ОСНОВНЫЕ РАЗЛИЧИЯ МЕЖДУ MPI И ПОТОКОВЫМ ТЕХНОЛОГИЧЕСКИМ УПРАВЛЕНИЕМ ТЕХНОЛОГИИ

Ибрагимова Муяссар Назарали қизи

Ихрорва Сурайё Исроилжон қизи

Ташкентский государственный медицинский университет

Примечание. *В общем, если есть выбор между параллельной реализацией MPI и параллельной реализацией OpenMP, какой вариант является оптимальным с точки зрения производительности? Программист предполагает, что это зависит от того, какие части пакета используются чаще всего. Однако мы не будем подробно объяснять, в чем между ними разница. Эта тема, в основном, отличается от MPI и многопоточности.*

Ключевые слова: *Интерфейс передачи сообщений, OpenMP, многопоточность,*

Аннотация. *В общем, если у вас есть выбор между параллельной реализацией MPI и Open MP, какой выбор, производительность? Программист предполагает, что это зависит от того, какая часть пакета используется чаще всего. Тем не менее, мы не более объясненем, в чем их отличие. Эта тема по существу отличается от MPI и Threading.*

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

Аннотация. *Умуман Ольганда, параллель MPI ва Open MP, параллель тизимлардан бирини танлашда, оптимальные усулни танлаш циин масала. Дастурчилар одатда ишлаб чикилиш кисмидан фойдаланган холда танловни амальга оширади. Halichacha biz bu kutubxonalarning Farqlarini to'liq muhokama qila olmaymiz. Если вы хотите, MPI и технологии Threading или O'xshash bo'lmagan jihatlari yoritilgan.*

Калит со'злар: *Xabar юбориш интерфейса, Open MP, Threading,*

Введение. Стандарт интерфейса передачи сообщений (MPI) — это стандарт библиотеки для передачи сообщений, основанный на консенсусе форума MPI, в котором участвуют более 40 организаций, включая поставщиков, исследователей, разработчиков программных библиотек и пользователей. Цель интерфейса передачи сообщений — создать переносимый, эффективный и гибкий стандарт для передачи сообщений, который будет широко использоваться для написания программ, использующих передачу сообщений. Таким образом, MPI является первой стандартизированной, независимой от поставщиков библиотекой для передачи сообщений. Преимущества разработки программного обеспечения для передачи сообщений с использованием MPI точно соответствуют целям проектирования: переносимости, эффективности и гибкости. MPI не является стандартом IEEE или ISO, но фактически стал «отраслевым стандартом» для написания программ, использующих передачу сообщений, на высокопроизводительных вычислительных платформах.

Потоки — это одна из технологий, позволяющих одновременно выполнять несколько участков кода в рамках одного приложения. В информатике поток — это сокращение от «поток выполнения». Потоки — это способ для программы разделить себя (называемый «разделить») на две или более одновременно (или псевдоодновременно) выполняющихся задач. Потоки и процессы различаются в зависимости от операционной системы, но, как правило, поток находится внутри процесса, и разные потоки в одном и том же процессе используют одни и те же ресурсы, в то время как разные процессы в одной и той же многозадачной операционной системе — нет. Потоки являются легковесными с точки зрения потребляемых системных ресурсов по сравнению с процессами.

Программная модель Message Passing Interface (MPI) : Первоначально MPI был разработан для распределенных архитектур памяти, которые в то время (1980-е — начало 1990-х годов) становились все более популярными.

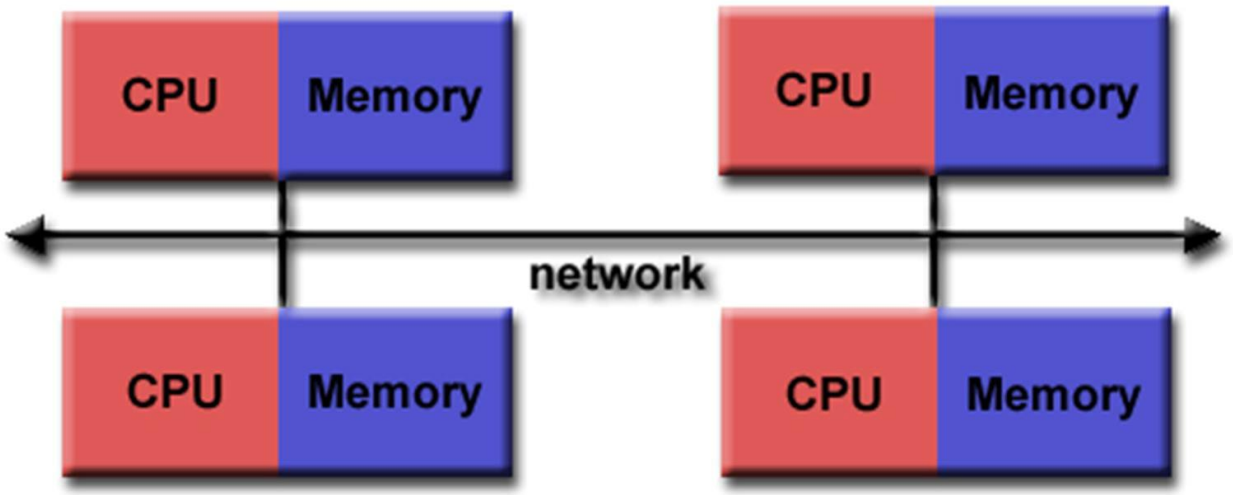

Рисунок 1. Архитектуры распределенной памяти

1. По мере изменения архитектурных тенденций, многопроцессоры с общей памятью объединялись в сети, создавая гибридные системы распределенной и общей памяти.

2. Разработчики MPI адаптировали свои библиотеки для бесперебойной работы с обоими типами базовых архитектур памяти. Они также адаптировали/разработали способы обработки различных межсоединений и протоколов.

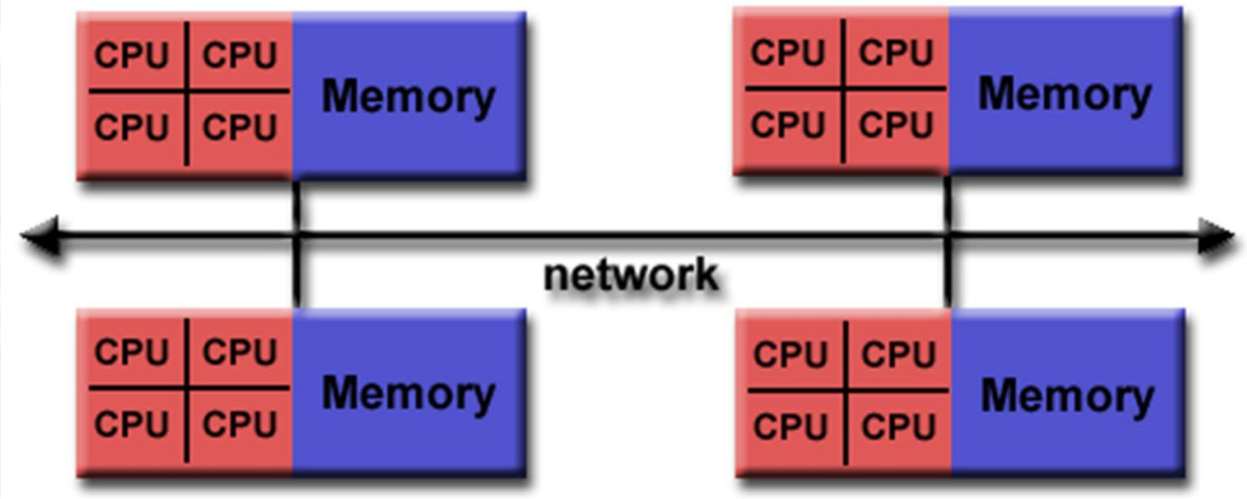

Рисунок 2. Базовые архитектуры памяти.

1. Сегодня MPI работает практически на любой аппаратной платформе:

- ✓ Распределенная память
- ✓ Общая память

✓ Гибридный

Причины использования MPI: Стандартизация — MPI — единственная библиотека для передачи сообщений, которую можно считать стандартом. Она поддерживается практически на всех высокопроизводительных вычислительных платформах. Практически она заменила все предыдущие библиотеки для передачи сообщений.

Портативность — при портировании приложения на другую платформу, поддерживающую (и совместимую) со стандартом MPI, практически нет необходимости изменять исходный код.

Возможности повышения производительности — Реализации от поставщиков должны иметь возможность использовать встроенные аппаратные функции для оптимизации производительности. Любая реализация может свободно разрабатывать оптимизированные алгоритмы.

Функциональность — В MPI-3 определено более 430 подпрограмм, включая большинство подпрограмм из MPI-2 и MPI-1. Доступность — Доступно множество реализаций, как от поставщиков, так и общедоступных. Общая структура программы MPI.

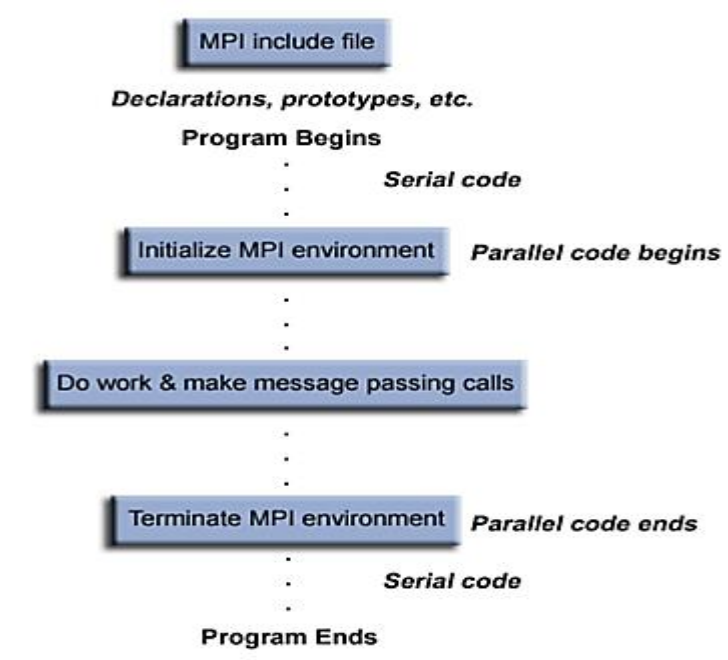

Рисунок 3. Общая структура программы MPI.

Многопоточная модель программирования и технологии

Три причины для создания тем:

Таким образом, становится ясно, почему нам не нужно множество потоков. Аналогично, существует ограниченное количество причин для постоянного создания потоков: использование преимуществ многопроцессорности. Это классическая причина. Если ваше программное обеспечение обычно будет работать на многопроцессорной или многоядерной машине, и у вас действительно больше работы, чем один процессор может обработать за короткое время, имеет смысл разделить эту обработку на несколько независимых потоков выполнения.

Перенос определённой обработки с потока пользовательского интерфейса. Это гораздо более прагматичная причина. В обычной среде Windows потоки обычно создаются для того, чтобы фоновый процесс не перегружал пользовательский интерфейс. В мире Silverlight, в котором я работаю в последнее время, это меньшая проблема: Silverlight принудительно переводит все виды ввода-вывода в асинхронный режим, и это, как правило, предотвращает блокировку потока пользовательского интерфейса. Однако иногда может возникнуть обратная ситуация: поток пользовательского интерфейса может стать настолько загруженным, что потребуется запустить фоновые потоки, чтобы важные задачи (например, кодирование или декодирование звука) могли быть выполнены достаточно быстро. Тем не менее, идея та же: вы хотите, чтобы ваши пользователи не раздражались из-за того, что важные части их приложения, кажется, постепенно остановились.

Для упрощения сложных шаблонов асинхронных вызовов. Если вам не требуется использовать преимущества многопроцессорной архитектуры, обычно можно добиться эффекта потоков с помощью совершенно другого механизма, а именно, используя события для передачи управления от одной части программы к другой. (Это называется двойственностью Лауэра/Нидхема. В известной статье 1979 года Лауэр и Нидхем показали, что «ориентированные на сообщения» и «ориентированные на процедуры» системы — то есть событийно-ориентированные и многопоточные —

являются двойственными друг другу и, следовательно, логически эквивалентными архитектурами.) Но хотя синхронизацию потоков сложно правильно настроить, еще сложнее написать программу, использующую исключительно асинхронные вызовы. API для этого часто сложны и неясны, и они требуют разделения логики вашей программы на различные искусственные границы. В зависимости от сложности вашего приложения, они также могут потребовать реализации модели кооперативной многозадачности, где любой заданной функции может быть предложено передать управление другим функциям, и только позже продолжить с того места, где она остановилась. Подобные проблемы могут значительно затруднить отладку и сопровождение. Потоки — довольно сложная система, но, за исключением мест, где запускается поток, ожидается его завершение или блокируется какой-либо ресурс, многопоточный код выглядит достаточно похожим на синхронный однопоточный код. И это почти всегда хорошо. Циклы выполнения являются частью фундаментальной инфраструктуры, связанной с потоками.

Цикл выполнения — это цикл обработки событий, используемый для планирования работы и координации получения входящих событий. Цель цикла выполнения — поддерживать поток в рабочем состоянии, когда есть работа, и переводить его в спящий режим, когда работы нет. Управление циклом выполнения не является полностью автоматическим. Вам все равно необходимо разработать код потока таким образом, чтобы цикл выполнения запускался в подходящее время и реагировал на входящие события. И Cocoa, и Core Foundation предоставляют объекты цикла выполнения, которые помогают настраивать и управлять циклом выполнения потока . Вашему приложению не нужно создавать эти объекты явно; каждый поток, включая основной поток приложения, имеет связанный с ним объект цикла выполнения. Однако только вторичным потокам необходимо явно запускать свой цикл выполнения. Фреймворки приложений автоматически настраивают и запускают цикл выполнения в основном потоке в рамках процесса запуска приложения.

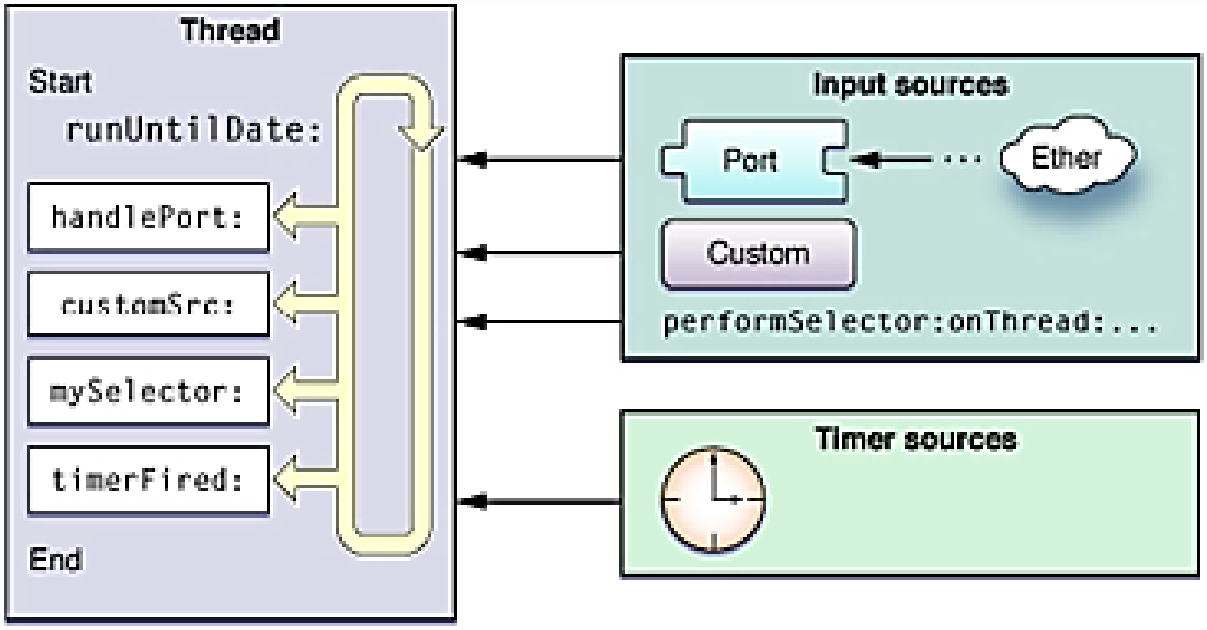

Рисунок 3. Структура цикла выполнения и его источники.

В заключение. Многопоточность предполагает совместное использование всей памяти потоками. Это довольно опасно, поскольку очень легко случайно изменить данные, которые может использовать другой поток, что приводит к неприятным ошибкам. Ответственность за тщательную защиту данных от небезопасного доступа лежит на программисте. Это также (обычно) требует, чтобы все процессы работали на одной машине и имели доступ к одной и той же физической памяти. Использование независимых процессов с интерфейсом передачи сообщений дает больший контроль над тем, какие данные являются общими, а какие — частными для каждого процесса; практически отсутствует опасность того, что один процесс неожиданно изменит состояние другого процесса. Кроме того, как вы правильно заметили, интерфейс передачи сообщений можно обобщить для передачи сообщений по сети между процессами на разных машинах.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.

1. Нормаматов Сардор Фахриддинович, Сафаров Улугбек Каршибоевич Цифровые индивидуальные планы работы профессорско-подавательского состава в медицинском образовании. мониторинг и оценка в системе высшего образования Журнал инноваций нового века 1, 51-58 2026.

2. Нормаматов Сардор Фахриддинович, Рахимов Бобур Тургунович Технологии и медицина. диагностическая точность, прогнозирование и качество медицинских услуг Журнал инноваций нового века 1, 43-50 2026.

3. Нормаматов Сардор Фахриддинович, Отаханов Полвонназир Эргашович Искусственный интеллект в медицине и его значение Журнал инноваций нового века 1, 35-42 2026.

4. Нормаматов Сардор Фахриддинович, Отаханов Полвонназир Эргашович Мониторинг автоматизированных индивидуальных планов работы профессорско-подавательского состава в системе медицинского высшего образования. Журнал инноваций нового века 1, 29-34 2026 .

5. ТСМ Нормаматов Сардор Фахриддинович, Рахимов Бобур Тургунович Искусственный интеллект в медицине и его значение Журнал инноваций нового века 1, 8-15 2026.

6. УБС Нормаматов Сардар Фахриддинович , Рахимов Бабур Тургунович Медицинский верховный образование в системе профессор учителей автоматизированный личный работа планов Мониторинг инноваций нового века 1, 3-7 2026.

7. Н.С. Фахриддинович, С.У. Каршибоевич, Х.Дж. Музаффар о'гли. Технологии ИИ в медицине. Точность диагностики, прогноз и качество обслуживания. Журнал инноваций нового века 93 (1), 16-23 2026

8. Р.Б. Тургунович, Н.С. Фахриддинович, Дж.З. Равшановна. Роль информационных технологий в медицине и биомедицинской инженерии в подготовке будущих специалистов в период цифровой трансформации образования. Веб сельского хозяйства: Журнал сельского хозяйства и биологических наук 2 (6), 1-8 2024.

9. С. Нормаматов, У. Сафаров, П. Отоханов, А. Карабаев. Алгоритм обучения фундаментальным предметам с использованием инновационных образовательных технологий, 2023.

10. С.Ф. Нормаматов, А. Корабойев. Методика преподавания информационных технологий в медицине с использованием инновационных технологий. Евразийские исследования в универсальных науках, 2023.

11. С. Нормаматов, З. Юраева, П. Отохонов. Методология преподавания информационных технологий в медицинских высших учебных заведениях. 2023.

12. С. Нормаматов, З. Джураева, П. Отоханов. Преподавание информационных технологий в высших медицинских учебных заведениях, 2023.

13. С. Нормаматов, У. Сафаров, П. Отахонов, А. Корабойев. Применение искусственного интеллекта в принятии клинических решений. Современный американский журнал инженерии, технологий и инноваций 1 (2 ...

14. Нормаматов С., Сабирджанова С., Сафаров У., Отаханов П., Корабоев А. Системы поддержки клинических решений на основе искусственного интеллекта . новый узбекский медицинский журнал. 2026.

15. С. Нормаматов, У. Сафаров, М. Мирзахакимов, О. Розмуродов. Прогнозирование сердечно-сосудистых заболеваний с помощью искусственного интеллекта . Новый узбекский медицинский журнал.

16. Н. Сардор, И. Фарход, М. Дилмурот. Технологии ускорения фармацевтических исследований посредством компьютерного моделирования. Современный американский журнал инженерии, технологий и инноваций, том 1.

17. Р. Бабур, Б. Муратали, С. Абдусамад, Дж. Зийода. Важность цифровых технологий в преподавании фундаментальных наук в медицинских университетах. Американский журнал медицины и медицинских наук. 1 2023

18. АУМ Абдуджаббарова, АЗ Собиржонов, КД Латипова. Особенности преподавания биофизики студентам-медикам. Британский журнал глобальной экологии и устойчивого развития. 1 2023

19. У.М. Абдуджабборова, А.С. Собиржонов, Ф.С. Тукстаходжаева. Обоснование религиозного сознания и моральных норм в различных религиях. Академические исследования в области педагогических наук, 59-63 1 2022

20. А. С. Собиржонов. Роль «Сайданы» Абу Райхана Беруни в фармакологии. Академические исследования в области педагогических наук, 335-339.