Программное обеспечение системы с глобально адресуемой памятью

Руководство пользователя

Версия 1.1 от 28 ноября 2016 г.

СПбПУ, СКЦ «Политехнический»

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
	28 ноября 2016 г.			28 ноября 2016 г.

Аннотация

Настоящий документ содержит руководство пользователя программного обеспечения «Системы с глобально адресуемой памятью» версии 1.1 от 28 ноября 2016 г. и включает 44 стр., 1 рис., 2 табл.

Содержание

История версий электронной документации

Версия	**Дата**	**Автор**	**Изменения**
1.1	28 ноября 2016 г.	expx	Добавлено описание основного ПО

Таблица 1 — История версий документации

Обозначения и сокращения

В настоящем документе применены следующие обозначения и сокращения с соответствующими определениями.

макроузел: объединение нескольких узлов вычислительного комплекса в единый узел с общей памятью посредством ведущего узла и адаптера IB FDR 56 Gb/s;

мастер-узел: основной узел макроузла, на котором установлен экземпляр операционной системы;

byte transfer layer; BTL: фрейморк OpenMPI, используемый для конкретной задачи — передачи данных посредством сети NumaConnect [1];

cache coherent non-uniform memory access; ccNUMA: архитектура с аппаратной поддержкой когерентности кэшей и неоднородного доступа к памяти;

thread local storage; TLS: локальное хранилище потока.

Примеры стилей

В настоящем документе применены стили с соответствующими определениями.

№	Пример	Определение
1	`$ ls`	команда, выполняемая с правами обычного пользователя
2	`# ls`	команда, выпоняемая с правами суперпользователя
3	`OMPI_PREFIX_ENV=/dev/shm`	переменная окружения

Таблица 1 — Примеры стилей

Глава 1. Общие сведения

Наименование вычислительного комплекса: *система с глобальнo адресуемой памятью* (далее — *комплекс*, *система*).

Полное наименование ПО: программное обеспечение системы с глобально адресуемой памятью.

Назначение: программное обеспечение предназначено для компиляции и запуска вычислительных приложений в системе с глобально адресуемой памятью.

Система с глобально адресуемой памятью предназначена для запуска приложений, требующих бо̀льшего количества процессорных устройств и оперативной памяти, чем доступно на кластере со стандартной архитектурой.

Глава 2. Аппаратное обеспечение

2.1. Основные характеристики вычислительного комплекса

Вычислительный комплекс имеет следующие общие характеристики:

- архитектура — ccNUMA.
- 64 базовых узла;
- 3072 ядра;
- 12 288 Gb общей памяти;
- пиковая производительность составляет 31 Tf;

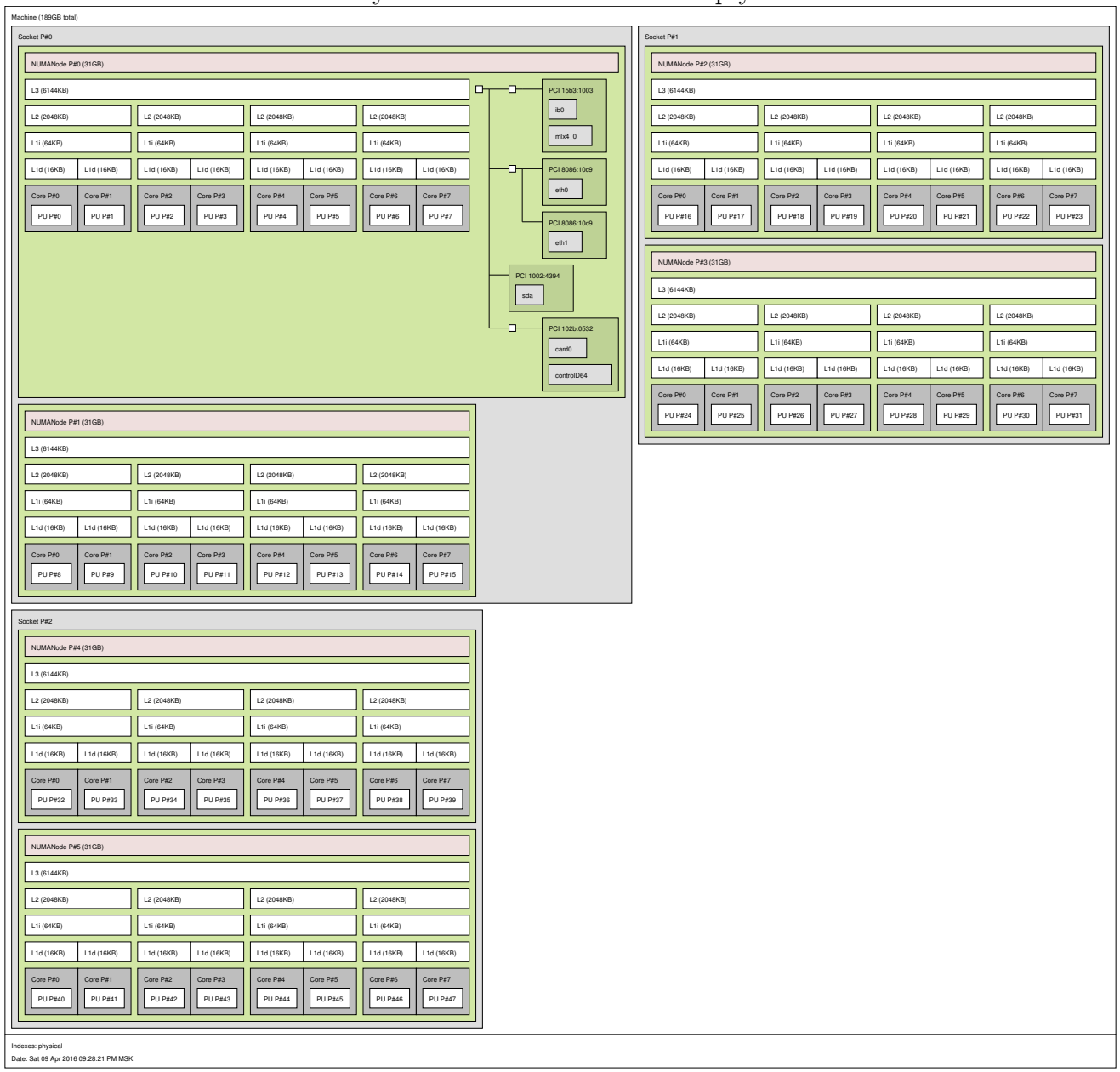
2.2. Вычислительные компоненты

Ведущий узел (синоним — *мастер-узел*) имеет следующие характеристики:

- платформа SuperMicro;
- CPU AMD Opteron 6380 (16 ядер, 2.5 ГГц) (× 3);
- 192 Gb DDR3-1600;
- адаптер NumaConnect, 19.2 Gb/s;
- адаптер IB FDR 56 Gb/s;

Топология ведущего узла представлена на рисунке 2.1.

Рисунок 2.1 — Топология мастер-узла



2.3. Сетевые компоненты

Сетевые компоненты включают в себя:

- сеть NumaConnect с топологией типа «трехмерный тор»;
- 16 узлов имеют адаптер IB FDR 56 Gb/s и могут быть ведущими в системе, образуя «макроузлы»;

Глава 3. Программное обеспечение

3.1. Компиляторы

3.1.1. Использование GCC

Доступно несколько версий (сборок) `gcc`, выбор которых осуществляется посредством загрузки соответствующего доступного модуля (см. раздел 3.2.1 — `environment-modules`).

Оптимизированная (рекомендуемая) версия

Оптимизированная версия отличается от стандартной применением патча библиотеки `libgomp` с использованием `numa_alloc_onnode()` из библиотеки `libnuma`. Патч обеспечивает локализацию стека и локального хранилища потока (TLS) для запущенных процессов. Оптимизированная версия поддерживает более 1024 потоков. Оптимизированная версия собрана на основе новейшей на данный момент версии компилятора, для которой доступен патч.

При необходимости использования OpenMPI для компиляции приложения, должен быть загружен соответствующий компилятор, с помощью которого библиотека была собрана:

```
$ module load numa/compilers/gcc/5.3.0 numa/ompi/1.10_gcc-5.3.0
```

Стандартная версия

При отсутствии загруженных модулей окружения будет доступен стабильный gcc, поставляемый с CentOS7.

Рекомендуемые параметры компиляции

- `-fopt-info-vec-missed` — для вывода информации о пропущенных возможностях векторизации[1];
- `-fprefetch-loop-arrays` — для повышения производительности циклов, которые имеют доступ к большим массивам. Генерация инструкций для предварительной выборки памяти поддерживается системой с глобально адресуемой памятью[2];
- используемые константы для управления объемами оптимизации:
  - `-param prefetch-latency=416` — для определения среднего числа инструкций, которые выполняются до окончания предварительной выборки. Увеличение данного числа может привести к снижению потоков упреждающей выборки[3];
  - `-param simultaneous-prefetches=16` — максимальное значение предварительной выборки[4];
- прочие параметры:
  - `-Ofast` — для включения всех оптимизаций `-O3`, а так же будет включен параметр `-ffast-math`;
  - `-march=bdver2` — компилятор определит тип процессора посредством чтения `\proc\cpuinfo` (AMD Opteron Processor 6380), будет геререровать инструкции и планировать их соответствующим образом;
  - `-flto` — для стандартной оптимизации во время линковки.

[1] Синоним `-fopt-info-missed-vec`; ранее `-ftree-vectorizer-verbose=2` (показывает локации, которые не могут быть векторизованы, а так же причины этого).

[2] недоступно при использовании опции `-Os`

[3] значение константы подбирается эмпирическим путем

[4] значение константы подбирается эмпирическим путем

3.2. Утилиты

3.2.1. `environment-modules`

Функциональное назначение

Пакет *Environment Modules* применяется для динамической модификации окружения пользователя с помощью файлов модулей. Модули используются для управлениями различными версиями (сборками) одного и того же приложения.

Каждый файл модуля содержит информацию, необходимую для настройки оболочки для приложения. После инициализации пакета `environment-modules` окружение может быть модифицировано на основе каждого модуля с использованием команды `module`, которая интерпретирует файлы модулей. Модули могут использоваться совместно всеми пользователями системы.

Условия применения

Модули могут быть загружены и выгружены динамически, поддерживаются все популярные оболочки. Пример переменных окружения, которые могут быть установлены с помощью модуля, представлен в листинге 3.1.

```
1  #%Module -*- tcl -*-
2  #
3  # GCC environment module
4  proc ModulesHelp { } {
5          global version modroot
6          puts stderr "sets the Environment for GCC"
7  }
8
9  module-whatis "Sets the environment for using GCC compilers (C, Fortran)"
10
11 set    version      5.3.0
12 set    prefix       /nfs/opt/software/compilers/gcc/${version}
13 set    sys      linux86
14
15 setenv     CC       ${prefix}/bin/gcc
16 setenv     GCC      ${prefix}/bin/gcc
17 setenv     FC       ${prefix}/bin/gfortran
18 setenv     F77      ${prefix}/bin/gfortran
19 setenv     F90      ${prefix}/bin/gfortran
20 prepend-path  PATH       ${prefix}/include
21 prepend-path  PATH       ${prefix}/bin
22 prepend-path  MANPATH      ${prefix}/man
23 prepend-path  LD_LIBRARY_PATH   ${prefix}/lib
24 prepend-path  LD_LIBRARY_PATH   ${prefix}/lib64
```

Листинг 3.1 — Пример файла модуля для GCC (версии 5.3.0)

Вывести список доступных модулей

```
$ module list
```

Вывод

```
1
2 ---------------------- /usr/share/Modules/modulefiles -----------------------
3 dot         module-git  module-info modules     null        use.own
4 ----------------------------- /nfs/opt/modules ------------------------------
5 numa/compilers/gcc/4.8.5        numa/ompi/1.10_gcc-4.8.5
6 numa/compilers/gcc/5.3.0        numa/ompi/1.10_gcc-5.3.0
7 numa/hwloc
```

Вывести список загруженных модулей

```
$ module list
```

Вывод

```
1
2 Currently Loaded Modulefiles:
3   1) numa/compilers/gcc/4.8.5   3) numa/ompi/1.10_gcc-5.3.0
4   2) numa/lib/acml/5.3.1        4) numa/hwloc
```

Выгрузить один модуль и загрузить другой

```
1
2 $ module unload numa/compilers/gcc/4.8.5 && module load numa/compilers/gcc/5.3.0
```

Как уже было сказано в разделе 3.1.1 — «Использование GCC», при использовании некоторых библиотек (например, OpenMPI) важно получить описание модулей и убедиться, что библиотека была собрана с использованием соответствующей версии компилятора:

Вывести описание модуля

```
$ module whatis numa/compilers/gcc/5.3.0
```

Вывод

```
1
2 numa/compilers/gcc/5.3.0: Sets the environment for using GCC compilers (C,
3 Fortran)
```

3.2.2. numactl

Функциональное назначение

Утилита `numactl` позволяет запускать многопоточные приложения в соответствии с политикой распределения памяти, привязывая приложение к определенным ядрам или NUMA-узлам. Приложения, использующие POSIX-потоки, могут быть запущены в определенных сегментах памяти. Топология системы доступна утилите `numactl` из `/sys`.

Условия применения

Рекомендуется использовать утилиту `numactl` для эффективного использования FPU в системе. Например, при запуске на мастер-узле[5] необходимо сначала получить информацию об используемом оборудовании в разрезе использования NUMA-узлов:

Показать доступные узлы

```
$ numactl --hardware
```

вывод

```
available: 6 nodes (0-5)
node 0 cpus: 0 1 2 3 4 5 6 7
node 0 size: 32169 MB
node 0 free: 31624 MB
node 1 cpus: 8 9 10 11 12 13 14 15
node 1 size: 32254 MB
node 1 free: 32188 MB
node 2 cpus: 16 17 18 19 20 21 22 23
node 2 size: 32254 MB
node 2 free: 31950 MB
node 3 cpus: 24 25 26 27 28 29 30 31
node 3 size: 32254 MB
node 3 free: 32143 MB
node 4 cpus: 32 33 34 35 36 37 38 39
node 4 size: 32254 MB
node 4 free: 32202 MB
```

[5] минимальная конфигурация используется в качестве примера для снижения объемов вывода утилиты

```
node 5 cpus: 40 41 42 43 44 45 46 47
node 5 size: 32254 MB
node 5 free: 32200 MB
node distances:
node   0   1   2   3   4   5
0:  10  16  16  22  16  22
1:  16  10  22  16  22  16
2:  16  22  10  16  16  22
3:  22  16  16  10  22  16
4:  16  22  16  22  10  16
5:  22  16  22  16  16  10
```

Дополнительная информация о политике использования NUMA-узлов:

Показать политики numactl

```
$ numactl --show
```

вывод

```
policy: default
preferred node: current
physcpubind: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
cpubind: 0 1 2 3 4 5
nodebind: 0 1 2 3 4 5
membind: 0 1 2 3 4 5
```

В совокупности с выводом команды `lscpu` параметры `--hardware` и `--info` дают исчерпывающую информацию об используемом оборудовании. Данный мастер-узел содержит 3 процессора AMD Opteron 6380, всего 6 NUMA-узлов. Каждый NUMA-узел имеет 8 ядер, но 4 FPU. Таким образом, для использования всех 24 FPU с помощью `numactl` необходимо использовать следующую команду:

```
$ numactl \
-C 0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46 \
-m 0,1,2,3,4,5,6 \
./app
```

где параметр `-C` (синоним `--physcpubind=cpus`) принимает значения ядер CPU как показывает вывод команды `cat /proc/cpuinfo | grep processor`; параметр `-m`

(синоним `--membind=nodes`) устанавливает NUMA-узлы, память которых будет выделена.

3.2.3. numastat

Функциональное назначение

Утилита `numastat` показывает статистику использования памяти по каждому NUMA-узлу для конкретных процессов и операционной системы.

Общая статистика для мастер-узла

```
1
2  Per-node numastat info (in MBs):
3                            Node 0          Node 1          Node 2          Node 3
4                   --------------- --------------- --------------- ---------------
5  Numa_Hit               1135.35         1248.89          355.38         2236.85
6  Numa_Miss                 0.00            0.00            0.00            0.00
7  Numa_Foreign              0.00            0.00            0.00            0.00
8  Interleave_Hit            8.78            9.16            9.07            9.07
9  Local_Node             1125.54         1222.70          329.04         2210.60
10 Other_Node                9.81           26.20           26.34           26.25
11
12                           Node 4          Node 5           Total
13                  --------------- --------------- ---------------
14 Numa_Hit                125.60         1470.32         6572.40
15 Numa_Miss                 0.00            0.00            0.00
16 Numa_Foreign              0.00            0.00            0.00
17 Interleave_Hit            9.08            9.16           54.33
18 Local_Node               99.90         1453.87         6441.64
19 Other_Node               25.70           16.45          130.76
```

Условия применения

Рекомендуется использовать утилиту `numastat` для анализа параметров доступа к памяти для определенного запущенного процесса по NUMA-узлам.

```
1 # export OMP_NUM_THREADS=6
2 # ./xhpcg
```

```
# ps -elf | grep xhpcg
0 R user51 21802 19873 99  80   0 - 257838 -  15:37 pts/0  00:04:59 ./xhpcg

# numastat -p 21802
Per-node process memory usage (in MBs) for PID 19945 (xhpcg)
                           Node 0          Node 1          Node 2
                  --------------- --------------- ---------------
Huge                         0.00            0.00            0.00
Heap                        12.82           13.49          635.56
Stack                        0.00            0.00            0.05
Private                     24.23            6.43           30.48
----------------  --------------- --------------- ---------------
Total                       37.06           19.92          666.09

                           Node 3          Node 4          Node 5
                  --------------- --------------- ---------------
Huge                         0.00            0.00            0.00
Heap                        13.50          108.02            1.24
Stack                        0.01            0.00            0.00
Private                     32.57           46.32            4.54
----------------  --------------- --------------- ---------------
Total                       46.08          154.34            5.79

                           Total
                  ---------------
Huge                         0.00
Heap                       784.63
Stack                        0.06
Private                    144.59
----------------  ---------------
Total                      929.28
```

3.2.4. `top` и `vmstat`

Функциональное назначение

Приложение `top` используется для мониторинга процессов операционной системы в реальном режиме времени. Для просмотра потоков OpenMP рекомендуется запуск в режиме потоков:

```
$ top H
```

Одновременно с `top` может быть запущена команда `vmstat`, которая показывает отчет о статистике виртуальной памяти:

```
$ vmstat -S m -w 5
```

Ниже представлен вывод данной команды.

```
procs -----------------------memory---------------------- ---swap-- -----io---- -system-- --------cpu--------
### КОММЕНТАРИЙ: ВЫВОД НЕПОСРЕДСТВЕННО ПРИ ЗАПУСКЕ ТЕСТА ПРОИЗВОДИТЕЛЬНОСТИ С 24 ПРОЦЕССАМИ
 r  b         swpd         free         buff        cache   si   so    bi    bo   in    cs us sy id wa st
24  0            0      3217161           38         1052    0    0     0     0    0     0  0  0 100  0  0
24  0            0      3217161           38         1052    0    0     0  5728 8702  8069  7  0  93  0  0
24  0            0      3217161           38         1052    0    0     0     8 6747  7598  7  0  93  0  0
25  0            0      3217161           38         1052    0    0     0     2 6630  7328  7  0  93  0  0
178 0            0      3217160           38         1051    0    0     0     2 6874  8113  7  0  93  0  0

### КОММЕНТАРИЙ: ЗАВЕРШЕНИЕ РАБОТЫ ТОГО ЖЕ ПРИЛОЖЕНИЯ
24  0            0      3217002           39         1054    0    0     0     2 7458  7962  8  0  92  0  0
28  0            0      3217002           39         1055    0    0     0     2 7143  7886  7  0  92  0  0
25  0            0      3217001           39         1055    0    0     0  1135 7331  7622  8  0  92  0  0
 0  0            0      3240537           39         1009    0    0     0  3135 8345  7476  3  0  97  0  0
 0  0            0      3240537           39         1009    0    0     0    13 1068  1661  0  0 100  0  0
 0  0            0      3240540           39         1009    0    0     0     6 1076  1744  0  0 100  0  0
```

Условия применения

Вывод команды `vmstat` содержит информацию о следующих параметрах:

- `r` — среднее число запущенных процессов;
- `b` — среднее число процессов, ожидающих I/O (должно быть низким, идеально равным 0);
- `in` — прерывания в секунду (при высоких значениях менее эффективно);
- `cs` — переключения контекста в секунду (возникают при выполнении системного вызова или блокировки);
- `us` — процент времени выполнения кода приложения (является индикатором потребления вычислительных ресурсов приложением, должно быть близко к проценту выделенных ядер);
- `sy` — процент времени выполнения кода ядра (ключевой показатель чрезмерного использования системных вызовов и переключений контекста; нормальное значение — менее %3);
- `wa` — процент времени ожидания I/O диска или сети.

Глава 4. Особенности запуска параллельных приложений

4.1. Запуск приложений, использующих OpenMP

4.1.1. Общие рекомендации

Аспекты запуска OpenMP-приложений сводятся к двум рекомендациям:

1. Использовать новейшую версию оптимизированного компилятора с модифицированной библиотекой `libgomp` (см. раздел 3.1.1).
2. Использовать соответствующие переменные окружения в зависимости от того, как приложение использует потоки.

4.1.2. Эффективное использование особенностей архитектуры процессора

Количество потоков и используемых ядер задается посредством следующих переменных окружения:

```
$ OMP_NUM_THREADS=48 GOMP_CPU_AFFINITY="0-47" ./app
```

В случае, если приложение использует 48 потоков, но в системе доступно 192 ядра, то эффективнее максимально распределять потоки. В данном примере для этого рекомендуется использовать каждое четвертое ядро:

```
$ OMP_NUM_THREADS=48 GOMP_CPU_AFFINITY="0-47:4" ./app
```

Время выполнения задачи может быть снижено таким образом в 2-4 раза. Это объясняется особенностью архитектуры процессора AMD Opteron 6380. Данный процессор

включает 2 блока, каждый из которых имеет 8 ядер, но только 4 FPU. Каждый блок соединен с соседним посредством интерфейса HyperTransport. Поэтому для увеличения производительности OpenMP-приложения рекомендуется балансировать нагрузку, распределяя потоки по всем доступным ядрам, используя больше FPU и при одновременно снижая нагрузку на HyperTransport Interface и кэш L3.

Динамическое регулирование числа потоков:

```
$ OMP_DYNAMIC=true ./app
```

Описание других переменных окружения доступно в документации [2].

4.2. Запуск приложений, использующих MPI

4.2.1. Использование оптимизированной версии OpenMPI

Для сборки и запуска MPI-приложений рекомендуется использовать оптимизированную версию OpenMPI, которая имеет BTL, максимально использующий NumaConnect.

Вывести описание модуля

```
$ module whatis numa/ompi/1.10_gcc-5.3.0:
```

Вывести описание модуля

```
Sets the environment for using OpenMPI build against GCC 5.3.0 with NumaConnect BTL
```

Пример запуска с использованием настроек NumaConnect:

```
$ mpiexec -np 48 \
--mca btl self,nc \
--mca btl_nc_shared_queues 1 \
--mca btl_nc_async_send 1 \
./app
```

Данный модуль выставит необходимые переменные `LD_LIBRARY_PATH` и `PATH`.

4.2.2. Использование tmpfs

Для снижения издержек чтения/записи временных файлов OpenMPI используется **tmpfs**. Хранение в **tmpfs** задается посредством переменной окружения OMPI_PREFIX_ENV:

```
$ OMPI_PREFIX_ENV=/dev/shm mpiexec -np 2 ./app
```

4.2.3. Использование rank-файлов

Данная рекомендация касается использования максимального числа FPU, как было предложено ранее в разделе 4.1 — «Запуск приложений, использующих OpenMP». Например, используется каждое 4-ое ядро в системе с 192 ядрами:

```
mpiexec -np 48 --rk rankfile_192_4 ./app
```

где **rankfile_192_4** содержит:

```
1  rank 0=node1 slot=0:0
2  rank 1=node1 slot=0:4
3  rank 2=node1 slot=0:8
4  rank 3=node1 slot=0:12
5  rank 4=node1 slot=1:0
6  ...
7  rank 43=node1 slot=10:12
8  rank 44=node1 slot=11:0
9  rank 45=node1 slot=11:4
10 rank 46=node1 slot=11:8
11 rank 47=node1 slot=11:12
```

Листинг 4.1 — Фрагмент **rank**-файла для использования каждого 4-го ядра

4.2.4. Дополнительные опции для запуска MPI-приложений

Для анализа аллокации процессов на нескольких узлах рекомендуется использование следующих параметров **mpiexec**:

- опции отладки, дающие представление о распределении процессов:
  - **--verbose**

- `--display-map`
- `--display-devel-map`
- `--display-allocation`
- `--debug-daemons`

- `--map-by core` — для лучшего распределения процессов в случае использования максимального числа ядер.

Описание других переменных окружения доступно в документации [1].

4.3. Запуск гибридных приложений (MPI + OpenMP)

4.3.1. Пример запуска

Пример сценария запуска параллельного приложения, использующего MPI и OpenMP:

```
1  #!/bin/sh
2  export OMPI_PREFIX_ENV=/dev/shm
3  export OMP_NUM_THREADS=4
4  mpiexec \
5    -np 48 \
6    -x OMP_NUM_THREADS=4 \
7    --host node1,node2 \
8    --map-by core \
9    --mca btl self,nc \
10   --mca btl_nc_shared_queues 1 \
11   --mca btl_nc_async_send 1 \
12   ./xhpcg
```

Листинг 4.2 — Запуск гибридного приложения

В листинге 4.2 для использования определенного количества потоков на разных узлах используется опция `-x OMP_NUM_THREADS=4`, которая помечена как *deprecated*, но работает в актуальной версии OpenMPI (1.10). В документации используемых версий OpenMPI предлагается использовать вместо опции `-x OMP_NUM_THREADS=4` новый вариант: `--mca mca_base_env_list "OMP_NUM_THREADS=4"`. Без указания данной опции на

втором и последующих узлах будет использовано то число потоков, которое определит для себя само приложение. Переменные окружения OpenMP не будут заданы на этих узлах.

4.3.2. Пример rank-файла

Осуществляется запуск 24-х MPI-процессов с 8 потоками на макроузле. Макроузел имеет 192 ядра. Рекомендуется следующая структура rank-файла:

```
1 rank 0=node1 slot=0:0,0:2,0:4,0:6,0:8,0:10,0:12,0:14
2 rank 1=node1 slot=1:0,1:2,1:4,1:6,1:8,1:10,1:12,1:14
3 rank 2=node1 slot=2:0,2:2,2:4,2:6,2:8,2:10,2:12,2:14
4 rank 3=node1 slot=3:0,3:2,3:4,3:6,3:8,3:10,3:12,3:14
5 ...
6 rank 23=node1 slot=23:0,23:2,23:4,23:6,23:8,23:10,23:12,23:14
```

Листинг 4.3 — Фрагмент rank-файла для запуска 24 процессов с 8 потоками

Пример запуска:

```
1 export OMP_NUM_THREADS=8 \
2 mpiexec -np 24 \
3         --mca btl self,nc \
4         --mca btl_nc_shared_queues 1 \
5         --mca btl_nc_async_send 1 \
6         ./hybrid_app
```

Глава 5. Подготовка к работе

Требования, предъявляемые к квалификации пользователя, а так же описание общего принципа использования системы и порядок получения реквизитов для удалённого доступа представлены в разделах 3.1–3.3 документации «Программное обеспечение «РСК БАЗИС» — руководство пользователя».

Ссылочная документация

[1] Project The Open MPI. Open MPI Documentation // Current release series. — 2016. — URL: `https://www.open-mpi.org/doc/` (дата обращения: 28 ноября 2016 г.).

[2] The OpenMP API specification for parallel programming // OpenMP Specifications. — 2016. — URL: `http://openmp.org/wp/openmp-specifications/` (дата обращения: 28 ноября 2016 г.).

Список иллюстраций

Список таблиц

Листинги

Предметный указатель

Лист регистрации изменений бумажной версии

№	Номера листов (страниц)				Всего страниц	№ разрешающего документа	Подпись	Дата	Примечание
	Измененных	Замененных	Новых	Аннулированных					