Е. О. Тютляева

Тестирование корректности и производительности реализации MPI

Научный руководитель: к.х.н. А. А. Московский

Аннотация. В статье описывается разработанная система тестов для MPI. Данная система ориентирована на тестирование коммуникационной библиотеки, базирующейся на стандарте MPI-2. Статья включает сравнительный анализ систем тестирования MPI и систем тестирования общего назначения.

Ключевые слова и фразы: тестирование, MPI, коммуникационные библиотеки, системы тестирования.

1. Введение

Важной частью хорошей реализации MPI [1] является полное и тщательное тестирование как корректности реализации, так и производительности полученной системы. В настоящее время существует достаточное количество различных наборов тестов для различных реализаций MPI.

В данной статье описывается проведенная работа по реализации системы тестирования MPI для тестирования корректности и производительности на всех этапах разрабатываемой коммуникационной библиотеки SKIF–MPI, удовлетворяющей международному стандарту MPI-2 и использующей для коммуникации аппаратные средства узлов суперкомпьютеров семейства «СКИФ» ряда 4 — коммуникационную сеть с топологией трехмерного тора (3D–тор) на базе программируемых логических интегральных схем (ПЛИС).

[1] MPI (Message Passing Interface) — программный интерфейс для передачи информации, который позволяет обмениваться сообщениями между процессами, выполняющими одну задачу.

В рамках данной работы был проведен обзор и анализ существующих наборов тестов различных реализаций MPI, а также существующих систем тестирования программного обеспечения для реализации удобной библиотеки, которая будет позволять осуществлять тестирование на всех этапах разработки новой коммуникационной библиотеки.

В статье решаются вопросы выбора тестов MPI из существующего разнообразия вариантов, формулируются те аспекты системы тестирования, которые нельзя упускать из виду при разработке, освещаются начальные подходы к визуализации результатов.

2. Системы тестирования

2.1. Системы тестирования программного обеспечения

Выбор или разработка удобной и эффективной системы тестирования является нетривиальной задачей, с которой, тем не менее, приходилось сталкиваться всем разработчикам программного обеспечения. Существует множество решений, в том числе и на рынке открытого (Open Source) программного обеспечения. В рамках данного проекта были рассмотрены некоторые из существующих систем тестирования для построения основных концепций дизайна и содержания разрабатываемой системы тестирования.

2.1.1. *Automated Testing Framework*

Automated Testing Framework [1] — это набор библиотек и утилит для автоматического тестирования. Свободная (распространяется под TNF лицензией) система, предоставляющая возможности автоматического тестирования.

Легко позволяет запускать несколько тестов за раз, объединять их в группы и представляет результат в удобном формате. В частности, тесты пишутся с использованием C++ или POSIX shell binding. Каждый тестовый файл содержит необходимые разделы. В описании самого теста можно использовать C++/Shell и специализированные команды системы ATF. Затем тесты компилируются и создается конфигурационный файл Atffile, в котором объявляются тестовые секции, в которых перечисляются тесты. Соответственно, можно создать несколько таких файлов для различных «областей применимости» с

различными наборами тестов. Отчет о тестировании в удобном формате показывает, какие тесты passed/skipped/failed, статистику. Подробнее можно посмотреть при помощи специальной утилиты.

Утверждается, что работает на большинстве операционных систем семейства Unix.

Предоставляет возможность легко реализовывать и запускать тесты, но это накладывает некоторые ограничения на то, что конкретно делают тесты. Практически, пишутся тесты, результаты которых обрабатываются в соответствии с условиями и выдается, прошел тест, или нет. Весь IO теста можно опционально сохранять.

2.1.2. *Autotest*

Autotest [2] — это рабочая среда для полностью автоматизированного тестирования. Она разработана, в первую очередь, для тестирования ядра Linux, но может быть применена и в прочих ситуациях. Распространяется под лицензией GPL и используется во многих организациях, включая google, GPL и проч. Среди основных задач тестирования разработчики называют:

- Следует всегда запускать тесты при аналогичных условиях.
- Необходимо, чтобы знание о том, как провести тестирование, не принадлежало только одному человеку, а было доступно всем разработчикам.
- Должен быть простой способ распространить тесты среди широкого круга пользователей.
- Тесты должны быть легко воспроизводимы.

2.2. Средства тестирования MPI

Кроме систем тестирования общего назначения, существуют как системы тестирования, так и пакеты тестов нацеленные на тестирование реализаций MPI (порой — конкретных реализаций). Обзор некоторых из них интересен как для исследования существующих тестов, так и для включения некоторых из них целиком или частично в разрабатываемый набор тестов.

2.2.1. *MPI Testing Tool*

MPI Testing Tool [3] — это комплексная инфраструктура для тестирования реализаций MPI и запуска тестов производительности в

полностью автоматической манере. Потенциально MTT является системой тестирования распределенной между различными кластерами/средами и собирающей все результаты в центральную базу данных. Тестируются следующие аспекты реализации MPI:

(1) Успешно ли установлен MPI
(2) Могут ли быть успешно скомпилированы тестовые программы при помощи данной реализации MPI
(3) Могут ли тестовые программы быть успешно запущены и завершены с удовлетворительным результатом производительности.

Данная система тестирования реализована полностью на Perl, относится к открытому программному обеспечению.

2.2.2. *Intel MPI Benchmarks*

Intel MPI Benchmarks [4] — набор тестов, ранее известный как Pallas MPI Benchmark. Основные задачи этого пакета тестов:

- Предоставление краткого набора тестов, нацеленных на измерение наиболее важных MPI функций.
- Предоставить точную методологию измерения производительности
- Не предоставлять интерпретацию полученных результатов: просто выдать отчет о времени.

Intel MPI Benchmarks предоставляет качественно структурированные результаты производительности наиболее важных коллективных и двухточечных операций. Кроме того, он входит в программу Intel Cluster Ready, поэтому очень важно включить данный пакет в состав тестирования.

2.2.3. *Система тестов производительности для параллельных компьютеров*

Система тестов производительности для параллельных компьютеров (mpi-bench-suite) [5], разработана в НИВЦ МГУ. Все тесты распространяются свободно в виде исходных текстов на языке Си. Тесты для своей работы требуют наличия реализации интерфейса MPI (например, MPICH). Разработанный пакет включает четыре теста, три из которых тестируют эффективность среды (сети) передачи данных между процессорами (узлами кластера), а четвертый тестирует производительность совместного доступа узлов к сетевому файл–серверу.

(1) transfer — тест латентности и скорости пересылок между двумя узлами;
(2) nettest — тест пропускной способности сети при сложных обменах по различным логическим топологиям;
(3) mpitest — тест эффективности основных операций MPI;
(4) nfstest — тест производительности файл–сервера.

2.2.4. *NAS Parallel Benchmarks*

NAS Parallel Benchmarks [6] — это небольшой набор программ, разработанный для вычисления производительности параллельных суперкомпьютеров. Эти задачи пришли из реальных приложений вычислительной гидродинамики. Они разрабатываются и поддерживаются NASA Advanced Supercomputing (NAS) Division базирующихся на NASA Ames Research Center. Среди этих тестов — приближенное решение трехмерного уравнения Пуассона (mg), оценка наибольшего собственного значения симметричной разреженной матрицы (cg), параллельная сортировка N целых чисел без использования операций с плавающей точкой (is), генерация пар случайных чисел Гаусса (ep) и т.п.

2.2.5. *Прочее*

Существует значительное количество пакетов тестов, сравнительный анализ которых не входит в рамки данной статьи. В частности, это наборы тестов для различных реализаций Intel, ANL, IBM и т.п., наборы тестов для проверки определенных возможностей MPI и измерения определенных классов функций. Таким образом, очевидно, что включение всех существующих в открытом доступе тестов реализаций MPI в систему будет избыточным; тем не менее, ряд тестов стоит включить вопреки сходству концепций, в связи с тем, что данные тесты позволяют оценить и сравнить новую реализацию с уже существующими и рассчитывать на какую-то определенную позицию на рынке суперкомпьютеров.

3. Базовые концепции построения системы тестирования

Таким образом, были изучены существующие системы тестирования. Задача тестирования реализации MPI имеет свою специфику, которая отличает ее от тестирования иного программного обеспечения, к примеру, модулей ядра linux, что не позволяет воспользоваться

полностью готовым продуктом. Однако некоторые концепции дизайна систем тестирования не следует упускать из виду при разработке. На основании изученных систем тестирования были сформулированы некоторые идеи, которые были заложены в основу построения системы тестирования.

- При тестировании реализации MPI следует уделять внимание не только тестам производительности, но и, прежде всего, тестированию корректности.
- В связи с тем, что всестороннее тестирование производительности может занимать достаточное количество времени, необходимо предоставить пользователю возможность проводить как ночные стресс–тесты, так и повседневные тесты корректности и производительности, позволяющие быстро выявить явные недочеты и отклонения от нормы.
- Тем не менее, одновременно с гибкостью настроек, необходимо сохранить возможность проведения абсолютно идентичных тестов (как по набору тестирующих программ, по параметрам, так и по общему состоянию системы). Необходимо, чтобы воспроизвести тест с заданными параметрами мог провести не только автор данных настроек, но и любой человек, участвующий в процессе разработки/тестирования и имеющий доступ к отчетам системы тестирования.
- Система должна обладать возможностью представлять базовую часть данных графически, чтобы позволять проводить визуальный сравнительный анализ тестируемых реализаций MPI.
- Полученная в результате тестирования информация должны быть качественно структурирована для облегчения задачи дальнейшего анализа.

4. Реализованная система тестирования

4.1. Конфигурация

Важно отметить, что реализация MPI, на которую ориентирована описываемая система тестирования, разрабатывается на базе MPICH2. В связи с этим, перед проведением тестирования производительности рекомендуется провести базовый тест MPICH2 [2], как основной тест корректности. Базовые тесты включены в реализацию mpich2 и могут быть выполнены при использовании команды make

[2] http://wiki.mcs.anl.gov/mpich2/index.php/Testing_MPICH2

testing в корневом каталоге реализации. Результаты теста будут сохранены в файле summary.xml в поддиректории test. Данное тестирование позволяет выявить ошибки и явные недостатки производительности, которые отражаются на результатах некоторых тестов как MPIEXEC_TIMEOUT. Структура данного набора тестов позволяет легко локализовать проблему — достаточно выяснить, в каком из элементарных тестов она возникла.

Если в результате выполнения данного набора тестов все компоненты теста завершились успешно, имеет смысл приступать к тестированию производительности. Реализованная система тестирования производительности включает в себя следующие тесты:

- Intel MPI Benchmarks
- NAS Parallel Benchmarks
- Система тестов производительности для параллельных компьютеров (за исключением теста производительности файл–сервера)

Для того, чтобы предоставить пользователю возможность гибких настроек, система использует конфигурационные файлы, в которых пользователь может указывать, какие тесты нужно проводить, сколько раз и для каких параметров, а также оставлять комментарии. Ниже приведен пример такого конфигурационного файла, который расположен в корневом каталоге системы тестирования.

```
#imb - Intel MPI Benchmarks 3.2
#Для того, чтобы был проведен тест imb для n процессов:
#imb N, где N - число процессов

imb 2
imb 4

#параметры для следующих 3-х тестов можно посмотреть на parallel.ru
#transfer test: В данном случае, в качестве единицы длины сообщения
# выбирается Кбайт (uK), длина сообщений увеличивается в
# геометрической прогрессии (K2) от 1 Кбайта (m1) до 1 Мбайта (M1024).
# По умолчанию, каждая тестовая процедура будет повторяться 10 раз
# (T10), но это количество будет увеличено, если общее время меньше,
# чем 0.1 сек (f100). Полностью тест будет повторен 2 раза (R2).

#Последняя цифра - количество процессов при запуске mpi
transf2 uK m1 M1024 K2 T10 f100 R2 2
#проведем еще 1 transfer-тест с такими же параметрами но на четырех
# процессах
transf2 uK m1 M1024 K2 T10 f100 R2 4
```

```
#Далее - nettest из того же пакета.
nettest uK m1 M16 K2 R2 T1000 Ptotal 4

#MPITEST из того же пакета на 2-х процессорах
mpitest m1 M100 K10 T100 t100000 R2 2
# Для тестов NAS необходимо указывать класс сложности и количество
# узлов
lu S 1
# Число процессов должно быть квадратом (1,4,9,...) для теста bt
bt S 4
cg S 1
dt S 1
ep S 1
ft S 1
is S 1
mg S 1
sp S 1
```

Тесты NAS компилируются специально для каждого класса задач („S”, „W”, „A”, „B”, „C”, „D”, или „E”) и каждого количества процессоров, поэтому проводить прекомпиляцию для этих тестов не рекомендуется. Классы существенно различаются сложностью, требованиями к ресурсам, к примеру оперативной памяти и жесткому диску. Приведенный в примере класс S является самым простым классом и прежде всего проверяет корректность выполнения. После того, как в конфигурационном файле пользователь задает интересующий его класс и количество процессов для каждой задачи, в процессе тестирования необходимые тесты будут автоматически скомпилированы и запущены с заданными параметрами.

После того, как пользователь создал/отредактировал/принял решение использовать стандартный конфигурационный файл следует запустить систему тестирования путем запуска тестирующего скрипта на языке Perl.

4.2. Тестирование и результаты

После того, как тестирование было запущено с использованием пользовательского конфигурационного файла, система попросит ввести комментарии к запуску, причину запуска и имя пользователя, который запустил тестирование. Эта информация может пригодиться при дальнейшем анализе результатов тестирования.

Из отчета системы тестирования, видно, какие тесты прошли успешно. Результаты тестирования будут сохранены в директорию,

имя которой имеет вид yyyymmdd_hhmm, где yyyy — год, mm — месяц, dd — день, hh — час, mm — минуты запуска теста. Предполагается, что тестирование занимает больше минуты, так что имена директорий будут уникальны.

В данной директории будут сохранены результаты всех тестов в файлах, имена которых будут состоять из названия теста, количества процессов, на которых тест был запущен и названии класса для тестов NAS.

В этой же директории будет сохранен конфигурационный файл тестирования, что позволяет воспроизвести тестирование с заданными параметрами любому человеку, участвующему в проекте и провести сравнение полученных результатов. В файле logfile.log записаны примечания к тестированию и запущенные команды.

5. Построение сравнительных графиков

Для получения наглядных результатов на всех длинах следует строить следующие графики:

- Тесты на пропускную способность (в основном 2x узловые: Ping–pong, ping–ping, ...). Два графика:
  - Время выполнения (линейный масштаб) от размера сообщений (логарифмический масштаб) только для маленьких длин.
  - Пропускная способность (линейный масштаб) от размера сообщений (логарифмический масштаб) для всех длин.
- Тесты коллективных операций. Два графика.
  - Время выполнения (линейный масштаб) от размера сообщений (логарифмический масштаб) только для маленьких длин.
  - Время выполнения (линейный масштаб) от размера сообщений (логарифмический масштаб) для всех длин.

Такая система позволяет наглядно оценить результаты на всех длинах как для коллективных, так и для точечных операций. В качестве примера на рисунках ниже приведены графики для маленьких длин 1 и для всех длин 2, визуализирующие результаты теста SendRecv из набора Intel MPI Benchmarks для mpi-2 сконфигурированного с использованием виртуальных устройств ch3:nemesis и ch3:shm.

Для построения графиков использовалась система Gnuplot [7] — качественная система построения научных графиков, запускаемая из командной строки, свободно распространяемая.

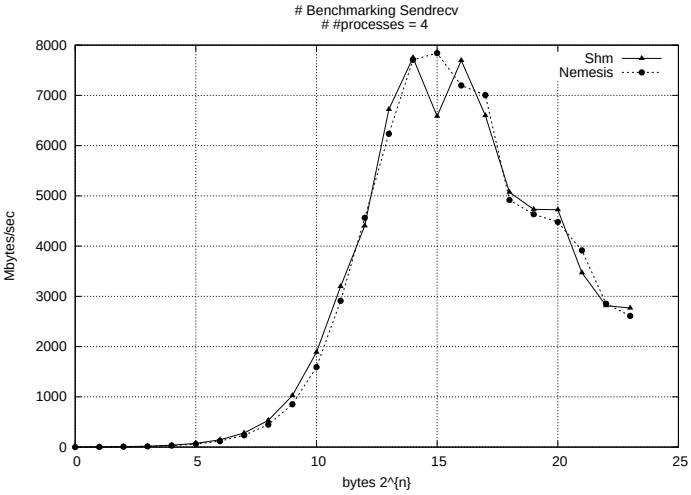

Рис. 1. Тест Send-Receive для маленьких длин

На языке Perl был написан скрипт, который позволяет автоматически строить графики по сравнительным результатам всех тестов из набора Intel MPI Benchmarks, выполненных с одинаковым количеством процессов. Таким образом, при проведении тестирования графики не строятся, что позволяет компактно хранить данные. Тем не менее, при необходимости анализа данных при помощи данного модуля (Do_graphs.pl) можно легко получить визуализацию результатов тестов imb.

6. Заключение

Был проведен обзор систем тестирования и существующих тестовых пакетов для оценки производительности MPI (в частности, mpich2). На основании изученных данных была реализована первая версия системы тестирования, объединяющая Intel MPI Benchmarks,

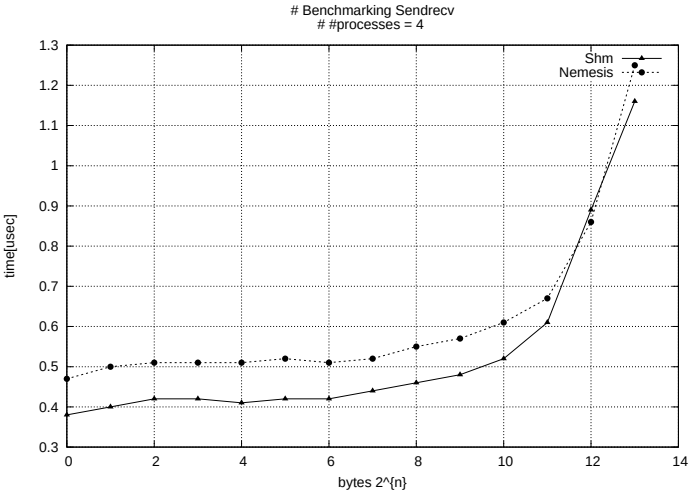

Рис. 2. Тест Send-Receive для всех длин

NAS Parallel Benchmarks и тесты производительности MPI, разработанные в НИВЦ МГУ, позволяющая проводить тестирование и сохранять результаты в удобном для последующей обработке формате. Выбранный набор тестов позволяет отследить большое количество «узких» мест в тестируемой реализации MPI. С использованием реализованной системы тестирования и построения графиков были проведены работы по сравнительному тестированию реализации MPICH2 (актуальная версия — 1.2), сконфигурированному с использованием виртуальных устройств ch3:nemesis и ch3:shm соответственно. Тестирование проводилось на опытном образце «СКИФ–4–Ботик». Описанная система используется для тестирования корректности и производительности промежуточных версий разрабатываемой коммуникационной библиотеки SKIF–MPI, удовлетворяющей международному стандарту MPI-2 и использующей для коммуникации аппаратные средства узлов суперкомпьютеров семейства «СКИФ» ряда 4 — коммуникационную сеть с топологией трехмерного тора (3D–тор) на базе программируемых логических интегральных схем (ПЛИС).

7. Перспективы

Планируется вести доработку системы тестирования в соответствии с процессом разработки коммуникационной библиотеки SKIF–MPI. Необходимо протестировать все возможные «узкие» места реализуемого программного продукта, чтобы полученная коммуникационная библиотека удовлетворяла самым высоким стандартам качества и производительности. Для достижения поставленных целей следует повысить степень автоматизации модуля построения графиков и расширить его возможности для более подробной и качественной визуализации полученных результатов. Планируется также расширение набора тестов, включенных в систему. К примеру, интерес представляет тест b_eff, недавно включенный в программу Intel Cluster Ready. Также проводятся работы по поиску стресс–тестов на корректность реализации — к примеру, тест, который сутки работает с одними и теми же процессами.

Список литературы

[1] The Automated Testing Framework Home Page: The NetBSD Foundation, `http://www.netbsd.org/~jmmv/atf/`.

[2] The Autotest Home Wiki Page, `http://autotest.kernel.org/`.

[3] MPI Testing Tool (MTT) Home Page: The Open MPI Project, `http://www.open-mpi.org/projects/mtt/`.

[4] Intel MPI Benchmarks 3.2: Intel Corporation, `http://www.intel.com/software/imb/`.

[5] Система тестов производительности для параллельных компьютеров: Лаборатория параллельных информационных технологий НИВЦ МГУ, `http://parallel.ru/testmpi/`.

[6] NAS Parallel Benchmarks Homepage, `http://www.nas.nasa.gov/Resources/Software/npb.html`.

[7] gnuplot homepage, `http://www.gnuplot.info/`.

Институт программных систем РАН, аспирантура

E. O. Tutlyaeva. *Correctness and performance testing of MPI realization* // Proceedings of Junior research and development conference of Ailamazyan Pereslavl university. — Pereslavl, 2010. — p. 5–16. (*in Russian*).

Abstract. This paper aimed at description of realized MPI test suite. The suite targeted on testing MPI-2 based communication library. The paper includes some review of freely available MPI test suites and general purpose testing system.

Key Words and Phrases: testing, MPI, communication library, testing systems .