**Разница между режимами MPI send**

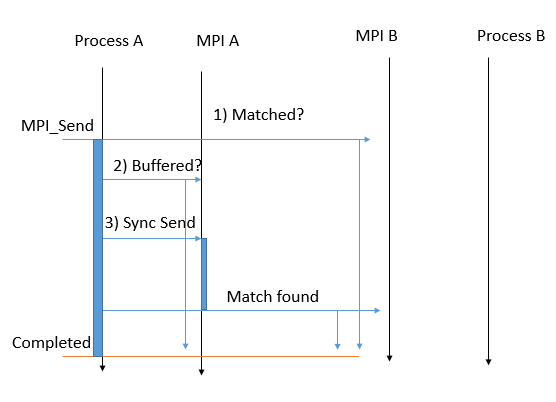
09 ноя 2020

**Введение**

Существуют различные режимы *MPI посыла* : MPI\_Send, MPI\_Isend, MPI\_Ssend, MPI\_Bsend, и так далее. Они могут быть локальными или нелокальными, блокирующими или неблокирующими, синхронными или асинхронными. Их определение и различия подробно описаны здесь.

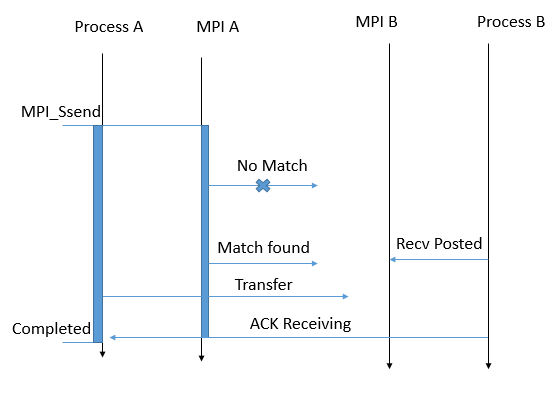
**MPI\_Send**

Это **стандартный** режим. Когда он вызывается, (1) сообщение может быть напрямую передано в буфер приема, (2) данные буферизируются (во временной памяти в реализации MPI) или (3) функция ожидает появления принимающего процесса. См. Картинку ниже. Следовательно, он может быстро вернуться (1) (2) или заблокировать процесс на некоторое время (3). MPI решает, какой сценарий лучше всего с точки зрения производительности, памяти и так далее. В любом случае данные можно безопасно изменить после возврата из функции.



**MPI\_Ssend**

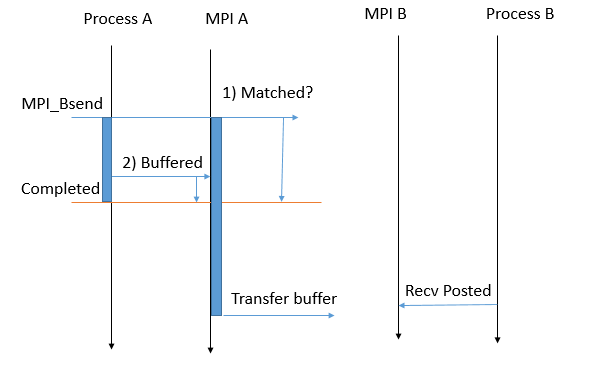
Это **синхронизированная** функция блокировки. Когда эта функция возвращается, адресат начал получать сообщение. В тот момент, когда пункт назначения начинает получать, он отправляет сигнал *ACK* источнику.



Обратите внимание, что сигнал о получении сообщения - это разница между MPI\_Ssendи MPI\_Send.

**MPI\_Bsend**

Это локальная блокирующая отправка. При вызове этой функции программист определяет **локальный буфер** . Если подходящего приема нет, процесс блокируется до тех пор, пока сообщение не будет скопировано в буфер. Следовательно, программист может сразу изменить исходные данные после возврата из функции.



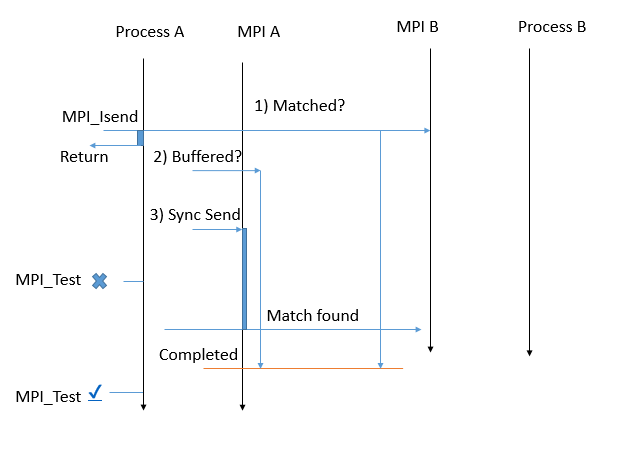
Обратите внимание: когда функция возвращает сообщение, вероятно, еще не отправленное, это произойдет одновременно в фоновом режиме процесса, когда будет найдено согласованное получение.

**MPI\_Rsend**

Это функция блокировки, такая же, как MPI\_Send, но она ожидает, что **адресат готов** к приему сообщения. Это может повысить производительность MPI, если программист уверен, что этого ожидает функция приема. Если ранее не было получено сообщение о получении, оно ошибочно.

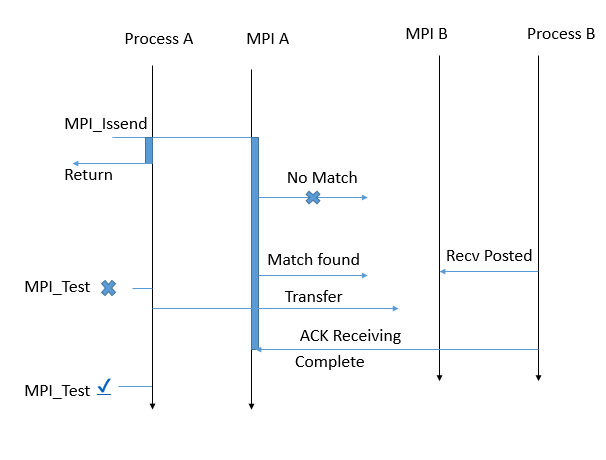
**MPI\_Isend**

Это неблокирующая версия MPI\_Send. Когда эта функция вызывается, функция немедленно возвращается, но выполняет MPI\_Sendдействия в фоновом режиме процесса. Таким образом, после завершения функции, данные **должны не** быть изменены , если MPI\_Testи MPI\_Waitподтверждения MPI\_Isendзавершения. После завершения данные можно использовать повторно, поскольку они либо помещаются в буфер в MPI, либо отправляются по назначению.



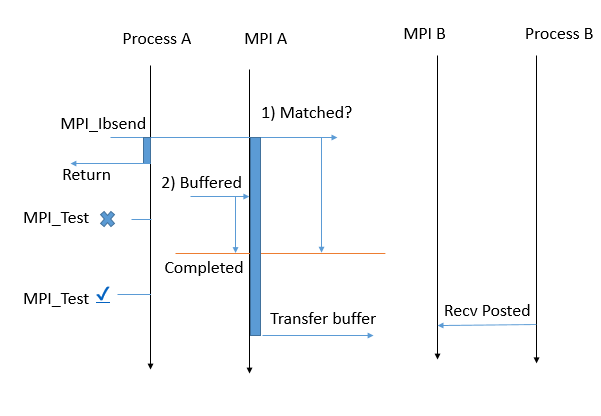
**MPI\_Issend**

Это неблокирующая версия MPI\_Ssend. Он возвращается немедленно, но выполняет MPI\_Ssendдействия в фоновом режиме. MPI\_Testили MPI\_Wait **должен** использоваться, чтобы оценить, выполняется ли функция в фоновом режиме. В этот момент не только сообщение было отправлено, но и пункт назначения начал получать сообщение.



**MPI\_Ibsend**

Это местная неблокирующая отправка. Он блокирует ни копирование сообщения в буфер, ни отправку сообщения. После положительного результата теста или ожидания мы можем изменить исходные данные, потому что, если они не отправлены, они локально копируются в выделенный буфер.



**MPI\_Irsend**

То же, что и MPI\_Rsend, но без блокировки.

**Список приоритетов**

Трудно дать рецепт всех проблем. Однако некоторые моменты могут помочь выбрать правильный режим:

* Когда возникает **тупик** , может помочь неблокирующая связь.
* Когда возникает состояние [**гонки**](https://iamsorush.com/posts/mpi-race-condition/) , может помочь блокировка связи.
* Когда есть задача, **требующая больших вычислительных ресурсов** , неблокирующее сообщение, размещенное перед задачей, может улучшить производительность.
* MPI\_Sendи MPI\_Isendзанимают **первое** место в списке приоритетов, поскольку MPI решает, что лучше.
* MPI\_Ssendи MPI\_Issend, когда отправителю необходимо знать, **когда** сообщение получено, и избежать локальной буферизации. MPI\_Ssendможет быть полезно для отладки.
* MPI\_Bsend, MPI\_Ibsend, MPI\_Rsend, MPI\_IrsendПредназначены для **тонкой настройки** на производительность.

**использованная литература**

Я получил идеи и коды с указанных ниже веб-сайтов.

[Отчет MPI 3.1 - разделы 3.4, 3.7](https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf)

**Связанные теги**

[MPI](https://iamsorush.com/tags/mpi)

**Последние посты**

* [Span - новая норма в кодах C ++](https://iamsorush.com/posts/cpp-span/)
* [Основной шрифт и расширения VS Code для C ++](https://iamsorush.com/posts/vs-code-extensions/)
* [Обзор C ++ perfect forwarding](https://iamsorush.com/posts/perfect-forwarding-cpp/)
* [Полезен ли статический полиморфизм C ++?](https://iamsorush.com/posts/static-polymorphism-cpp/)
* [Из lvalue, prvalue и xvalue для перемещения семантики в C ++](https://iamsorush.com/posts/move-semantics-cpp/)