can sometimes be an issue: the UI thread can get so busy that you need to spin up background threads so important tasks (like audio encoding or decoding) can get dispatched quickly enough. Nevertheless, the idea is the same: you want to keep your users from getting grumpy because important parts of their application appear to have slowly stopped.

To simplify complicated asynchronous calling patterns. Unless you actually need to take advantage of multiple processors, it is usually possible to get the effect of threads through a completely different mechanism, namely, using events to pass control from one part of the program to another. (This is called the Lauer/Needham Duality. In a famous 1979 paper, Lauer and Needham showed that "message-oriented" and "procedure-oriented" systems – read event-driven vs. multithreaded – were duals of each other and hence were logically equivalent architectures.) But although thread synchronization is difficult to get right, it can be even more painful to write a program using entirely asynchronous calls. The APIs for doing so are often complicated and obscure, and they require that you split your program's logic across various artificial boundaries. Depending on the complexity of your application, they may also require a that you implement a cooperative multitasking model, where any given function can be requested to yield to other functions, and only later pick up where it left off. These sorts of issues can make debugging and maintenance quite difficult. Threads are plenty complicated, but apart from the places where you start a thread, wait for a thread to finish, or lock some resource, multithreaded code looks reasonably similar to synchronous single-threaded code. And that's almost always a good thing. Run loops are part of the fundamental infrastructure associated with threads.

A run loop is an event processing loop that you use to schedule work and coordinate the receipt of incoming events. The purpose of a run loop is to keep your thread busy when there is work to do and put your thread to sleep when there is none. Run loop management is not entirely automatic. You must still design your thread's code to start the run loop at appropriate times and respond to incoming events. Both Cocoa and Core Foundation provide run loop objects to help you configure and

manage your thread's run loop. Your application does not need to create these objects explicitly; each thread, including the application's main thread, has an associated run loop object. Only secondary threads need to run their run loop explicitly, however. The app frameworks automatically set up and run the run loop on the main thread as part of the application startup process.



Picture 3. Structure of a run loop and its sources

Conclusion. Threading shares all memory between the threads. This is rather dangerous, since it is very easy to accidentally modify data that another thread might be using, leading to nasty bugs. The onus is on the programmer to carefully protect data against unsafe access. This also (usually) requires all processes to be running on the same machine, with access to the same physical memory. Using independent processes with a message-passing interface gives you more control over which data is shared and which is private to each process; there is little or no danger of one process unexpectedly modifying another process's state. In addition, as you say, the message-passing interface can be generalized to pass messages across a network between processes on separate machines.

REFERENCES.

1. Normamatov Sardor Fakhriddinovich, Safarov Ulug'bek Karshiboevich Tsifrovye individual plany raboty professorsko-podavatelskogo sostava v meditsinskom

obrazovanii. monitoring i otsenka v sisteme vysshego obrazovaniya Journal of new century innovations 1, 51-58 2026.

2. Normamatov Sardor Fakhriddinovich, Rakhimov Bobur Turgunovich Technology and medicine. diagnosticheskaya tochnost, prognozirovanie i kachestvo meditsinskikh uslug Journal of new century innovations 1, 43-50 2026.

3. Normamatov Sardor Fakhriddinovich, Otakhanov Polvonnazir Ergashovich Iskusstvennyy intellect v meditsine i ego znachenie Journal of new century innovations 1, 35-42 2026.

4. Normamatov Sardor Fakhriddinovich, Otakhanov Polvonnazir Ergashovich Montoring automatizirovannyx individualnyx planov raboty professorsko-podavatelskogo sostava v systeme meditsinskogo vyshego obrazovaniya. Journal of new century innovations 1, 29-34 2026 .

5. TSM Normamatov Sardor Fakhriddinovich, Rakhimov Bobur Turgunovich Artificial intelligence in medicine and its importance Journal of new century innovations 1, 8-15 2026.

6. UBS Normamatov Sardar Fakhriddinovich , Rakhimov Babur Turgunovich Medical supreme education in the system professor of teachers automated personal work of plans monitoring Journal of new century innovations 1, 3-7 2026.

7. NS Fakhriddinovich, SU Qarshiboyevich, XJ Muzaffar o'g'li AI technologies in medicine. Diagnostic accuracy, prognosis and service quality Journal of new century innovations 93 (1), 16-23 2026

8. RB Turgunovich, NS Fakhriddinovich, JZ Ravshanovna The Role Of Information Technology In Medicine And Biomedical Engineering In Training Future Specialists During The Period Of Digital Transformation In Education Web of Agriculture: Journal of Agriculture and Biological Sciences 2 (6), 1-8 2024.

9. S Normamatov, U Safarov, P Otokhanov, A Karabayev Algorithm for Teaching Fundamental Subjects Using Innovative Educational Technologies 2023.

10. SF Normamatov, A Koraboyev Methodology of teaching information technologies in medicine using innovative technologies Eurasian research in universal sciences 2023

11. S Normamatov, Z Jurayeva, P Otokhonov Methodology of teaching information technology in medical higher education institutions 2023.

12. S Normamatov, Z Jurayeva, P Otokhanov Teaching information technology in higher medical educational institutions 2023.

13. S Normamatov, U Safarov, P Otakhonov, A Koraboyev Application OF Artificial Intelligence in Clinical Decision-making Modern American Journal of Engineering, Technology, and Innovation 1 (2 ...

14. S Normamatov, S Sabirjanova, U Safarov, P Otakhanov, A Koraboyev clinical decision support systems based on artificial intelligence . the new uzbekistan journal of medicine. 2026.

15. S Normamatov, U Safarov, M Mirzahakimov, O Rozmurodov prediction of cardiovascular diseases with the help of artificial intelligence . the new uzbekistan journal of medicine.

16. N Sardor, I Farkhod, M Dilmurot Technologies for Accelerating Pharmaceutical Research Through Computer Modeling Modern American Journal of Engineering, Technology, and Innovation 1.

17. R Babur, B Muratali, S Abdusamad, J Ziyoda. The Importance of Digital Technologies in the Teaching of Fundamental Sciences in Medical Universities. American Journal of Medicine and Medical Sciences. 1 2023

18. AUM Abdujabbarova, AZ Sobirjonov, KD Latipova. Features of teaching biophysics to medical students. British Journal of Global Ecology and Sustainable Development. 1 2023

19. UM Abdujabborova, AZ Sobirjonov, FS Tuxtakhodjaeva. Justification of religious consciousness and moral norms in different religions. Academic research in educational sciences, 59-63 1 2022

20. AZ Sobirjonov. The role of Abu Rayhan Beruni's "Saydana" in pharmacology. Academic research in educational sciences, 335-339