Примерные варианты заданий для летней научной практики студентов в рамках суперкомпьютерного комплекса Московского университета и научных проектов НИВЦ МГУ

Общая часть

Доступ к суперкомпьютеру

Отработка базовых навыков доступа к суперкомпьютерам.

Практическое задание — зарегистрироваться в системе OctoShell, предназначенной для поддержки работы суперкомпьютерного центра МГУ, получить доступ к суперкомпьютерным системам Ломоносов/Ломоносов-2, запускать тестовые задачи с разными опциями на различном числе узлов/ядер.

Необходимо проанализировать особенности выполнения на основании собранной информации по запускам. Выполняется ли задача на 10 узлах суперкомпьютера быстрее в 10 раз, чем та же задача на 1 узле? Выполняется ли задача на 14 ядрах процессора-хоста суперкомпьютера быстрее в 14 раз, чем та же задача на 1 ядре? Выполняется ли задача на 28 ядрах процессора-хоста суперкомпьютера в режиме гипертрейдинга быстрее в 28 раз, чем та же задача на 1 ядре?

Индивидуальные проекты

Эволюция суперкомпьютеров

Провести анализ обновлений списка Топ50 самых мощных компьютеров России и СНГ относительно характера обновляемости суперкомпьютерных систем.

Обновленная часть систем списка состоит из новых, недавно установленных систем и систем, прошедших модернизацию. Модернизация может состоять в улучшении достигнутой производительности на той же конфигурации, в расширении системы как за счет добавления новых узлов, так и за счет изменения ранее установленных. В рейтинге суперкомпьютеров Топ50 в подробном описании систем указана конфигурация узлов, модели процессоров, ускорителей, сетей и др.

Задача — выделить категории обновлений систем и проанализировать развитие всего списка (а вышло уже 30 редакций), показав/изобразив эволюцию списка Топ50 относительно типа обновлений. Как часто встречается тот или иной тип обновлений? Насколько тот или иной тип характерен для определенной предметной области? Каково время жизни системы в списке? Каково изменение в распределении времени жизни систем по редакциям списка?

Почитать: http://top50.supercomputers.ru

Локальность использования данных

Провести исследование и написать тестовую программу, демонстрирующую существенное влияние локальности данных на производительность приложения.

Размещение данных в памяти и методы доступа к ней могут существенно влиять на производительность всего приложения. В рамках данного задания предлагается создать

тестовую программную реализацию перемножения матриц, используя различные варианты организации циклов и адресации. Программа может быть написана практически в любой среде программирования.

Датчик системного мониторинга

Написать программу-датчик, периодически отправляющий некоторый параметр состояния программно-аппаратной среды в систему мониторинга. Низкоуровневое программирование на C++.

Системы мониторинга широко используются для контроля состояния вычислительных систем, обеспечения их бесперебойной работы и анализа эффективности их использования. В зависимости задач и особенностей исследуемой платформы могут использоваться десятки датчиков. один из таких и предлагается реализовать в рамках данной работы.

Визуализация данных мониторинга

Обработка и отображение системных данных (загрузка процессора, степень параллелизма, кэш-промахи и т.п.) динамики выполнения параллельных задач.

Реализовать веб-страничку с отображением какой-то конкретной статистики по данным из уже существующей базы. Например, достать из базы интегральные характеристики за определённые периоды и отобразить распределение (можно предложить и другие характеристики, например, сколько данных передано по сети суперкомпьютера за период в пересчёт на дискеты/CD и т.п.).

Для выполнения задания: нужно знать или изучить какой-нибудь высокоуровневый язык (python/ruby), достать данные из СУБД, запустить веб-сервер, сделать веб-страничку для отображения;

ссылки:

https://www.google.com/search?q=single+page+application+python

https://www.google.com/search?q=single+page+application+ruby

Сейчас у нас в базе данных PostgreSQL хранятся интегральные данные мониторинга о каждой задаче (например, средняя загрузка процессора за время исполнения). Нужно достать данные обо всех задачах за определённый интервал и отобразить распределение задач по набору характеристик. Для начала — текстом, потом можно и графически.

Например, число задач с загрузкой CPU 0%..25% = 100, 25%..75% = 2000, 75%..100% = 200.

Файлы на суперкомпьютер

Требуется реализовать эффективное скачивание/закачивание больших файлов на суперкомпьютер.

Текущая проблема: доступ на суперкомпьютер есть только по ssh и ключу, и при скачивании больших файлов связь обрывается, т.к. на сервере действуют лимиты по процессорному времени. Требуется создать web-сервис, работающий от суперпользователя, который по одноразовой ссылке выполнит следующее:

- сбросит привилегии до пользовательских;
- позволит перемещаться по каталогам пользователя;
- позволит скачивать и закачивать файлы от имени пользователя;
- позволит менять права на файлы пользователя;
- ссылка перестаёт работать, если она один раз использовалась, причем в рамках одной сессии она активна лишь заданное время (например, сутки).

Для получения ссылки пользователь должен войти на суперкомпьютер по ssh и выполнить специальную команду, которая сформирует ссылку и выведет её на консоль.

Проект важен для облегчения работы пользователей суперкомпьютера с файлами.

Требуется знать/владеть: любым скриптовым языком (ruby/perl/python) и любым простым web-фреймворком для него, уметь создавать сессии, работать с sqlite или yaml (для хранилища ссылок). Потребуется изучить работу с UNIX Socket и SO_PEERCRED для определения имени пользователя при создании ссылки.

Предварительно почитать: ruby+sinatra или python+flask, SO_PEERCRED, UNIX Socket, sendfile(не обязательно).

Трафик суперкомпьютерных приложений

Требуется реализовать интерфейс для визуализации статистики по входящему/исходящему трафику, генерируемому отдельными пользователями суперкомпьютера.

В настоящее время есть средство, собирающее текущую статистику по трафику от каждого активного пользователя суперкомпьютера.

Его интерфейс: два файла в /proc: /proc/Traffic и /proc/Traffic_storage

При чтении у них формат одинаковый: cat /proc/Traffic

```
TRAFFIC
UID    0 bytes_in: 11644 packets_in:   167 bytes_out: 12504 packets_out:   98
UID   65 bytes_in:     0 packets_in:     0 bytes_out:     0 packets_out:    0
UID 10018 bytes_in:    0 packets_in:     0 bytes_out:     0 packets_out:    0
UID  998 bytes_in:     0 packets_in:     0 bytes_out:     0 packets_out:    0
```

TRAFFIC – заголовок, число после UID – собственно UID пользователя, к которому относится эта строка, {bytes,packets}_{in,out} – говорят сами за себя, это счетчики, т.е. значения либо увеличиваются, либо сбрасываются в 0 (см. ниже). Сейчас строчки не удаляются, т.е. если пользователь один раз засветился, то для него строчка будет до перезагрузки модуля, может быть и со всеми нулями (после сброса).

В файле /proc/Traffic - текущие значения, они обновляются в реальном времени. Если в этот файл записать строку, начинающуюся с '1', например:

```
echo 1 > /proc/Traffic
```

то атомарно будут выполнены 2 действия:
1. То, что в момент выполнения было в /proc/Traffic, перенесется в /proc/Traffic_storage
2. Все счетчики в /proc/Traffic обнулятся.
Если в любой из этих двух файлов записать строку, начинающуюся с '0'
echo 0 > /proc/Traffic_storage
то в соответствующем файле обнулятся счетчики (но все строки все равно останутся на месте).

Нужно сделать "обвязку":

- Периодически (раз в час, 15 минут, 5 минут - должно настраиваться) переносить информацию из /proc/Traffic в
- /proc/Traffic_storage, посылкой единицы в /proc/Traffic.
- Затем считать все, что есть в /proc/Traffic_storage, а там будет информация о том, кто сколько накачал за этот интервал времени, и перенести в базу данных.
- При показе из базы или суммировать объемы по пользователям по нужным интервалам, или для каждого интервала пересчитывать в скорость и строить графики.

Потребуется: работа с СУБД (PostgreSQL, MySQL) и любой фреймворк для построения сайтов (Ruby on Rails, DJango ...).

Доработка планировщика задач

Нужно создать плагин для старта программ при запуске агента slurmd на вычислительных узлах суперкомпьютера.

При помощи интерфейса SPANK написать плагин, который будет срабатывать в момент старта демона slurmd на вычислительных узлах суперкомпьютера и запускать программу, заданную в конфигурационном файле (в slurm.conf – это предпочтительно или сделать отдельный файл). Фактически, нужно повторить код, которые есть в slurm для запуска прологов/эпилогов для работы в другом месте.

Потребуется: уметь программировать на Си, собирать сторонние программы по инструкции, тестировать полученное.

Что почитать:

- описание SPANK https://slurm.schedmd.com/spank.html
- код для пролога: https://github.com/SchedMD/slurm/blob/master/src/slurmctld/job_scheduler.c#L4379