Высокопроизводительный вычислительный центр НИЯУ МИФИ

Руководство пользователя

Версия 1.4.7

Москва, 2023

Все замечания и предложения по этой документации с благодарностью принимаются по адресу hpc@lists.ut.mephi.ru.

Создано при помощи XeLaTeX и BibTeX.

Последнее изменение: 28 декабря 2023 г.

Содержание

1. Общие положения

Высокопроизводительный вычислительный центр НИЯУ МИФИ (далее — HPC-Центр) предназначен для выполнения ресурсоёмких и/или распределённых вычислений сотрудниками и учащимися Университета при выполнении научных, исследовательских и образовательных задач, в частности, для обучения использованию современных HPC-технологий.

В рамках имеющихся аппаратных ресурсов пользователям предоставляется возможность выполнять задачи с использованием следующих технологий на основе ОС Linux:

- SLURM (менеджер ресурсов [1]);
- MPI (реализация OpenMPI [2]);
- GlusterFS (сетевая файловая система [3]).

2. Как стать пользователем

Перед тем как подавать заявку на использование ресурсов Центра, пожалуйста, ознакомьтесь с правилами использования. Фактом подачи заявки Вы подтверждаете своё согласие с данными правилами и обязательность их исполнения Вами.

2.1. Правила использования

1. Ресурсы HPC-Центра предназначены для поддержки фундаментальных и прикладных научных исследований,исследовательских и образовательных задач, требующих привлечения HPC.
2. Пользователями могут быть: сотрудники, учащиеся НИЯУ МИФИ и организаций, имеющих с НИЯУ МИФИ совместные научные проекты.
3. Пользователям категорически запрещается передавать свою учётную запись, пароль к ней или секретный ssh-ключ иным лицам.
4. Пользователь обязуется не использовать Центр для задач не указанных в п.1, в т.ч. для какой-либо деятельности, противоречащей законодательству РФ.
5. Установка и использование нелегального программного обеспечения категорически запрещена, как и любое нарушение лицензий

на используемое ПО (например, попытка использования в HPC-Центре бесплатного только для частного использования ПО). При использовании лицензионного ПО пользователь должен передать копию лицензии администраторам.

6. Пользователям запрещается пытаться обойти систему защиты, квот или административных ограничений HPC-Центра, в частности, эксплуатировать уязвимости.

7. В случае обнаружения уязвимости системы, пользователь обязан незамедлительно сообщить об этом администраторам лично по e-mail hpc-private@ut.mephi.ru.

8. Административные объявления, а также проблемы и предложения пользователей обсуждаются через список рассылки hpc@lists.ut.mephi.ru. Почта администраторов (hpc-private@ut.mephi.ru) предназначена только для решения проблем безопасности.

9. Администрация HPC-Центра выполняет тщательный аудит действий пользователей.

10. В случае обнаружения нелегитимной активности или нарушения данных правил со стороны пользователя, администрация оставляет за собой право блокирования соответствующей учётной записи с возможным удалением нелегальных данных.

11. Пользователи в публикациях работ, выполненных при помощи HPC-Центра, обязуются ссылаться на использование его ресурсов. Для русскоязычных работ следует использовать формулировки вида «при проведении работ были использованы ресурсы высокопроизводительного вычислительного центра НИЯУ МИФИ» для англоязычных — “our work was performed using resources of NRNU MEPhI high-performance computing center”.

12. Администрация прилагает все возможные усилия для безотказной работы HPC-Центра и сохранности пользовательских данных. Однако, в силу объективных обстоятельств, невозможно гарантировать абсолютную стабильность и сохранность информации, поэтому пользователи должны регулярно сохранять полученные результаты и хранить копии особо важных данных вне ресурсов HPC-Центра.

2.2. Получение учётной записи

Для получения учётной записи необходимо, чтобы сотрудник МИФИ в должности доцента (начальника отдела) и выше заполнил на Вас заявку https://it.mephi.ru/service-notes/32. Затем, пользователю необходимо сгенерировать пару из публичного и приватного ключей SSH-2 RSA длиной не менее 16384 бит (см. п. 2.2.2) для удаленного доступа ssh или пару из публичного и приватного ключей OpenPGP [4] [5] [6] для передачи пароля по шифрованной почте.

Публичный ключ необходимо отправить на почтовый ящик hpc-private@ut.mephi.ruс корпоративной почты (@mephi.ru для сотрудников, @ut.mephi.ru для студентов) пользователя или автора заявки.

Затем, пользователю необходимо подписаться на список рассылки hpc@lists.ut.mephi.ru. Для этого необходимо зайти на сайт https://lists.mephi.ru/listinfo/hpc, в поле `Your email address:` указать Ваш почтовый адрес и нажать на кнопку `Subscribe`.

Сотрудник, заполнивший заявку, получит уведомление об изменении статуса заявки по корпоративной почте.

При желании Вы сможете изменить пароль с помощью команды `passwd`, но при этом он должен будет соответствовать строгим требованиям безопасности как по длине, так и по сложности. Система не позволит Вам установить слабый пароль.

2.2.1. Регистрация группы пользователей

При необходимости получить учётные записи для большой группы пользователей, можно подать заявку сразу на всю группу в виде служебной записки от руководителя группы или подразделения на имя начальника управления информатизации Романова Николая Николаевича. В служебной записке должны быть перечислены ФИО пользователей, их e-mail, цель и срок завершения работ (например, срок окончания обучения). Ответственность за достоверность предоставленных данных находится на руководителе, подавшем служебную записку.

2.2.2. Использование ssh-ключей

При желании, пользователь после может использовать ssh-ключи для доступа к ресурсам НРС-Центра. Для этого необходимо *на клиентской машине* создать пару из публичного `id_rsa.pub` и приватного `id_rsa` ключей с помощью команды:

```
ssh-keygen -t rsa -b 16384
```

для клиента OpenSSH [7] на ОС Linux, FreeBSD, MacOS и окружении Cygwin [8] для ОС Windows, или команды:

```
puttygen.exe -t rsa -b 16384
```

для клиента PuTTY [9] [10] на ОС Windows.

Не забудьте указать сложный пароль для защиты ключа! Затем поместите публичный ключ в файл `~/.ssh/authorized_keys` на сервере аутентификации. Обратите внимание, что передача приватного ключа иным лицам, в том числе администраторам HPC-центра, категорически запрещена.

Для использование ssh-ключей между фермами центра нужно создать пару из публичного и приватного ключа на сервере аутентификации и поместить публичный ключ в файл `~/.ssh/authorized_keys` там же. Раз в час файлы `~/.ssh/authorized_keys` синхронизируются на фермы.

3. Ресурсы HPC-Центра

3.1. Ферма «Университетский кластер»

Вычислительные ресурсы Университетского кластера составляют:

- 216 ядер;
- 864 GB RAM;
- сеть 1 Gbit/s;
- подключение к хранилищам данных 10 Gbit/s;
- пиковая производительность ~ 3.0 TFlops64.

Ферма состоит из 1 управляющего и 27 вычислительных узлов. Каждый узел состоит из:

- 2 x E5450 Intel Xeon CPU;
- 4 физических ядра на CPU;
- 32 GB RAM (DDR2 667 MHz).

Обеспечивает 4-х кратное ускорение векторных SIMD вычислений. Для задач, не использующих методы векторного программирования, эта ферма должна использоваться в первую очередь.

На этой ферме не рекомендуется использовать более одного узла в одной задаче.

На ферме имеется резерв ресурсов для срочных задач продолжительностью менее часа (`sbatch -p fast` см. п. 4.5.1).

3.2. Ферма «Basov»

Основные параметры фермы:

- 304 физических ядра;
- 2.4 TiB RAM;
- сеть 10 Gbit/s;
- пиковая производительность ~ 6.6 TFlops64.

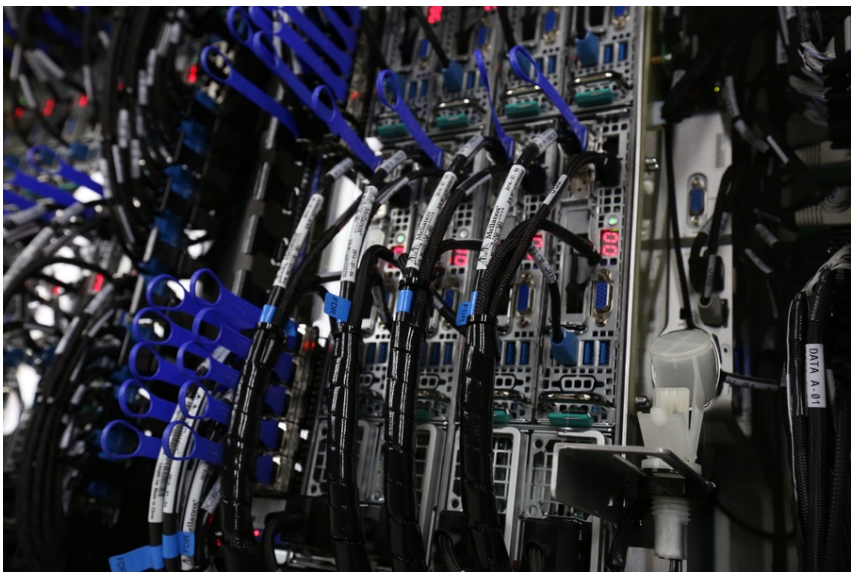
Ферма включает в себя 1 управляющий и 19 вычислительных узлов. Каждый узел состоит из:

- 2 x E5-2680 Intel Xeon CPU;
- 8 физических ядер на CPU;
- 128 GiB RAM (DDR3 1600 MHz).

Обеспечивает 8-ми кратное ускорение векторных SIMD вычислений.

На ферме имеется резерв ресурсов для срочных задач продолжительностью менее часа (`sbatch -p fast` см. п. 4.5.1).

3.3. Ферма «Cherenkov»



Основные параметры фермы:

- 304 физических ядра;
- 2.4 TiB RAM;
- 44 TiB полезного дискового пространства;
- сеть 56 Gbit/s;
- пиковая производительность $\sim$ 12 TFlops64.

Ферма включает в себя 1 управляющий и 19 вычислительных узлов. Каждый узел состоит из:

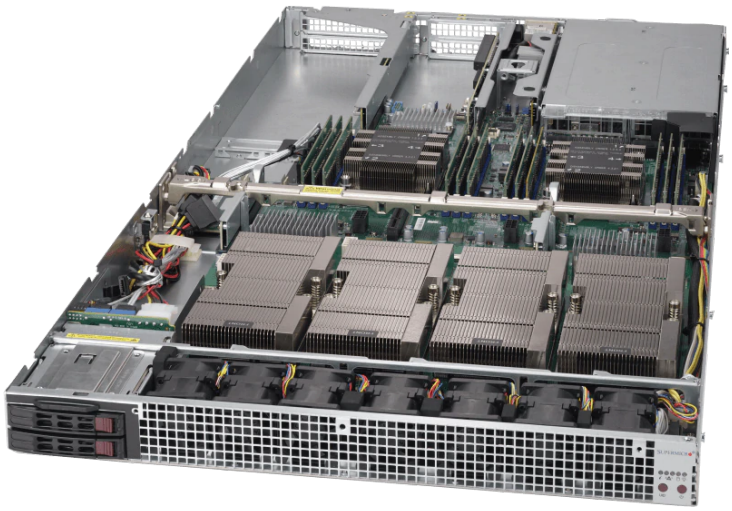
- 2 x E5-2630v3 Intel Xeon CPU;

- 8 физических ядер на CPU;
- 128 GiB RAM (DDR4 2133 MHz).

Особенностью фермы Cherenkov является использование высокопроизводительной неблокирующей сети Infiniband FDR. Эта ферма хорошо подходит для задач с интенсивным обменом данными MPI.

Обеспечивает 16-ти кратное ускорение векторных SIMD вычислений. Если Ваше приложение поддерживает SIMD библиотеки, используйте на ферме Cherenkov очередь (раздел, partition) задач avx2 (`sbatch -p avx2` см. п. 4.5.1). Не используйте очередь avx2 без векторных SIMD-библиотек!

3.4. Ферма «Tensor»



Основные параметры фермы:

- 612 физических ядер CPU;
- 1280 тензорных ядер GPU;
- 10240 ядер CUDA;
- 6.5 TiB RAM;
- 64 GiB GPU RAM;
- 131 TiB полезного дискового пространства;
- сеть 2x100 Gbit/s;

- пиковая производительность CPU ~ 47.0 TFlops64;
- пиковая производительность GPU ~ 15.6 TFlops64.

Ферма включает в себя 1 управляющий, 1 файловый, 1 GPU и 17 CPU узлов. GPU-узел состоит из:

- 2 x Intel Xeon Gold 6252 CPU;
- 12 физических ядер на SNC, по 2 SNC на CPU;
- 2 x NVIDIA Tesla V100 SXM2 32GB GPU;
- 192 GiB RAM (DDR4 2933 MHz).

Каждый CPU-узел состоит из:

- 2 x Intel Xeon Gold 6240 CPU;
- 9 физических ядер на SNC, по 2 SNC на CPU;
- 384 GiB RAM (DDR4 2933 MHz).

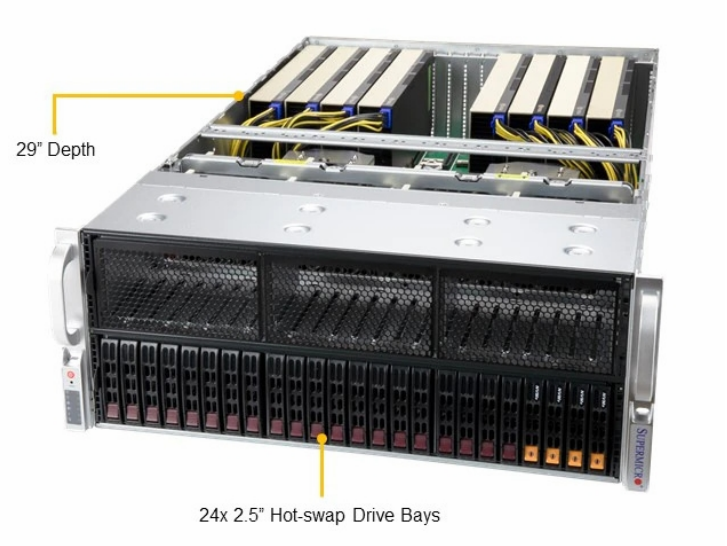
Особенностью фермы Tensor является использование графических ускорителей GPGPU с поддержкой языка программирования CUDA. Этот модуль подходит для тензорных алгоритмов наподобие нейронных сетей.

Внимание! GPU-узел фермы Tensor запрещается использовать для задач, не использующих графические ускорители GPGPU! Центральные процессоры GPU-узла предназначены для обеспечения работы GPGPU.

Другой особенностью фермы Tensor является использование высокопроизводительной неблокирующей сети Infiniband EDR dual-rail. Эта ферма хорошо подходит для задач с интенсивным обменом данными MPI.

Обеспечивает 32-х кратное ускорение векторных SIMD вычислений. Если Ваше приложение поддерживает SIMD библиотеки, используйте на ферме Cherenkov очередь (раздел, partition) задач avx512 (`sbatch -p avx512` см. п. 4.5.1). Не используйте очередь avx512 без векторных SIMD библиотек!

Для задач, использующих методы векторного программирования, эта ферма должна использоваться в первую очередь.



3.5. Модуль «Basis»

Основные параметры модуля:

- 64 физических ядра CPU;
- 3456 тензорных ядер GPU;
- 55296 ядер CUDA;
- 512 GiB RAM;
- 320 GiB GPU RAM;
- сеть 100 Gbit/s;
- пиковая производительность GPU ~ 78 TFlops64.

Модуль включает в себя 1 управляющий и 1 GPU узел. GPU-узел состоит из:

- 2 x AMD EPYC 7452 CPU;
- 4 физических ядра на SNC, по 8 SNC на CPU;
- 8 x NVIDIA Tesla A100 PCIe4 40G GPU;
- 512 GiB RAM (DDR4 3200 MHz);
- 2 x NVMe SSD накопителя 6.4 TB 6.2/4.0 GB/s 1000/250 kIOPS.

Особенностью модуля Basis является использование графических ускорителей GPGPU с поддержкой языка программирования CUDA. Этот модуль подходит для тензорных алгоритмов наподобие нейронных сетей.

Внимание! GPU-узел модуля Basis запрещается использовать для задач, не использующих графические ускорители GPGPU! Центральные процессоры GPU-узла предназначены для обеспечения работы GPGPU.

3.6. Хранилище pool/6

Хранилище `pool/6` представляет собой GlusterFS Distributed Volume [3] кластер с использованием локальных файловых систем XFS и программных RAID-10 массивов на узлах.

Полезная емкость хранилища составляет 524 TiB.

Хранилище состоит из 4-х узлов. Каждый узел состоит из:

- 1 x Intel Xeon 3204 CPU;
- 96 GiB RAM (DDR4 2133 MHz);
- LSI 3008 SAS3 JBOD контроллер;
- 36 x SAS3 8TB HDD 7200 rpm 256 MB cache;
- сетевой интерфейс IB FDR 56 Gbit/s.

3.7. Какие фермы следует использовать

Если Ваша задача поддерживает использование GPU (видеокарт, графических ускорителей), используйте в порядке приоритета:

- очередь gpu на ферме Basis;

- очередь gpu на ферме Tensor.

Если Ваша задача поддерживает векторные SIMD библиотеки, используйте в порядке приоритета:

- очередь avx512 на ферме Tensor;
- очередь avx2 на ферме Cherenkov.

Если Ваша задача не поддерживает векторные SIMD библиотеки, используйте в порядке приоритета:

- очередь cpu на ферме Unicluster;
- очередь cpu на ферме Basov;
- очередь cpu на ферме Cherenkov;
- очередь cpu на ферме Tensor.

Если Вы выбрали ферму Unicluster, старайтесь не использовать более одного узла в одной задаче.

3.8. Программная инфраструктура

Фермы работают на базе операционной системы GNU/Linux Debian [11].

Пользователи непосредственно работают по ssh только с управляющими узлами ферм, они же предназначены для компиляции приложений. Работа с вычислительными узлами осуществляется посредством инструментов SLURM без прямого доступа пользователя.

На фермах предоставляются следующие инструменты:

SLURM На фермах используется система управления распределёнными вычислениями SLURM [1]. SLURM имеет обратную совместимость со стартовыми скриптами стандарта OpenPBS [12].В рамках SLURM реализована поддержка MPI задач.

MPI Поддержка MPI (Message Passing Interface, инструмент для обмена данными между параллельно работающими задачами на разных узлах) реализована с помощью пакетов OpenMPI [2] и MPICH2 [13]. Текущие версии пакетов можно узнать при помощи утилиты apt-cache (подробнее в 4.4).

С точки зрения пользователя, MPI-приложение работает внутри PBS-задачи. Поддерживается ROMIO [14] I/O API.

Работа MPI-приложений ускорена с использованием технологии KNEM [15], особо эффективной для передачи больших объёмов данных, асинхронного и векторного обмена данными.

GCC Для компиляции пользовательских приложений предоставляется стандартная для Linux коллекция компиляторов GCC [16], поддерживающая следующие языки: C, C++, Assembler, Fortran, Objective C, Objective C++. Поддерживается технология OpenMP [17]. Для компиляции MPI приложений следует использовать соответствующие команды с префиксом «mpi»: `mpicc`, `mpic++`, `mpif77`, `mpi90`. Реализацию MPI можно указать в виде постфикса, например: `mpicc.openmpi` или `mpicc.mpich`.

Текстовые редакторы Представлены текстовые редакторы Vim, Emacs и, для новичков, mcedit и nano.

Обратите внимание, что X-сервер на наших фермах не поддерживается. Аппаратные акселераторы визуализации физически отсутствуют. Фермы предназначен для вычислительных задач, а не для визуализации полученных результатов, которую необходимо выполнять на клиентских системах.

3.9. Предустановленное ПО

Системные администраторы обеспечивают поддержку ПО, имеющегося в официальных репозиториях установленных дистрибутивов. Поддержка остального ПО осуществляется в формате ответов на вопросы пользователей. Пользователям разрешается собирать ПО из исходных кодов и устанавливать бинарные приложения в персональных директориях при условии соблюдения лицензионных соглашений (подробнее в 4.4).Разрешается использование пользовательских окружений chroot (например, с помощью proot [18]).

3.10. Политика обновлений

С целью поддержания актуальности и безопасности установленного общесистемного ПО, проводятся регулярные обновления системы. Плановое обновление проходит между учебными семестрами. Администраторы оставляют за собой право проводить экстренное обновление отдельных компонент при выявлении серьёзных проблем безопасности.

3.11. Несвободное ПО

Intel oneAPI На фермах доступно несвободное программное обеспечение Intel oneAPI. В состав программного комплекса oneAPI входят компиляторы `icc` (C/C++), `ifort` (Fortran),средства отладки и профилирования, библиотеки MKL, Intel MPI и другие.

ПО oneAPI располагается в директориях `/opt/intel/oneapi/`. Ссылки на документацию доступны в файлах `/opt/intel/oneapi/readme...` Полные тексты лицензионных соглашений содержатся в директории `/opt/intel/oneapi/licensing/`.

Intel PS XE На фермах доступно несвободное лицензионное программное обеспечение Intel Parallel Studio XE 2020 Cluster Edition for Linux - Floating Academic 2 seats. В состав программного комплекса PS XE входят архивные версии компиляторов `icc` (C/C++), `ifort` (Fortran), средств отладки и профилирования, библиотек MKL, Intel MPI и других.

ПО PS XE располагается в директориях `/opt/intel/`. Руководство пользователя доступно в директории `/opt/intel/documentation_2020/`. Приобретённая лицензия накладывает ограничения:

- лицензия ограничена двумя пользователями, компилирующими одновременно;
- количество одновременных запусков скомпилированных приложений не ограничено;
- лицензия разрешает использование в научной и преподавательской деятельности, включая выполнение научных грантов;
- лицензия не разрешает использование в работах по хозяйственным договорам (с целью коммерциализации).

Полные тексты лицензионных соглашений содержатся в директориях продуктов, например `/opt/intel/.../licensing/EULA.txt`.

NVIDIA HPC SDK На фермах доступно несвободное программное обеспечение NVIDIA HPC SDK. В состав программного комплекса HPC SDK входят компиляторы `nvcc` и `nvfortran` (CUDA), средства отладки и профилирования, библиотеки libcublas, libcufftw, libcurand и другие.

ПО располагается в директории `/opt/nvidia/hpc_sdk/`.

Полные тексты лицензионных соглашений содержатся в директориях продуктов, например `/opt/nvidia/.../license/LICENSE.txt`.

NVIDIA CUDA Toolkit На фермах доступно несвободное программное обеспечение NVIDIA CUDA Toolkit из состава репозитория Debian non-free. В состав программного комплекса CUDA Toolkit входят компилятор `nvcc` (CUDA), средства отладки и профилирования, библиотеки libcublas, libcufftw, libcurand и другие.

Полный текст лицензионного соглашения доступен в файле `/usr/share/doc/nvidia-cuda-doc/copyright`.

4. Использование ферм

4.1. Предварительные требования

Для работы с фермами пользователю необходимо обладать базовыми навыками работы в ОС Linux; в частности, необходимо владеть `bash`, `ssh`, `gcc`, одним из установленных текстовых редакторов (`vim`, `emacs`, `nano`, `mcedit`), уметь устанавливать приложения и обладать навыками программирования, достаточными для компиляции пользовательских приложений.

Большинство этих аспектов выходит за рамки данного руководства, которое посвящено описанию предоставляемой рабочей среды (см. раздел 3.8) и особенностям, специфичным для ферм HPC-Центра. Для каждого приложения на фермах установлена документация, доступная посредством `man`, `info` и в `/usr/share/doc`. Однако, начинающим в Linux можно порекомендовать следующие материалы для ознакомления: Linux in a Nutshell [19], Основы операционной системы UNIX [20], Advanced Bash-Scripting Guide [21]/Искусство программирования на языке сценариев командной оболочки [22], руководства для отладчиков `gdb` [23], `valgrind` [24], `strace` [25], профилировщика `perf` [26].

В общественном доступе имеются лекции по основам параллельного программирования [27] [28] [29] и руководства для высокопроизводительных математических библиотек [30] [31] [32] [33]. Следует помнить, что любая оптимизация программного обеспечения должна начинаться и заканчиваться профилированием.

4.2. Работа с ssh

Доступ пользователей на фермы осуществляется по протоколу `ssh` через сервер аутентификации hpc.mephi.ru. Для подключения необходимо выполнить команду:

```
$ ssh username@hpc.mephi.ru
```

для клиента OpenSSH, или команду:

```
$ putty.exe -i C:\path\to\private\key.ppk username@hpc.mephi.ru
```

для клиента PuTTY.

Резервный сервер аутентификации доступен по адресу hpc2.mephi.ru. Управляющие узлы фермы доступны с сервера аутентификации по адресам:

```
Available HPC clusters:
    basis
    basov
    cherenkov
    tensor
    unicluster

To log on cluster, type:
    ssh cluster_name

For example:
    ssh cherenkov

$ ssh cherenkov
```

При работе с `ssh` следует учитывать ограничения на интенсивность соединений, вызванные соображениями политики безопасности. Необходимо избегать повторных подключений, удерживая уже установленные соединения. При необходимости реализации нескольких соединений можно использовать механизм сокетов `ssh`. Для использования этого механизма следует включить его на стороне пользователя. Например, для пакета OpenSSH достаточно дописать в файл `/etc/ssh/ssh_config`:

```
ControlMaster auto
ControlPath ~/.ssh/socket-%r@%h:%p
```

При отсутствии доступа к фермам по протоколу `ssh` следует обращаться к системным администраторам. Обращаем Ваше внимание на то, что утилита `ping` не предназначена для проверки доступности `ssh` соединений.

4.3. Работа с файловой системой

Пользователям предоставляется три вида дисковых хранилищ:

- `/home`;
- `/mnt/pool`;

- `/tmp`.

Скопировать файл или директорию с локального компьютера на хранилище и обратно можно с помощью команд:

```
$ scp -r /path/to/local/file username@hpc.mephi.ru:/mnt/pool/6/username
```

```
$ scp -r username@hpc.mephi.ru:/mnt/pool/6/username/file /local/path
```

для клиента OpenSSH или команд:

```
$ pscp.exe -i C:\path\to\private\key.ppk -r C:\path\to\local\file ^
username@hpc.mephi.ru:/mnt/pool/6/username
```

```
$ pscp.exe -i C:\path\to\private\key.ppk -r ^
username@hpc.mephi.ru:/mnt/pool/6/username/file C:\local\path
```

для клиента PuTTY [9] [10].

4.3.1. /home

Домашние директории пользователей на разных фермах раздельные. Внутри каждой фермы домашняя директория доступна на всех вычислительных узлах за счет использования POSIX-совместимой сетевой файловой системы NFSv4.2 [34].

Данный раздел предназначен хранения конфигурационных файлов, ключей шифрования и прочих инфраструктурных данных небольшого объёма. Не используйте данный раздел для хранения приложений и вычислительных данных.

4.3.2. /mnt/pool

Основные хранилища данных ферм доступны в директории `/mnt/pool`. На актуальном хранилище `/mnt/pool/6` используется POSIX-совместимая кластераня файловая система GlusterFS [3].

Внимание! Использование утилиты `rsync` приводит к очень высокой нагрузке на файловую систему GlusterFS. Старайтесь использовать утилиту `cp`. При необходимости использования утилиты `rsync` применяйте опции `rsync --whole-file --inplace`.

На устаревших хранилищах `/mnt/pool/3` и `/mnt/pool/4` используется NFSv4.2.

Хранилище предназначено для исходных данных, результатов обработки и прочих пользовательских данных, а также для установки пользовательских приложений. Система квот не применяется, но пользователям не рекомендуется занимать более необходимого и более 1/3 от общего объёма хранилища.

Файловые системы, смонтированные в `/mnt/pool`, доступны со всех узлов всех кластеров.

Внимание! Отказоустойчивость уровня дисков обеспечивается реализацией RAID-10 из состава ядра Linux. Отказоустойчивость уровня сервера хранения данных не обеспечивается. Бэкапирование данных не производится. Хранилища вычислительного центра предназначены только для проведения текущих расчетов, использование их для целей архивирования не допускается. Не забывайте сохранять важные данные на собственные носители!

Внимание! Следует помнить о существенных различиях в аппаратном обеспечении ферм. Запускаемые приложения нужно компилировать для различных ферм отдельно.

Внимание! Различные экземпляры задач следует запускать в различных директориях. Запуск нескольких экземпляров однотипных задач в одной директории может привести к конфликту по именам временных файлов, повреждению данных, получению неверных результатов.

Пользователям следует иметь в виду, что производительность передачи данных по сети сильно зависит от размера блока данных. Если Ваше приложение будет производить работу с блоками данных размером порядка килобайт, Вы получите резкое падение производительности. Рекомендуется накапливать изменения данных до размеров порядка мегабайт, и только затем передавать их за один раз. Данное требование распространяется на любые сетевые протоколы, будь то работа с файлами на Gluster/NFS, или передача сообщений через MPI.

Если Вы ставите в очередь несколько задач одновременно, необходимо проследить, что с увеличением числа запущенных задач не возрастает время их выполнения. Такой негативный эффект может быть вызван ограничением производительности хранилища данных. В этом случае нужно запускать новые задачи только после завершения предыдущих используя инструмент `sbatch --dependency` (см. пример в разделе 4.5.4).

4.3.3. /tmp

На узле `t2n1` модуля Basis в директории `/tmp` доступна локальная высокопроизводительная файловая система для временных данных, работающая на основе RAID-0 массива NVMe SSD накопителей.

Данная файловая система предназначена исключительно для интенсивной обработки временных данных непосредственно в момент запуска задачи. Пример работы с директорией `/tmp` доступен в разделе 4.5.4.

Внимание! Файлы старше 2-х месяцев автоматически удаляются с данной файловой системы.

4.4. Подготовка пользовательских приложений

В общем случае, пользовательское приложение должно быть откомпилировано на управляющем узле соответствующей фермы. Для этого используется стандартный набор компиляторов `gcc`, описанный в разделе 3.8. Также доступны отладчики `gdb`, `valgrind`, `strace` и профилировщик `perf`.

На фермах предоставляется широкий набор системных и научных библиотек. Для того, чтоб узнать, установлена ли библиотека (или любой другой пакет) на ферме, нужно выполнить команду:

```
$ dpkg -l | grep имя_библиотеки
```

Чтобы узнать, есть ли библиотека в подключенных репозиториях необходимо использовать команду `apt-cache search`, например (для библиотеки быстрых Фурье-преобразований fftw):

```
$ apt-cache search fftw
...
libfftw3-3 - Library for computing Fast Fourier Transforms
...
```

Для того, чтобы узнать доступные версии необходимо использовать утилиту `apt-cache policy`, например:

```
$ apt-cache policy libfftw3-3
libfftw3-3:
  Installed: 3.3.8-2
  Candidate: 3.3.8-2
  Version table:
 *** 3.3.8-2 500
        500 http://mirror.mephi.ru/debian buster/main amd64 Packages
        100 /var/lib/dpkg/status
```

Подробно использование данной команды описано в `man apt`.

Полный путь до файлов установленной библиотеки можно узнать командой:

```
$ dpkg-query -L libfftw3-double3:amd64
...
/usr/lib/x86_64-linux-gnu/libfftw3.so.3.5.8
/usr/lib/x86_64-linux-gnu/libfftw3_omp.so.3.5.8
/usr/lib/x86_64-linux-gnu/libfftw3_threads.so.3.5.8
...
```

Обращаем внимание пользователей на наличие различных реализаций библиотек линейной алгебры, поддерживающих следующие интерфейсы программирования приложений (API): BLACS, BLAS, CBLAS, LAPACK, LAPACKE. Использование аппаратно оптимизированных высокопроизводительтных библиотек позволяет ускорить выполнение программы в десятки раз. Помимо версий из репозиториев дистрибутивов предоставляются новейшие сборки этих библиотек в директории `/opt` с готовыми скриптами настройки переменных окружения `customvars.sh`. Подключить их можно, например, так:

```
$ source /opt/fftw-3.3.9/double/customvars.sh
$ source /opt/openblas-0.3.13/customvars.sh
```

Обратите внимание, что эти команды надо выполнить как перед конфигурированием и компиляцией приложения, так и внутри стартового скрипта задачи (см. п. 4.5.1).

Проверить подключение внешних библиотек можно так:

```
$ ldd путь_до_приложения
...
        libopenblas.so.0 => /opt/openblas-0.3.13/lib/libopenblas.so.0
...
```

Для использования графических ускорителей доступны свободный интерфейс программирования API OpenCL и проприетарный язык программирования CUDA. OpenCL поддерживается компилятором gcc, CUDA поддерживается компилятором nvcc. Доступны готовые свободные библиотеки clBLAS, clFFT, и проприетарные CuBLAS, CuFFT, CuFFTW, CuRAND. Доступно много готовых высокоуровневых библиотек, поддерживающих работу с графическими ускорителями.

Если Вам необходима библиотека или приложение, имеющееся в официальных репозиториях, но не установленное на фермах, обратитесь к администраторам. Но обратите внимание, что X сервер и сопутствующие приложения не поддерживаются. Вычислительные задачи можно откомпилировать без поддержки X.

Не забывайте использовать флаги оптимизации компилятора, например:

```
$ gcc -march=native -mtune=native -Ofast example.c
```

Не забывайте использовать отладочные флаги компилятора при отладке и профилировании:

```
$ gcc -ggdb3 -O0 example.c
```

При использовании автоматизированных систем сборки флаги компилятора и линковщика можно указывать через переменные окружения CFLAGS, CXXFLAGS, FFLAGS, LDFLAGS, например:

```
$ CFLAGS=”-ggdb3 -O0” LDFLAGS=”-ggdb3” ./configure
```

, или

```
$ CFLAGS=”-ggdb3 -O0” LDFLAGS=”-ggdb3” cmake .
```

При использовании автоматизированных систем сборки не забывайте указывать префиксы директорий установки, например:

```
$ ./configure --prefix=/mnt/pool/6/username
```

, или

```
$ cmake -DCMAKE_INSTALL_PREFIX:PATH=/mnt/pool/6/username .
```

В противном случае у Вас возникнут ошибки отсутствия прав доступа при выполнении установки.

Для использования самостоятельно собранных библиотек и приложений по умолчанию необходимо настроить переменные окружения:

```
$ export PATH=/mnt/pool/6/username/bin:/$PATH
$ export LD_LIBRARY_PATH=/mnt/pool/6/username/lib:$LD_LIBRARY_PATH
$ export LIBRARY_PATH=/mnt/pool/6/username/lib:$LIBRARY_PATH
$ export CPATH=/mnt/pool/6/username/include:$CPATH
```

Не забывайте настраивать переменные окружения внутри стартового скрипта задачи (см. п. 4.5.1).

4.5. Запуск задач

Запуск и удаление задач выполняются с помощью инструментов менеджера ресурсов SLURM [1], исчерпывающе описанных в официальном руководстве пользователя SLURM и в соответствующих страницах man. На каждой ферме используется свой менеджер ресурсов.

4.5.1. Запрос ресурсов

В простейшем случае для подготовки задачи нужно составить скрипт исполнения задачи (например, `myjob.sh`) для любой доступной командной оболочки (например, `bash`), указав необходимые для работы ресурсы используя специально сформированный заголовок задачи, содержащий директивы `#SBATCH`:

```
#!/bin/bash
#
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=10:00:00

/mnt/pool/6/useraname/myprogramm my args
```

Данный скрипт предназначен для последовательной (не параллельной) программы и запрашивает 1 MPI поток на 1 вычислительном узле на 10 часов.

Менеджер ресурсов SLURM обратно совместим с синтаксисом заголовков стандарта OpenPBS [12]:

```
#!/bin/bash
#
#PBS -l nodes=1:ppn=1,walltime=10:00:00

/mnt/pool/6/username/myprogramm my args
```

Предпочтительным является собственный синтаксис SLURM.

Не следует запускать несколько задач в одной директории, поскольку разные экземпляры приложения могут создавать временные файлы с одинаковыми относительными путями, что приведет к конфликту записи нескольких задач в один файл.

Для постановки задачи в очередь на исполнение нужно выполнить команду:

```
$ sbatch myjob.sh
```

После завершения работы программы, stdin и stdout будут размещены в файле вида `${jobname}.out` в директории, в которой была выполнена команда `sbatch`.

Если Вы хотите использовать очередь (раздел, partition), отличную от умолчания, Вы можете использовать заголовок:

```
#SBATCH --partition=partition_name
```

или аргумент утилиты `sbath -p $partition_name` . Информацию об очередях можно узнать при помощи утилиты `sinfo`.

Полное описание доступных ресурсов можно найти в [1]. Пожалуйста, объективно оценивайте необходимые ресурсы и максимально точно их указывайте — это позволяет повысить эффективность работы диспетчера задач и уменьшить задержки в очередях.

Внимание! Задачи, превысившие отведённое им время walltime, уничтожаются.

4.5.2. Работа с параллельными приложениями

На фермах предусмотрена возможность работы с POSIX threads и OpenMP на общей памяти внутри одного вычислительного узла. Для работы с раздельной памятью на нескольких узлах поддерживаются различные реализации MPI: OpenMPI и MPICH2. SLURM использует следующие понятия для управления ресурсами:

- thread - вычислительный поток (нить), последовательная подпрограмма;
- task - вычислительный поток стандарта MPI, может содержать множество потоков OpenMP или POSIX threads;
- job - запуск программы, может содержать множество потоков MPI;
- cpu - логическое ядро процессора, обеспечивает работу одного thread без переключений контекста;
- core - физическое ядро процессора, может содержать несколько логических ядер;
- socket - сокет, процессор (SNC), может содержать несколько физических ядер;
- node - узел вычислительной сети, компьютер, сервер, может содержать несколько сокетов.

Информацию об аппаратной конфигурации вычислительного узла можно узнать при помощи команды:

```
$ scontrol show node $node_name
```

Выбор MPI окружения По умолчанию на фермах используется OpenMPI, для его применения никаких дополнительных действий выполнять не нужно.

Вы можете явно указать желаемую реализацию MPI при помощи постфиксов компилятора и утилиты запуска, например:

```
$ mpicc.openmpi example.c
$ mpirun.openmpi ./a.out
```

, или

```
$ mpicc.mpich example.c
$ mpirun.mpich ./a.out
```

Аналогично, постфиксы используются для разделения последовательных версий библиотек и библиотек, использующих различные реализации MPI, например:

```
$ apt-cache search hdf5
...
libhdf5-103 - Hierarchical Data Format 5 (HDF5)
libhdf5-dev - Hierarchical Data Format 5 (HDF5)
libhdf5-mpich-103 - Hierarchical Data Format 5 (HDF5)
libhdf5-mpich-dev - Hierarchical Data Format 5 (HDF5)
libhdf5-openmpi-103 - Hierarchical Data Format 5 (HDF5)
libhdf5-openmpi-dev - Hierarchical Data Format 5 (HDF5)
...
```

Внимание! Приложение, скомпилированное и слинкованное с одной реализацией библиотеки MPI не может быть запущено с использованием утилиты запуска другой реализации MPI.

Запуск MPI-приложений Для запуска MPI-приложений необходимо запросить нужное число tasks и nodes (см. раздел 4.5.2) и использовать `mpirun` внутри скрипта запуска задачи для выполнения нужного приложения. Указывать число потоков для `mpirun` не нужно: SLURM автоматически настроит mpirun через переменные окружения. В остальном работа с MPI-задачами не отличается от обычных задач.

Пример стартового скрипта задачи:

```
#!/bin/bash
#
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
```

```
#SBATCH --time=5:00:00

cd /mnt/pool/6/username/workdir
mpirun ./my_mpi_app
```

Запуск OpenMP-приложений Использование вложенного MPI+OpenMP (Hybrid-MPI) параллелизма позволяет экономить оперативную память параллельного приложения, но может приводить к потере производительности на современных серверах с многоуровневой иерархической структурой общей памяти. Оптимальное соотношение потоков OpenMP threads к потокам MPI tasks нужно определять экспериментально, однако, рекомендуется использовать не менее одного потока MPI task на socket. Пример использования:

```
#!/bin/bash
#
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=2
#SBATCH --ntasks-per-socket=1
#SBATCH --cpus-per-task=4
#SBATCH --time=5:00:00

cd /mnt/pool/6/username/workdir
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
srun --mpi pmix ./my_mpi_app
```

Как определить оптимальное число потоков? Для определения оптимального числа потоков необходимо выбрать ферму и выбрать хранилище данных в соответствии с рекомендациями (см. разделы 3.7, 4.3.2). Далее необходимо построить кривую масштабируемости. По оси абсцисс в логарифмических координатах откладываются запрошенные вычислительные потоки. По оси ординат откладываются затраченные ресурсы в виде произведения запрошенных потоков на затраченное время выполнения. Для подавляющего большинства практических алгоритмов с ростом числа потоков растет количество затраченных ресурсов, эффективность работы приложения падает. Это вызвано ростом потерь на синхронизацию, ввод-вывод и другие последовательные участки алгоритма. Ни в коем случае не следует использовать число потоков, большее, чем число потоков, при котором количество затраченных ресурсов увеличивается вдвое относительно последовательного однопоточного запуска. Если при указанном числе потоков время выполнения Вашей задачи все ещё не укладывается в ограничение для фермы, не следует увеличивать число потоков, следует разбивать задачу на части.

Ограничение на число потоков снизу может быть вызвано недостатком оперативной памяти на выбранном числе узлов. Обратите внимание на сочетание MPI и OpenMP. Объем запрошенной в задаче оперативной памяти на каждый вычислительный поток можно регулировать инструментом `#SBATCH --mem-per-cpu=`.

4.5.3. Работа с графическими ускорителями

Для запуска приложения с использованием графических ускорителей необходимо запросить нужное количество ускорителей через инструмент gres:

```
#!/bin/bash
#
#SBATCH --partition=gpu
#SBATCH --nodes=1
#SBATCH --gres=gpu:2
#SBATCH --gres-flags=enforce-binding
#SBATCH --ntasks=2
#SBATCH --time=1:00:00

cd /mnt/pool/6/username/workdir
mpirun ./my_gpu_app -some_my_gpu_args -my_gpu_count 2
```

Кроме того, для большинства приложений необходимо указать параметры запуска на графических ускорителях вручную. Эти параметры не стандартизованы. Описание параметров запуска для конкретного приложения необходимо искать в официальном руководстве пользователя для этого приложения.

Большинство приложений, поддерживающих графические ускорители, являются параллельными. Рекомендации по выбору числа потоков CPU на каждый GPU нужно искать в официальном руководстве пользователя для приложения. Широко распространены две модели использования GPU:

- 1 MPI task на каждый GPU (QUANTUMESPRESSO, LAMMPS+KOKKOS);
- максимальное возможное число MPI tasks на GPU (LAMMPS+GPU).

Аналогично, следует уточнить в руководстве пользователя приложения рекомендации по использованию OpenMP threads с GPU.

При использовании GPU может возникнуть проблема недостатка объема оперативной памяти ускорителя. Эта проблема может быть решена перебором количества MPI tasks и OpenMP threads на GPU и увеличением числа используемых GPU в задаче.

Для задач, использующих несколько GPU одновременно, настоятельно рекомендуется использовать библиотеку MPI их состава NVIDIA HPC SDK и указывать дополнительные параметры MCA:

```
mpirun --mca btl self,smcuda ./my_gpu_app -some_my_gpu_args
```

4.5.4. Использование директории /tmp

Для задач, требующих большую интенсивность работы с файлами, рекомендуется использовать высокопроизводительную временную директорию `/tmp` см .п. 4.3.3 и составной запуск задачи.

Такой подход позволяет не дожидаться освобождения основных вычислительных ресурсов (например, графических ускорителей) в момент копирования исходных данных с постоянного хранилища во временную директорию, сократить время расчета за счет высокой скорости работы временной директории и освободить основные ресурсы для следующей задачи в момент копирования результатов из временной директории в постоянную.

Пример скрипта препроцессинга `pre.sh`:

```
#!/bin/bash
#
#SBATCH --partition=gpu
#SBATCH --nodelist=t2n1
#SBATCH --ntasks=1
#SBATCH --time=1:00:00

mkdir /tmp/2022-08-24-100500
chmod 700 /tmp/2022-08-24-100500

cp -r /mnt/pool/6/username/100500/* /tmp/2022-08-24-100500
```

Пример скрипта запуска расчета `comp.sh`:

```
#!/bin/bash
#
#SBATCH --partition=gpu
#SBATCH --nodelist=t2n1
#SBATCH --gres=gpu:1
#SBATCH --gres-flags=enforce-binding
#SBATCH --ntasks=1
#SBATCH --time=1:00:00

cd /tmp/2022-08-24-100500
/mnt/pool/6/username/run/my_gpu_app
```

Пример скрипта постпроцессинга `post.sh`:

```
#!/bin/bash
#
#SBATCH --partition=gpu
#SBATCH --nodelist=t2n1
#SBATCH --ntasks=1
#SBATCH --time=1:00:00

mv /tmp/2022-08-24-100500 /mnt/pool/6/username/
```

Пример запуска составной задачи:

```
$ sbatch pre.sh
Submitted batch job 4376
$ sbatch --dependency=afterok:4376 calc.sh
Submitted batch job 4377
$ sbatch --dependency=afterany:4377 post.sh
Submitted batch job 4378
```

4.6. Мониторинг и управление задачами

Пользователь может просмотреть статус задач с помощью команды `squeue`. Для просмотре подробной иформации о задаче можно использовать команду:

```
$ scontrol show job $job_id
```

Статистику использования ресурсов в задаче можно посмотреть с помощью команды:

```
$ sstat -a -j $job_id
```

Для удаления задач следует использовать: `scancel $job_id`.

5. Контакты

Сообщить о проблеме можно используя форму https://tasks.mephi.ru/projects/hpc-support/issues/new.

Для связи с администраторами HPC-Центра следует воспользоваться e-mail hpc-private@ut.mephi.ru. *Пожалуйста*, пользуйтесь указанным коллективным адресом. Письма, отправленные на личные адреса администраторов могут быть обработаны с существенной задержкой.

Подать заявку на использование ресурсов НРС-Центра можно по адресу https://it.mephi.ru/service-notes/32, но перед этим Вы *должны* тщательно ознакомиться с разделом 2.

С опытными пользователями можно связаться через список рассылки e-mail hpc@lists.ut.mephi.ru. Перед отправкой сообщения необходимо подписаться на лист рассылки https://lists.mephi.ru/listinfo/hpc. Архив листа рассылки доступен по адресу https://lists.mephi.ru/pipermail/hpc/.

Письма, отправленные на hpc@lists.ut.mephi.ru, получают администраторы и все пользователи НРС-Центра. Письма, отправленные на hpc-private@ut.mephi.ru, получают только администраторы.

Список литературы

[1] Sched MD LLC. — Slurm Workload Manager, 2020. — URL: https://slurm.schedmd.com/. 3, 13, 22, 24

[2] The Open MPI Project. — Open MPI: A High Performance Message Passing Library, 2020. — URL: https://www.open-mpi.org/. 3, 13

[3] Gluster, 2022. — URL: https://www.gluster.org/. 3, 12, 18

[4] The GNU Privacy Handbook, 2021. — URL: https://www.gnupg.org/gph/en/manual/c14.html#AEN25. 5

[5] The Gpg4win Compendium, 2021. — URL: https://gpg4win.org/doc/en/gpg4win-compendium_12.html. 5

[6] Миллер В. В., Наги Д. А. — Создание и использование ключа OpenPGP с подключами, 2020. — URL: https://www.pgpru.com/biblioteka/rukovodstva/upravleniekljuchami/podkljuchiopenpgp. 5

[7] Аутентификация на SSH сервере с использованием ключей, 2021. — URL: https://www.opennet.ru/base/sec/ssh_pubkey_auth.txt.html. 6

[8] Cygwin, 2023. — URL: https://www.cygwin.com/. 6

[9] PuTTY, 2023. — URL: https://www.putty.org/. 6, 18

[10] How To Configure SSH Keys Authentication With PuTTY And Linux Server, 2021. — URL: https://putty.org.ru/articles/putty-ssh-key-auth.html. 6, 18

[11] The Debian Project. — Debian Linux, 2020. — URL: https://www.debian.org/index.ru.html. 13

[12] Adaptive Computing Inc. — TORQUE™ Resource Manager, 2020. — URL: https://adaptivecomputing.com/cherry-services/torque-resource-manager/. 13, 23

[13] MPICH: High-Performance Portable MPI, 2020. — URL: https://www.mpich.org. 13

[14] ROMIO: A High-Performance, Portable MPI-IO Implementation, 2020. — URL: https://www.mcs.anl.gov/projects/romio/. 14

[15] KNEM: High-Performance Intra-Node MPI Communication, 2022. — URL: https://knem.gitlabpages.inria.fr/. 14

[16] Free Software Foundation Inc. — GCC, the GNU Compiler Collection, 2020. — URL: https://gcc.gnu.org/. 14

[17] The OpenMP, 2020. — URL: https://www.openmp.org/. 14

[18] PRoot, 2020. — URL: https://proot-me.github.io/. 14

[19] Linux in a Nutshell / Ellen Siever, Stephen Figgins, Robert Love, Arnold Robbins. — 6th edition. — O'Reily Media, 2009. — P. 944. — Url for 3rd edition. Google Books : wXIvheS3r_gC. 16

[20] Кравчук В. — Основы операционной системы UNIX, 2020. — URL: https://www.opennet.ru/docs/RUS/unix_basic/index.html. 16

[21] Cooper Mendel. — Advanced Bash-Scripting Guide, 2020. — URL: https://tldp.org/LDP/abs/abs-guide.pdf. 16

[22] Cooper Mendel. — Искусство программирования на языке сценариев командной оболочки, 2020. — URL: https://www.opennet.ru/docs/RUS/bash_scripting_guide/. 16

[23] Schmidt Ryan Michael. — RMS's gdb Debugger Tutorial, 2020. — URL: http://www.unknownroad.com/rtfm/gdbtut/gdbtoc.html. 16

[24] Valgrind Development Team. — Valgrind, 2018. — URL: http://valgrind.org/. 16

[25] Strace. Linux syscall tracer, 2020. — URL: https://strace.io/. 16

[26] kernel.org. — Linux kernel, Perf, 2018. — URL: https://perf.wiki.kernel.org/index.php/Main_Page. 16

[27] Lawrence Livermore National Laboratory. — LLNL, Parallel Computing, 2022. — URL: https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial. 16

[28] Lawrence Livermore National Laboratory. — LLNL, MPI, 2022. — URL: https://hpc-tutorials.llnl.gov/mpi/. 16

[29] Lawrence Livermore National Laboratory. — LLNL, OpenMP, 2022. — URL: https://hpc-tutorials.llnl.gov/openmp/. 16

[30] GNU Foundation. — GSL: GNU Scientific Library, 2018. — URL: https://www.gnu.org/software/gsl/doc/html/index.html. 16

[31] Netlib. — BLAS: Basic Linear Algebra Subprograms, 2018. — URL: http://www.netlib.org/blas/. 16

[32] Netlib. — LAPACK: Linear Algebra Package, 2018. — URL: http://www.netlib.org/lapack/. 16

[33] fftw.org. — FFTW: Fast Fourier Transform in the West, 2018. — URL: http://www.fftw.org/. 16

[34] Network file system, 2013. — URL: http://linux-nfs.org/wiki/index.php/Main_Page. 18