DOI: 978-966-301-266-7/1.2024.59

Захаренков Д.Ю.
https://orcid.org/0000-0003-3951-022X
e-mail: dmit.zakharen@gmail.com

Литвиненко Л.О.
https://orcid.org/0000-0002-0828-383X
e-mail: leonid.lytvynenko@e-u.edu.ua

Яровий Р.О.
https://orcid.org/0000-0001-8978-8137
e-mail: roman.yaroviy@e-u.edu.ua

СУЧАСНІ ТЕХНОЛОГІЇ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ В МОДЕЛІ РОЗПОДІЛЕНОЇ ПАМ'ЯТІ

Анотація. Представлений огляд реалізацій MPI-засобів паралелізації, орієнтований на область високопродуктивних обчислень. Наведено аналіз продуктивності MPI-засобів, досягнутої на основних реалізаціях MPI при використанні сучасних міжз'єднань з віддаленим прямим доступом до пам'яті (RDMA), таких як Intel Omni-Path та Infiniband EDR. Особливу увагу приділено одностороннім RMA-комунікаціям, які сприяють підвищенню продуктивності та підтримці перспективних PGAS-моделей паралелізації.

Ключові слова: засоби паралельного програмування, Message Passing Interface (MPI), OpenMPI, MVAPICH2, Intel MPI, тестування продуктивності.

Zakharenkov Dmytro, Lytvynenko Leonid, Yarovyi Roman

MODERN TECHNOLOGIES OF PARALLEL PROGRAMMING IN THE DISTRIBUTED MEMORY MODEL

Abstract. Presentations on the implementation of MPI parallelization features, targeting the area of highly productive calculations. An analysis of the productivity of MPI functions achieved on the main MPI implementations with a choice of daily interconnections with remote direct memory access (RDMA), such as Intel Omni-Path and Infiniband EDR, is carried out. Particular attention is paid to one-way RMA communications, which promote increased productivity and support for promising PGAS models of parallelization.

Key words: features of parallel programming, Message Passing Interface (MPI), OpenMPI, MVAPICH2, Intel MPI, productivity testing.

Виклад основного матеріалу. Ця стаття присвячена аналізу технологій паралельного програмування. Оскільки спектр таких технологій дуже широкий, у роботі розглянуті лише сучасні, широко поширені та перевірені на практиці засоби розпаралелювання, які стали стандартами. З урахуванням тенденції збільшення кількості доступних процесорів (процесорних ядер) у всіх сучасних обчислювальних системах – від ПК до суперкомп'ютерів – актуальність паралелізації зростає у різних галузях практичного застосування. Однак стаття в першу чергу орієнтована на область високопродуктивних обчислень (High Performance Computing, HPC). Використання паралельних технологій давно стало одним із ключових способів підвищення продуктивності та скорочення часу розрахунку.

Хоча засоби паралелізації доступні в різних операційних системах, в області HPC і суперкомп'ютерах основним рішенням стала кластерна архітектура на базі операційної системи Linux. В першу чергу такі засоби застосовуються на платформі x86-64, де використовуються високошвидкісні міжз'єднання з віддаленим прямим доступом до пам'яті (RDMA), які не навантажують процесори віддаленого вузла, такі як Infiniband, Ethernet з RDMA (RoCE або iWARP) або Intel Omni-Path. Оцінки продуктивності, наведені у статті, насамперед ставитимуться до таких систем. Тим не менш, засоби паралелізації, звичайно ж, застосовуються і для обчислювальних систем з іншими процесорами та міжз'єднаннями.

Зазвичай моделі паралельного програмування класифікуються за топологією пам'яті, що використовується наступним чином:

• моделі із загальною пам'яттю, призначені для SMP та NUMA-систем;

• моделі з розподіленою пам'яттю, які застосовуються, в тому числі, для кластерів;

• гібридні моделі, що поєднують розподілену та загальну пам'ять, що характерно для сучасних кластерів із багатопроцесорними вузлами, як зазначено в оглядах [1, 2].

Основу для досягнення високої продуктивності на сучасних суперкомп'ютерах, які використовують кластерну архітектуру, становлять засоби паралелізації з розподіленою пам'яттю, які є найбільш поширеними. У цьому огляді проаналізовано саме ці засоби паралелізації. Засоби OpenMP [3], перспективні технології PGAS (Partitioned Global Address Space) [4], а також спеціалізовані засоби паралелізації для прискорювачів у цій статті не розглядаються.

Модель надсилання повідомлень. Розподілена пам'ять

Різні моделі передачі повідомлень відносяться до паралельного програмування з розподіленою пам'яттю, в яких процеси обмінюються даними, надсилаючи та отримуючи повідомлення. Це типово для кластерів обчислювальних вузлів, з'єднаних комунікаційною мережею, де кожен вузол має свою локальну пам'ять, а прямий доступ до пам'яті інших вузлів відсутній.

Існує безліч моделей передачі повідомлень, які представлені у вигляді програмних бібліотек. Далі будуть розглянуті засоби, що найбільш широко застосовуються і довели свою ефективність MPI [5]. Ці моделі передбачають використання бібліотечних викликів і відповідають типу MIMD (множинний потік команд, множинний потік даних класифікації Флінна [5]). Інтерфейс MPI може використовуватись у обчислювальних системах як із загальною, так і з розподіленою пам'яттю.

Стандарт MPI 2.0 запровадив можливість динамічної зміни числа процесів, розширені колективні комунікації та односторонні комунікації MPI_Put/MPI_Get. Ці можливості були додатково покращені у стандарті MPI 3.0, який також дозволив безпосередньо використовувати можливості RDMA. З MPI 2.0 почало підтримуватися паралельне введення-виведення, а остання версія стандарту, включає колективні неблокуючі операції введення-виведення [6].

Односторонні зв'язки базуються на механізмі віддаленого доступу до пам'яті (Remote Memory Access, RMA). У MPI односторонні комунікації мають доступ тільки до так званого вікна, тобто області пам'яті на цілі, яка повідомляє про доступність вікна для інших процесів. Альтернативою використанню вікон є віртуальна пам'ять. Модель MPI RMA дозволяє блокувати вікно і має два режими – активний та пасивний. В активному режимі ціль встановлює період часу, протягом якого доступ до вікна можливий. У пасивному режимі RMA-процес оголошення цілі доступ до вікна не обмежує. Ефективне використання MPI RMA у загальній моделі паралельного програмування PGAS, що поєднує стилі загальної та розподіленої пам'яті, сприяло міграції деяких реалізацій MPI у бік PGAS.

В даний час існує безліч реалізацій MPI, включаючи MPICH, MVAPICH, OpenMPI, Intel MPI, Mellanox X-MPI, IBM Spectrum MPI, HPE MPI та Cray MPI. У рамках цієї статті ми докладніше розглянемо три найактивніші реалізації в широко поширених HPC системах на базі Linux з архітектурою x86-64: OpenMPI, MVAPICH2 та комерційну версію Intel MPI.

Ці реалізації MPI з RDMA використовують універсальні низькорівневі засоби API, засновані на драйверах. Раніше для цього застосовувалися ibverbs із OFED (OpenFabrics Enterprise Distribution) та uDAPL (User Direct Access Programming Library), стандартний API-інтерфейс для RDMA-міжз'єднань

Infiniband, iWARP та RoCE. Програмні засоби OFED мають більш широкі можливості і давно містять uDAPL. Ці низькорівневі API використовуються не тільки для MPI, але й для інших засобів паралелізації, які працюють з RDMA.

Хоча OFED є некомерційним продуктом з доступним вихідним кодом, виробники міжз'єднань його доробляють для своїх апаратних засобів, створюючи варіанти низькорівневих API. Наприклад, компанія Mellanox пропонує свій варіант OFED для міжз'єднань Infiniband та RoCE з низькорівневим драйвером MLX5. Альтернативою цим продуктам для Infiniband є вільно доступні програмні засоби UCX (Unified Communication X), які можуть використовуватися не тільки з MPI, але й у ширшому програмному інструментарії Mellanox HPC-X, що використовується в MPI та PGAS-засобах паралелізації. Також слід відзначити бібліотеку для Infiniband - UCCS (Universal Common Communication Substrate), яка може вважатися альтернативою низькорівневої бібліотеки Mellanox MXM. Вона надає високопродуктивні комунікаційні примітиви, забезпечуючи інтероперабельність мереж та підтримку різних моделей програмування, включаючи MPI та PGAS-засоби. За допомогою UCCS реалізуються різні операції комунікації, включаючи односторонні та двосторонні операції "точка-точка", колективні та віддалені атомарні операції.

Нещодавно для OFED з'явилася альтернатива у вигляді OFI (OpenFabrics Interfaces). Бібліотека libfabric, що входить до складу OFI і використовується в деяких сучасних реалізаціях MPI, представляє API з більш високим рівнем абстракції, ніж libibverbs. Однак виробники ibverbs дозволяють програмним засобам, що застосовують OFI, використовувати libibverbs для частини операцій.

Розгляд цих низькорівневих API виходить за рамки цієї статті, оскільки вони безпосередньо не використовуються в програмах, що розпаралелюються. Однак їх застосування важливо для отримання високої продуктивності MPI, і конкретна реалізація MPI може забезпечувати можливість використання різних таких API. Наприклад, Intel MPI працює з різними низькорівневими API.

Потенційно важливим для досягнення високої продуктивності MPI при паралелізації на спільній пам'яті є використання XPMEM, модуля ядра Linux, який дозволяє процесу відображати пам'ять іншого процесу у віртуальному адресному просторі. Однак розробники XPMEM вказують на можливі помилки в останній версії, тому на даний момент XPMEM можна вважати недостатньо стабільним. Тим не менш, робота з XPMEM підтримується у всіх трьох реалізаціях MPI, що розглядаються, і в цілому його слід вважати актуальним для використання з MPI в найближчому майбутньому.

Продуктивність MPI залежить від багатьох факторів: використовувані міжз'єднання або загальна пам'ять, версії MPI-реалізації, низькорівневий API, тестова система та конкретні MPI-операції. Ефективне рішення для оптимізації продуктивності може ґрунтуватися на механізмах вибору конкретної реалізації, що динамічно змінюється.

Всі три аналізовані реалізації MPI підтримують стандарт MPI 3.1 і роботу з усіма згаданими RDMA зі швидкістю 100 Гбіт/с, а також роботу з DAPL, OFED verbs, UCX та XPMOD. Слід зазначити, що нові версії реалізації MPI з'являються дуже швидко, і дані про продуктивність можуть швидко застаріти.

Поточна версія OpenMPI використовує засоби libfabric. Як і раніше вона поставляється у вигляді відкритого вихідного коду, що є перевагою, оскільки дозволяє реалізувати OpenMPI на різних операційних системах, включаючи різні дистрибутиви Linux, адаптуючи бібліотеки під різні компілятори. Для дистрибутивів доступні двійкові пакети та засоби для роботи з Python. OpenMPI включає також засоби, що належать до PGAS. Як зазначено, IBM Spectrum MPI базується на OpenMPI. У поточній версії є підтримка роботи на нових процесорах з архітектурою, яка відрізняється від x86-64.

Інший вільно доступною реалізацією MPI є MVAPICH, в даний час представлена сімейством MVAPICH2. Як низькорівневий API MVAPICH2 використовує libibverbs. MVAPICH2 відрізняється швидким випуском нових версій та підтримкою різних варіантів бібліотек, оптимізованих для різних цілей. Наприклад, MVAPICH2-GDR оптимізована для роботи з прискорювачами Nvidia, а спеціальний варіант MVAPICH2-X, який раніше орієнтувався на Infiniband, зараз працює зі всіма згаданими RDMA-міжз'єднаннями та підтримує PGAS.

Intel MPI є комерційною реалізацією MPI, що традиційно означає вищий рівень продуктивності порівняно з MVAPICH2 та OpenMPI. В якості низькорівневого API останні версії Intel MPI орієнтуються на libfabric.

Тести продуктивності під час використання MPI

Існує безліч різних тестів продуктивності, які застосовуються для оцінки ефективності використання MPI. Тести SPEC MPI 2007 надають переважно інформацію про продуктивність обчислювальних систем, а не про швидкодію MPI-комунікацій. Такі тести корисні для оцінки масштабованості продуктивності зі збільшенням кількості використовуваних процесів або процесорних ядер. SPEC MPI можна використовувати для оцінки ефективності різних реалізацій MPI та його версій на однакових обчислювальних системах.

Аналіз продуктивності MPI-комунікацій проводять за допомогою мікротестів, які вимірюють затримки та пропускну здатність при різних

операціях MPI. Найбільш широко застосовуються тести OMB (Ohio State University Micro Benchmarks) та IMB (Intel Micro Benchmark). У наборі тестів OMB вимірюються затримка та пропускна здатність для точкових, односторонніх та колективних операцій MPI у широкому діапазоні розмірів повідомлень.

У контексті цього огляду зосередимося на сучасних даних про продуктивність MPI, що використовують найпоширеніші високошвидкісні міжз'єднання з пропускною здатністю 100 Гбіт/с: Infiniband EDR, Intel Omni-Path та Ethernet RoCE/iWARP.

Дані мікротестів MPI дають змогу оцінити характеристики комунікацій, включаючи показники самих міжз'єднань на реальному програмному рівні. Як зазначалося, оцінки залежить від безлічі чинників. Ці оцінки залежать не тільки від характеристик самих міжз'єднань, але і від процесорів, процесорних ядер і оперативної пам'яті, що використовуються. Однак основний вплив на результати має вибір міжз'єднання.

Як зазначалося, результати мікротестів продуктивності MPI залежать як від характеристик міжз'єднання, а й від продуктивності процесорних ядер та оперативної пам'яті. Більше того, на затримки під час роботи з NUMA-серверами може значно впливати розташування мережної плати міжз'єднання. Така плата може знаходитися на PCIe-шині процесора, що виконує MPI-операцію, або процесора сусіднього сокету материнської плати. Тому точне зіставлення продуктивності міжз'єднань або реалізацій MPI можливе лише при порівнянні даних тестів, виконаних на одній і тій же обчислювальній системі.

Різні міжз'єднання або реалізації MPI можуть показувати високу продуктивність на одних тестах і поступатися іншими. Крім того, реалізації MPI мають багато параметрів, які можуть впливати на продуктивність. Швидка поява нових версій реалізацій MPI сприяє швидкому старінню даних щодо їх відносної продуктивності.

Наприклад, зазвичай вважається, що комерційна реалізація Intel MPI демонструє більш високу продуктивність, ніж OpenMPI та MVAPICH2. Однак є дані для нових версій OpenMPI та MVAPICH2, де на деяких тестах вони показують більш високі результати. Наприклад, вважається, що OpenMPI традиційно є високопродуктивною реалізацією, а MVAPICH зазвичай не перевищує OpenMPI більшості додатків. У той же час є дослідження, де демонструється вища продуктивність MVAPICH2. Результати тестів продуктивності MPI корисні також для оцінки продуктивності нових апаратних засобів міжз'єднань та ефективного написання HPC-програм.

Почнемо розгляд даних про продуктивність MPI зі порівняння міжз'єднань Mellanox Infiniband EDR та Intel Omni-Path. Ці дані отримані реалізації MVAPICH2-2.3. Затримка при односторонніх MPI-операціях для

невеликих повідомлень Omni-Path іноді перевищує EDR. Для великих повідомлень іноді EDR буває швидше Omni-Path. Однак частіше показники продуктивності MPI для EDR та Omni-Path близькі. В інших реалізаціях MPI ці співвідношення можуть відрізнятись.

Як зазначалося раніше, продуктивність MPI може істотно залежати від використовуваних реалізаціями програмних засобів MPI, включаючи низькорівневі API. Застосування UCX дозволило підвищити пропускну здатність OpenMPI під час роботи з EDR. Ще одним важливим засобом підвищення продуктивності MPI є XPMEM. Застосування XPMEM дозволяє зменшити затримки на тестах OMB для різних розмірів повідомлень, тому MVAPICH2-2.3 починає перевершувати Intel MPI 2017, хоча без XPMEM він поступався Intel-реалізації.

При роботі з RoCE на мікротестах IMB застосування Intel MPI для колективних MPI-операцій демонструє значно більшу продуктивність (нижчі затримки), ніж тести OMB з використанням MVAPICH2. Однак при операціях між двома процесами значно більша пропускна здатність досягається в тестах OMB на базі MVAPICH2. В даний час актуальність цих даних знизилася, оскільки версії обох реалізацій MPI були суттєво оновлені.

Наведені дані тестів продуктивності показують, що найбільш правильним підходом є тестування продуктивності з використанням конкретної програми, необхідної користувачеві, як зазначено у доповіді на офіційному сайті Intel.

Висновки. У цій роботі проведено порівняльний аналіз сучасних ефективних та широко використовуваних реалізацій засобів паралелізації MPI у моделі розподіленої пам'яті. Дані про їх продуктивність не дозволяють зробити однозначний вибір на користь будь-якої реалізації.

Оцінюючи продуктивність HPC-додатків доцільно відбирати дані для конкретних додатків, а не покладатися на "інтегральні" дані SPEC MPI 2007.

Література

1. Dimakopoulos V. V. Parallel programming models // Smart Multicore Embedded Systems. Springer New York, 2014. P. 3—20.

2. Abdulbaqi J. Programming at Exascale: Challenges and Innovations. arXiv preprint arXiv:1809.10023. 2018, URL: https://arxiv.org/pdf/1809.10023.pdf.

3. The OpenMP API specification for parallel programming. URL: https://www.openmp.org.

4. MPI: A Message-Passing Interface Standard. Version 3.1. Message Passing Interface Forum. June 4, 2015. URL: https://www. mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf.
5. Chai L., Noronha R., Panda D. K. MPI over uDAPL: Can High Performance and Portability Exist Across Architectures? // Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06), IEEE, 2006. P. 19 - 26.
6. Libfabric Programmer's Manual. URL: http://www.libfabric.org.

References

1. Dimakopoulos V. V. Parallel programming models // Smart Multicore Embedded Systems. Springer New York, 2014. P. 3—20.
2. Abdulbaqi J. Programming at Exascale: Challenges and Innovations. arXiv preprint arXiv:1809.10023. 2018, URL: https://arxiv.org/pdf/1809.10023.pdf.
3. The OpenMP API specification for parallel programming. URL: https://www.openmp.org.
4. MPI: A Message-Passing Interface Standard. Version 3.1. Message Passing Interface Forum. June 4, 2015. URL: https://www. mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf.
5. Chai L., Noronha R., Panda D. K. MPI over uDAPL: Can High Performance and Portability Exist Across Architectures? // Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06), IEEE, 2006. P. 19—26.
6. Libfabric Programmer's Manual. URL: http://www.libfabric.org.