A stylized, light-colored illustration of a plant with several leaves and a cluster of small, round buds or flowers, positioned on the left side of the page against a dark brown background.

РОБОТА З СУПЕРКОМП'ЮТЕРОМ ІНСТИТУТУ КІБЕРНЕТИКИ НАН УКРАЇНИ

підготував студент 402 групи
Сорочан Олександр

Чернівці – 2015

1. Загальні відомості

- Суперкомп'ютер – спеціалізована обчислювальна машина, яка значно переважає за своїми параметрами та швидкістю обрахунків більшість існуючих комп'ютерів.
- Як правило, сучасні суперкомп'ютери насправді є великим числом потужних серверних комп'ютерів – «вузлів», з'єднаних між собою високошвидкісною магістраллю для отримання максимальної продуктивності. Такі суперкомп'ютери називають ще обчислювальними кластерами.
- Найпоширенішим є використання однорідних кластерів, тобто таких, де всі вузли абсолютно однакові за своєю архітектурою й продуктивністю.

- Основним завданням суперкомп'ютера є запуск на ньому паралельних програм, тобто програм, призначених для запуску одразу на декількох процесорах (ядрах).
- Це може стати корисним у разі проведення великої кількості обчислень. Розділивши набір операцій між декількома процесорами, можна добитися більшої продуктивності, програма працюватиме менше часу.
- У такому разі кажуть, що процес, породжений такою програмою, складається з декількох потоків, які виконуються паралельно, тобто без наперед визначеного порядку за часом.
- Утворення декількох потоків часто призводить до додаткової задачі координації роботи між потоками, однак кінцевий результат виправдовує ці додаткові витрати.

- Сайт суперкомп'ютера Інституту кібернетики НАН України – <http://icybcluster.org.ua/>

ua en

Суперкомп'ютер ІК НАН України

Головна | Документація | Віртуальна організація GRIDIK | Контакти

Вхід

Головна

Нова черга для малих задач

15.10.2014

Шановні користувачі!

На кластері SKIT створено нову чергу scit4_it для задач, час виконання яких не перевищує 1 годину.

Модернізація системного ПЗ

15.10.2014

Шановні користувачі!

В черзі scit4 проводиться модернізація системного програмного забезпечення, тому вона тимчасово не приймає задачі.

Модернізація системи керування суперкомп'ютером

26.08.2014

Шановні користувачі!

На сайті модернізовано систему керування суперкомп'ютером SCMS. З питань роботи нової системи звертайтеся до адміністраторів кластера.

Український суперкомп'ютерний Інтернет-дайджест

 NVIDIA представляє найшвидший в світі прискорювач Tesla K80 19.11.2014

 Замітки з конференції

- Різноманітна інформація про обчислювальний кластер знаходиться у розділі «Документація»



The image shows a screenshot of the website for the Supercomputer of the Institute of Cybernetics of the National Academy of Sciences of Ukraine. The main header features the text 'Суперкомп'ютер ІК НАН України' and a globe icon. A red arrow points to the 'Документація' (Documentation) menu item in the navigation bar. Below the navigation bar, the 'Документація' section is expanded, showing a list of links: 'Про кластери', 'GPU-кластер', 'Інструкції для користувачів', 'Роботи користувачів', 'Прикладне програмне забезпечення', 'Публікації', 'Історія обчислювальної техніки', 'Статті', 'Відео', and 'Навчальні матеріали'. The 'Вхід' (Login) section is also visible. The main content area displays the title 'Обчислювальний комплекс СКІТ ІК НАН України' and 'СКІТ - 4'. It describes a 28-node cluster based on HP ProLiant Gen8 BladeSystems, featuring Intel Xeon E5-2600 processors, NVIDIA Tesla M2075 GPUs, and Infiniband FDR 56 Gbit/s network. The cluster has 448 cores, 36 GPUs, and 1.8 TB of memory. It is integrated with a 120 TB data storage system. Performance metrics include a peak of 30 TFlops and a real performance of 18 TFlops. The cluster is also integrated with a Lustre file system.

ua en

Суперкомп'ютер ІК НАН України

Головна | **Документація** | Віртуальна організація GRIDIK | Контакти

Документація

- Про кластери
- GPU-кластер
- Інструкції для користувачів
- Роботи користувачів
- Прикладне програмне забезпечення
- Публікації
- Історія обчислювальної техніки
- Статті
- Відео
- Навчальні матеріали

Вхід

Обчислювальний комплекс СКІТ ІК НАН України

СКІТ - 4

28-вузловий кластер на багатоядерних процесорах Intel Xeon E5-2600

Кластер базується на новітній платформі HP ProLiant Gen8 BladeSystems та має наступні характеристики:

- кластер складається з 28 вузлів на базі центральних процесорів Intel Xeon E5-2600 з частотою 2.6 ГГц, має 448 обчислювальних ядер, 36 прискорювачів Nvidia Tesla M2075, 1,8 ТБ оперативної пам'яті;
- інтегрований із загальним сховищем даних кластерного комплексу обсягом 120 ТБ;
- мережа передачі даних між вузлами Infiniband FDR 56 Гбіт/с.

Суперкомп'ютер у своєму складі має як класичні вузли з центральними процесорами, так і гібридні вузли з графічними прискорювачами.

Кожен вузол має 16 ядер (32 у режимі HyperThreading), 64 ГБ оперативної пам'яті. Гібридні вузли додатково мають 3 прискорювача Nvidia Tesla M2075.

В цілому кластер має наступні характеристики продуктивності:

- пікова продуктивність 30 ТФлопс;
- реальна продуктивність 18 ТФлопс.

СКІТ-4 інтегрований із високопродуктивним сховищем даних об'ємом 120 ТБ на основі паралельної файлової системи Lustre.

- Найбільш корисним є розділ «Інструкція для користувачів»

ua en

Суперкомп'ютер ІК НАН України

Головна | **Документація** | Віртуальна організація GRIDIK | Контакти

Документація

- Про кластери
- GPU-кластер
- Інструкції для користувачів
- Роботи користувачів
- Прикладне програмне забезпечення
- Публікації
- Історія обчислювальної техніки
- Статті
- Відео
- Навчальні матеріали

Вхід

Інструкції для користувачів

Приклад запуску Gromacs на кластері SKIT

В якості прикладу використовуємо gmxbench з домашнього сайту gromacs.

Компіляція та запуск MPI додатків на розділі scit4

Як можна виконати компіляцію та запуск MPI версії програм за допомогою MVARICH2 на розділі кластера scit4.

Написання MPI-програми, її компіляція та запуск на кластері

Розглянемо приклад написання найпростішої MPI-програми, її компіляцію та запуск на кластері.

Запуск задач на вузлах із заданною кількістю ядер

Іноді потрібно запустити задачу не на будь-яких вузлах, а тільки з певною кількістю ядер, наприклад, 4-ядерних.

Інструкція для користувачів Firefly

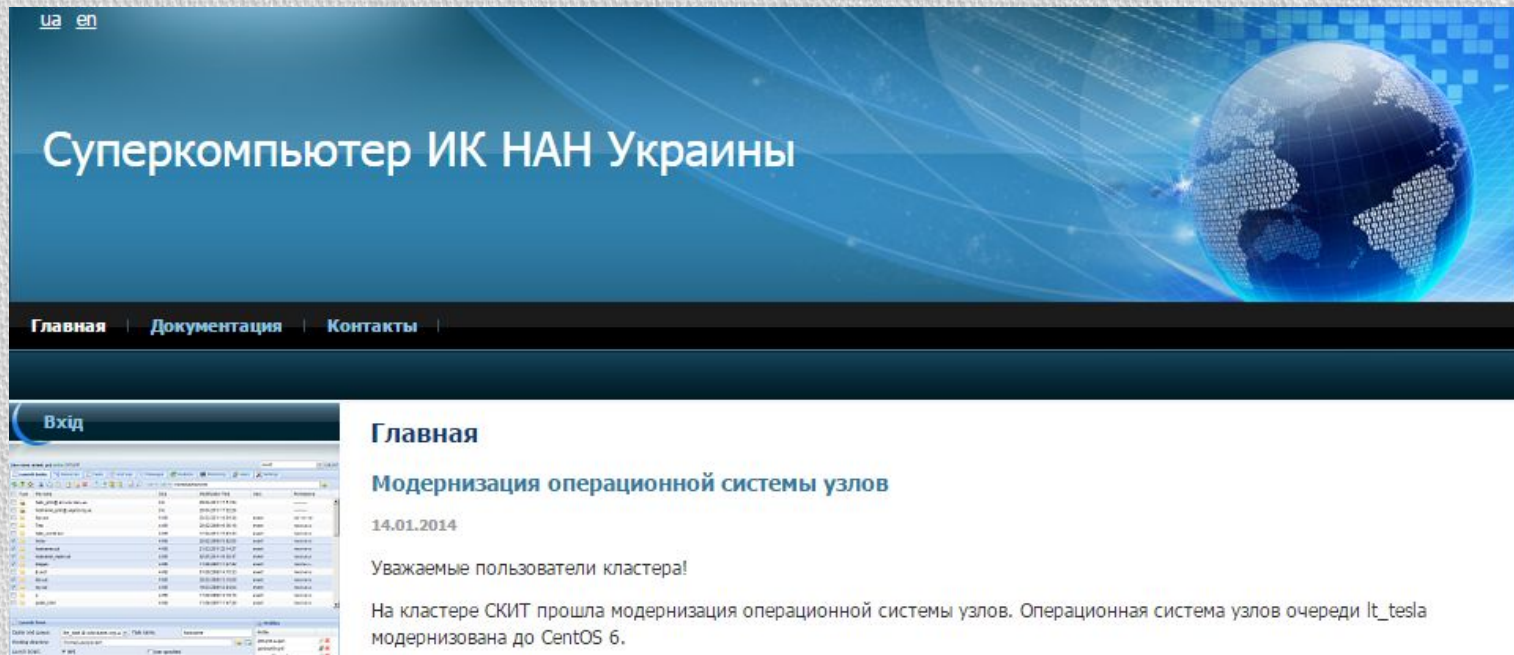
У нас є профіль запуску для пакету ps-games (firefly), але на жаль ліцензійна політика розробника потребує окремої реєстрації від кожного користувача.

Режим MPI-None пакетної обробки даних

Режим MPI-None запуску задачі дозволяє організувати запуск не-MPI програм та сформувати пакет із декількох послідовних операцій над даними.



- На жаль, сайт суперкомп'ютера не є надто зручним, а матеріали сайту часто не дублюються між різними мовами сайту. Деякі додаткові матеріали можна знайти на російськомовній версії сайту: http://icybcluster.org.ua/index.php?lang_id=1 – яка навіть не доступна з україномовної чи англomовної версії.
- Також варто відзначити, що і ця версія сайту не охоплює усі можливості роботи з суперкомп'ютером.



- Суперкомп'ютер Інституту кібернетики складається з чотирьох обчислювальних кластерів: СКІТ-1, СКІТ-2, СКІТ-3, СКІТ-4. З них перші два кластери застаріли і уже не використовуються.
- СКІТ-3 – 127-вузловий кластер на багатоядерних процесорах (75 вузлів на 2-ядерних процесорах Intel Xeon 5160 та 52 вузла на 4-ядерних процесорах Intel Xeon 5345).
- Тактова частота – 3,0 ГГц та 2,2 ГГц відповідно. Число процесорів у вузлі кластера – 2.
- Оперативна пам'ять вузла – 2 ГБ на ядро, відповідно, 8 та 16 ГБ.
- Число ядер процесорів у вузлі – 4 та 8. Всього у кластері 716 ядер.
- СКІТ-3 інтегрований із системою зберігання даних типу RAID5 на основі паралельної файлової системи Lustre обсягом 20 ТБ.
- Продуктивність кластера – 7500 ГФлопс (номінальна), 5317 ГФлопс (підтверджена).

- SKIT-4 – 28-вузловий кластер на 16-ядерних процесорах Intel Xeon E5-2600.
- Тактова частота – 2,6 ГГц. Число процесорів у вузлі кластера – 1.
- У режимі Hyper-threading відбувається імітація 32-ядерних процесорів.
- Оперативна пам'ять вузла – 64 ГБ.
- Всього у кластері 448 ядер.
- SKIT-4 інтегрований із високопродуктивним сховищем даних об'ємом 120 ТБ на основі паралельної файлової системи Lustre.
- З 28 вузлів 12 додатково мають по 3 графічні прискорювачі nVidia Tesla M2075.
- Завдяки наявності графічних процесорів nVidia продуктивність кластера становить 30 ТФлопс (номінальна) та 18 ТФлопс (реальна). В той же час на цьому кластері не рекомендується запускати задачі, які не використовують ресурс графічних прискорювачів.

2. Про паралельні технології

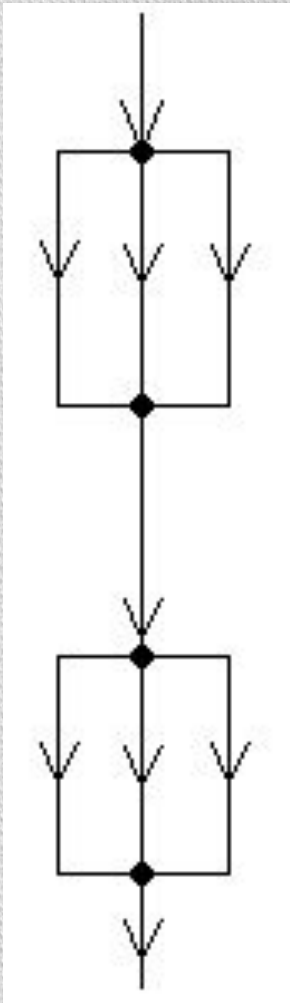
- Існує декілька способів зайняти обчислювальні потужності кластера:
 - **1. Запускання багатьох однопроцесорних завдань.** Це може бути сприятливим варіантом, якщо потрібно провести багато незалежних обчислювальних експериментів з різними вхідними даними, причому час проведення кожного окремого розрахунку не має значення, а всі дані розміщуються в об'ємі пам'яті, доступному одному процесу.
 - **2. Викликати у своїх програмах паралельні бібліотеки.** Для деяких областей, наприклад, лінійна алгебра, доступні бібліотеки, які дозволяють вирішувати широке коло стандартних підзадач з використанням можливостей паралельної обробки.

- Якщо звертання до таких підзадач становить більшу частину обчислювальних операцій програми, то використання такої паралельної бібліотеки дозволить одержати паралельну програму практично без написання власного паралельного коду. Прикладом такої бібліотеки є ScaLAPACK, яка доступна для використання на кластері.
- **3. Створювати власні паралельні програми.** Це найбільш трудомісткий, але й найбільш універсальний спосіб. Існує кілька варіантів такої роботи, зокрема , вставляти паралельні конструкції в готові паралельні програми або створювати з "нуля" паралельну програму.
- Надалі розглядатиметься саме останній варіант.

- На суперкомп'ютері встановлені компілятори мов C, C++ та Fortran.
- Можна встановлювати компілятори й інших мов, але тільки у свій домашній каталог. Варто відзначити також слабку підтримку паралельних технологій, встановлених на суперкомп'ютері, іншими мовами. Також не гарантується стабільність роботи таких компіляторів на обчислювальному кластері.
- Для усіх вищезазначених мов встановлені відкриті компілятори GCC та комерційні Intel.
- Версії компіляторів GCC: 3.4.6, 4.1.2, 4.4.7
- Версії компіляторів Intel: 10.1, 12.1.6, 13.0.1, 13.1.1, 13.1.3

- Суперкомп'ютер підтримує такі паралельні технології:
- 1. **MPI**. Програмний інтерфейс для передачі інформації, який дозволяє обмінюватися повідомленнями між процесами, які виконують одну задачу.
- 2. **CUDA**. Програмно-апаратна архітектура паралельних обчислень, яка дозволяє суттєво збільшити продуктивність обрахунків завдяки використанню графічних процесорів nVidia.
- 3. **OpenCL**. Фреймворк для написання паралельних програм на різноманітних графічних та центральних процесорах.
- 4. **OpenMP**. Відкритий стандарт для розпаралелювання програм на системах з загальною пам'яттю.
- Надалі буде розглянуто такі класичні технології, як OpenMP та MPI.

2.1. OpenMP



- OpenMP використовується на системах із загальною пам'яттю, тобто коли різні потоки можуть звертатися до однієї і тієї самої ділянки пам'яті.
- За допомогою директив OpenMP можна вказати компілятору, що деякі ділянки коду необхідно виконувати паралельно, декількома потоками (див. мал.).
- Перевагою цієї технології є те, що програму, написану для виконання на одному процесорі, можна легко перетворити на паралельну, дописавши лише декілька рядків коду.

- OpenMP підтримується, зокрема, такими компіляторами:
 - компілятори GCC, починаючи з версії 4.2;
 - Visual C++ 2005 та 2008 у редакціях Professional та Team System; 2010 – у редакціях Professional, Premium та Ultimate; починаючи з 2012 – у всіх версіях;
 - комерційні компілятори Intel, починаючи з версії 10.1 – зараз надаються у складі продукту Intel Parallel Studio.
- Щоби скомпілювати програму, написану з допомогою OpenMP, компілятором GCC, необхідно додати ключ `/fopenmp`; у випадку Visual C++ - `/openmp`; Intel – `/Qopenmp`
- Для останніх двох компіляторів, у разі використання середовища Microsoft Visual Studio (Intel Parallel Studio інтегровується у склад Microsoft Visual Studio) цю опцію можна ввімкнути у налаштуваннях проекту.

- Для демонстрації роботи OpenMP наведемо програму обчислення числа π на мові C++ за наступною формулою:

$$\pi \approx \frac{1}{n} \sum_{i=1}^n \frac{4}{1 + x_i^2}, \quad x_i = \frac{i - 0.5}{n}$$

```
double x = 0;
double sum = 0;
const int num_steps = 1000;
double step = 1.0 / num_steps;
#pragma omp parallel for private(x) reduction(+:sum)
for (int i = 0; i < num_steps; i++) {
    x = ( (double)i + 0.5) * step;
    sum += 4.0 / (1.0 + x * x);
}
sum *= step;
```


- У код послідовної програми вставлено лише одну стрічку:
`#pragma omp parallel for private(x) reduction(+:sum)` –
і вона стає паралельною.

- Деякі пояснення:

- Усі директиви OpenMP включаються у C та C++ за допомогою директиви препроцесора `#pragma` з подальшою інструкцією `omp`
- Усі змінні, оголошені до директив OpenMP вважаються глобальними для усіх потоків; всередині – локальними для кожного потоку.
- Область видимості змінної можна змінити. Наприклад, за допомогою інструкції `private` змінна `x` стала локальною для кожного потоку.
- Інструкція `parallel for` вказує на те, що ітерації циклу розділяються між декількома потоками, наприклад, дії для $i = 0 .. 249$ виконуватимуться першим потоком, $i = 250 .. 499$ – другим потоком і т. д.
- Існують й окремі інструкції `parallel` та `for`.

- Інструкція `parallel` вказує на те, що код, записаний у наступному блоці виконуватиметься одночасно усіма потоками.
- Інструкція `for` працює лише у вкладеному блоці `parallel` і означає розділення ітерацій циклу між різними потоками.
- Оператор редукції `reduction` робить одразу декілька речей. Так, у цьому прикладі, змінна `sum` стає локальною для кожного потоку, а після виконання циклу (як і паралельного блоку) усі локальні значення змінної додаються (+), а результат зберігається у тій же змінній.

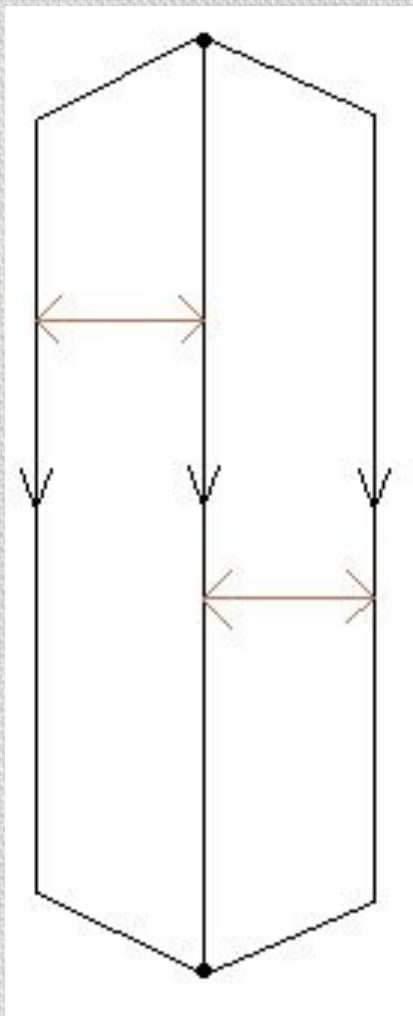
• Кількість потоків можна визначити змінною середовища `OMP_NUM_THREADS`, яка встановлюється у командному рядку:

```
set OMP_NUM_THREADS=4,
```

• або викликом функції `omp_set_num_threads` із заголовного файлу `omp.h`:

```
omp_set_num_threads(4);
```


2.2. MPI



- Message Passing Interface (MPI) використовується на системах з роздільною пам'яттю, коли кожному процесору виділено свою область пам'яті.
- На відміну від OpenMP, MPI передбачає існування декількох потоків з самого початку роботи програми. Протягом роботи програми потоки можуть комунікувати між собою за допомогою механізму «повідомлень» (див. мал.).
- Така особливість роботи MPI не дозволяє переробити послідовну програму у паралельну, натомість доводиться повністю змінювати логіку програми.

- MPI є найбільш розповсюдженим стандартом інтерфейсу обміну даними в паралельному програмуванні та є найбільш природною технологією для суперкомп'ютера Інституту кібернетики. Багато паралельних бібліотек, встановлених на суперкомп'ютері, базуються саме на MPI.
- Сам по собі MPI не є готовим програмним продуктом, а є лише набором правил, за якими мають працювати реалізації цієї технології. Серед її реалізацій виділяють відкриті MPICH та OpenMPI, безкоштовну Microsoft MPI та комерційну Intel MPI.
- Компіляція та запуск програм, написаних з допомогою MPI, зазвичай супроводжується великими труднощами: необхідно підключати додаткові заголовні файли та файли бібліотек, а запуск програми виконується через додатковий файл, наприклад з іменем trihexes.exe, де скомпільована програма передається одним із параметрів. Про ці налаштування найкраще прочитати в офіційних документаціях відповідних реалізацій.

- Для порівняння технологій знову наводиться програма обчислення числа π , але вже з використанням MPI.

```
#include "mpi.h"
#include <iostream>
using namespace std;
int main(int argc, char *argv[]) {
    int num_steps, myid, numprocs, i;
    double pi, step, sum, x;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    if (myid == 0) {
        num_steps = 1000;
    }
    MPI_Bcast(&num_steps, 1, MPI_INT, 0, MPI_COMM_WORLD);
```



```
step = 1.0 / num_steps;
sum = 0.0;
for (int i = myid; i < num_steps; i += numprocs) {
    x = step * ( (double)i + 0.5);
    sum += 4.0 / (1.0 + x * x);
}
sum *= step;
MPI_Reduce(&sum, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
if (myid == 0) {
    cout << pi;
}
MPI_Finalize();
return 0;
}
```

- Деякі пояснення:

- Абсолютно усі команди MPI мають знаходитися між функціями MPI_Init, у яку передаються параметри командного рядка, та MPI_Finalize. Знову відкрити секцію MPI після MPI_Finalize неможливо.

- Після виконання функції `MPI_Init` створюється група потоків, кожний з яких виконуватиме один і той самий код, написаний після цієї функції.
- Майже в усіх функціях MPI присутній параметр зі значенням `MPI_COMM_WORLD`. Це спеціальний об'єкт MPI, який називається комунікатором. Комунікатори об'єднують у собі певні групи потоків. Комунікатори дозволяють контролювати, щоби повідомлення передавалися лише між процесами з одного комунікатора. Після виклику функції `MPI_Init` автоматично створюється комунікатор `MPI_COMM_WORLD`, який об'єднує у собі усі доступні потоки.
- Функція `MPI_Comm_size` дозволяє визначити кількість потоків у комунікаторі, а `MPI_Comm_rank` – порядковий номер потоку. Нумерація починається з 0.
- Функція `MPI_Bcast` є функцією широкомовного обміну. Після її проведення у всіх потоках з деякого комунікатора (останній параметр) буде однакове значення деякої змінної (1-ий параметр). Значення береться з потоку, номер якого задається 4-им параметром. Другий та третій параметри визначають кількість і тип даних (у формі `MPI_INT`, `MPI_DOUBLE` тощо), що передаються.

- На відміну від OpenMP, де компілятор сам забезпечував розподіл ітерацій циклу між потоками, у MPI контроль за цим покладається на користувача. Розповсюдженою є така схема: початковим значенням змінної циклу є номер потоку, зміна йде не на 1, а на кількість потоків у програмі. Таким чином, у випадку чотирьох потоків потоку з порядковим номером 0 відповідатимуть значення 0, 4, 8, ...;
1: 1, 5, 9, ...; 2: 2, 6, 10, ...; 3: 3, 7, 11, ... Як бачимо, така схема забезпечує відсутність повторень індексу циклу між потоками.
- Так само, як і в OpenMP, у MPI визначено операцію редукції. За це відповідає функція `MPI_Reduce`. В якості параметрів передаються змінна, значення якої збиратимуться з потоків; змінна, куди запишеться результат, кількість і тип даних; операція збору даних (`MPI_SUM` – сума, `MPI_PROD` – добуток тощо); номер потоку, куди запишеться результат, та комунікатор.

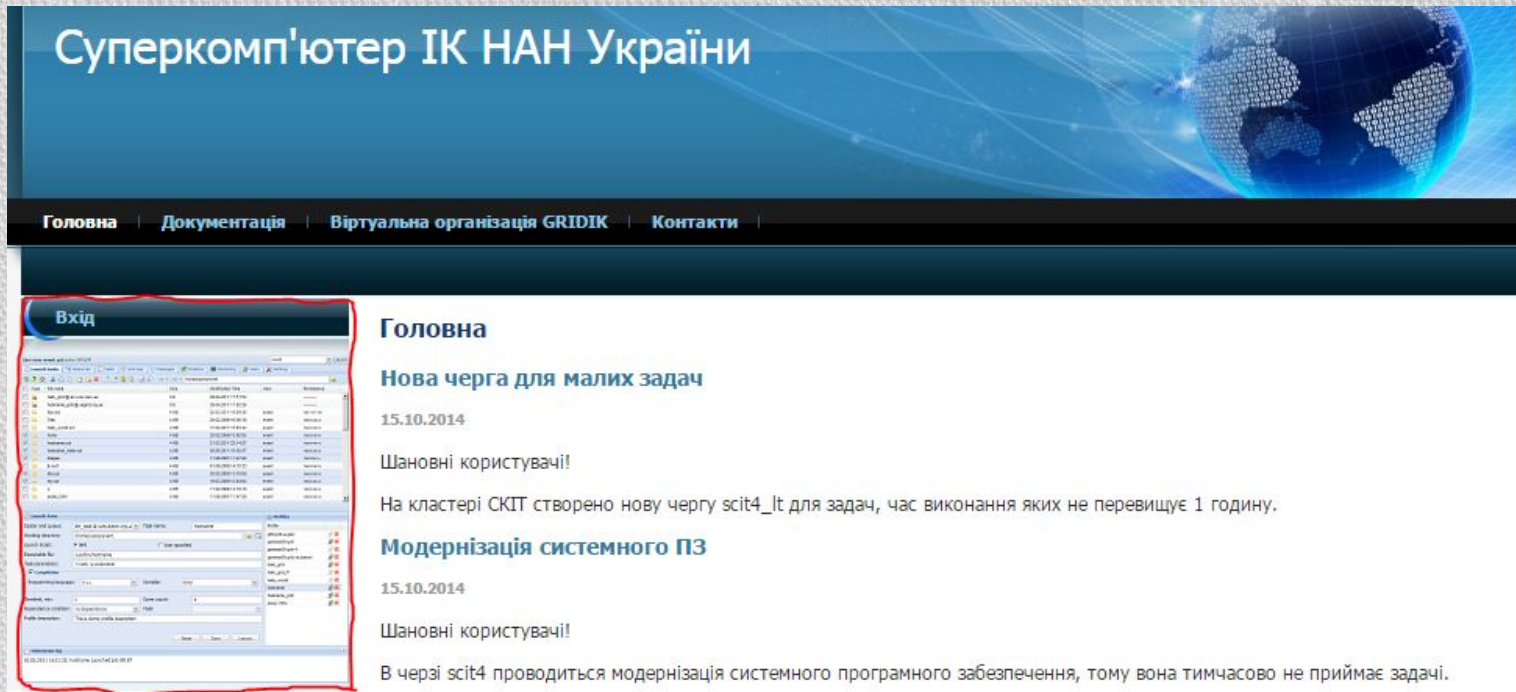
- У цьому прикладі механізм передачі повідомлень застосовано неявно – у функціях `MPI_Bcast` та `MPI_Reduce`. Стандарт `MPI` передбачає значно ширші можливості, як наприклад, передача даних між двома конкретними потоками (з'єднання «точка–точка»), розсилка та збір даних від кількох потоків тощо.
- Запуск скомпільованої програми, написаної за допомоги `MPI`, може запускатися таким чином:

```
<шлях до папки з mpiexec>\mpiexec.exe -np <кількість потоків>  
<шлях до скомпільованої програми>
```

- Більше про `MPI` можна дізнатися, наприклад, у наступній книзі:
Г.И. Шпаковский, Н.В. Серикова «Программирование для многопроцессорных систем в стандарте `MPI`»

3. Робота з суперкомп'ютером

- Для роботи із суперкомп'ютером доступні 2 інтерфейси: графічний через веб-браузер та консольний.
- Щоби зайти на суперкомп'ютер із веб-браузера, необхідно зайти за посиланням «Вхід» на сайті суперкомп'ютера <http://icybcluster.org.ua/>

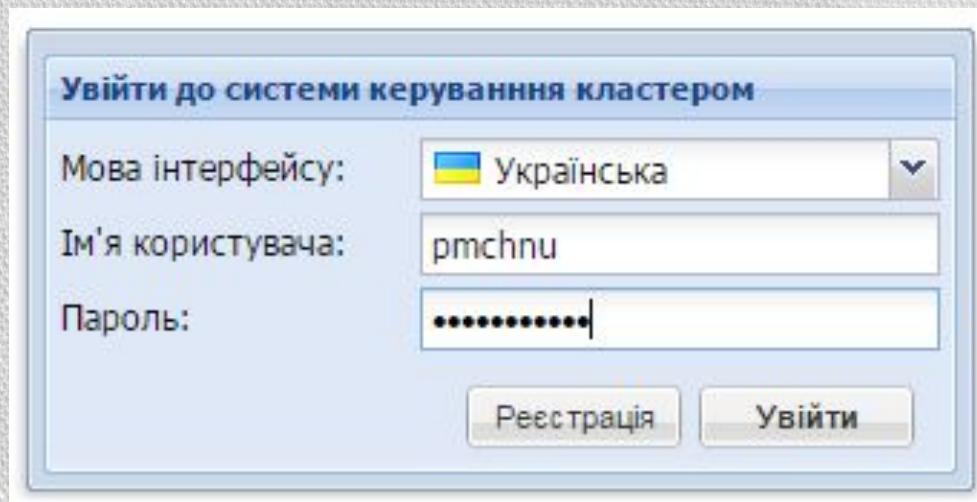


The screenshot shows the website for the Icybcluster supercomputer. The header features the text "Суперкомп'ютер ІК НАН України" and a navigation menu with links for "Головна", "Документація", "Віртуальна організація GRIDIK", and "Контакти". The main content area is titled "Вхід" (Login) and contains a table of system resources. Below the login section, there are two news items:

Головна
Нова черга для малих задач
15.10.2014
Шановні користувачі!
На кластері СКІТ створено нову чергу scit4_it для задач, час виконання яких не перевищує 1 годину.

Модернізація системного ПЗ
15.10.2014
Шановні користувачі!
В черзі scit4 проводиться модернізація системного програмного забезпечення, тому вона тимчасово не приймає задачі.

- Необхідно буде ввести облікові дані:



Увійти до системи керування кластером

Мова інтерфейсу:

Ім'я користувача:

Пароль:

- Відкриється наступна сторінка:

SCMS.pro Ім'я користувача: pmchnu Вихід

Тип	Ім'я файла	Розмір	Час зміни	Користувач	Права
	OpenMP.sh	64 b	18.02.2015 14:50:08	pmchnu	rw-r-x-w-
	SLAR	4 KB	01.03.2015 14:22:16	pmchnu	rw-rw---x
	SLAR3DIAG	4 KB	23.02.2015 19:02:29	pmchnu	rw-r--r-x
	intel	4 KB	18.02.2015 14:43:24	pmchnu	rw-r--r-x

Форма запуску

Кластер та черга:
 Ім'я задачі:

Робоча тека:

Скрипт запуску:
 MPI
 Заданий користувачем

Файл для виконання:

Параметри задачі:

Компіляція

Ліміт часу, хв:
 Кількість ядер: / на вузол

Умова залежності:
 Задача:

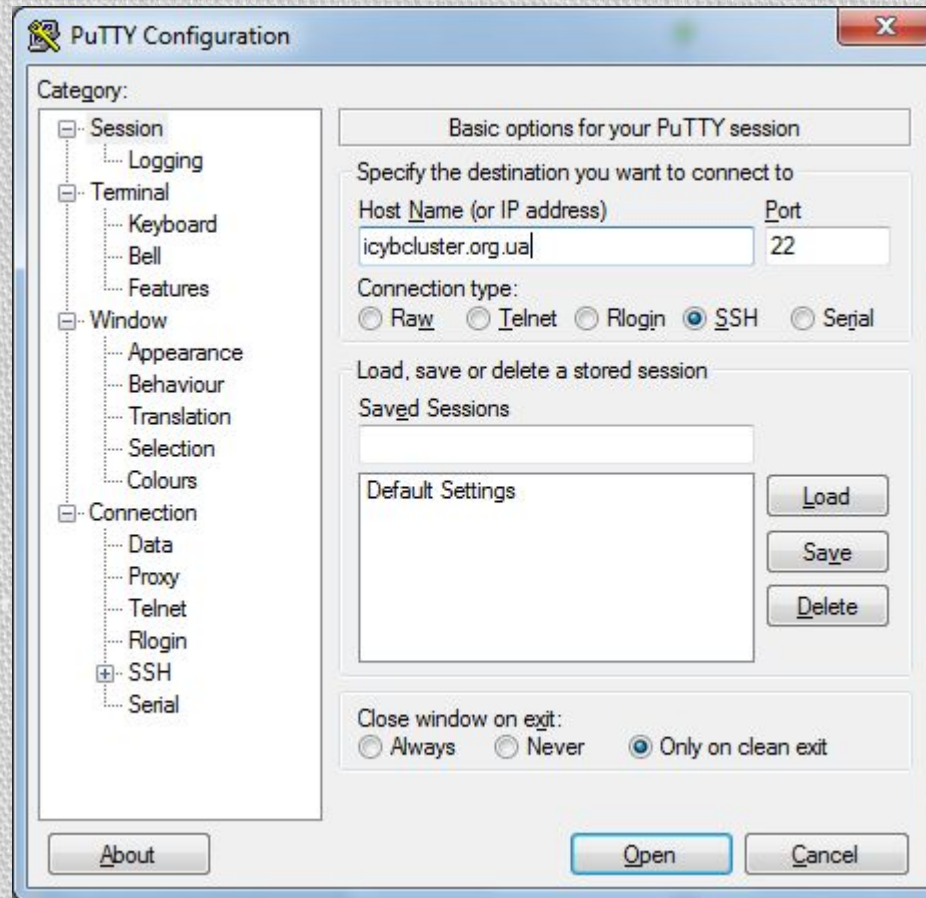
Профілі

Профіль			
games-2009-scit3			
grid-hostname			
grid-sleep			
hostname			
sleep			

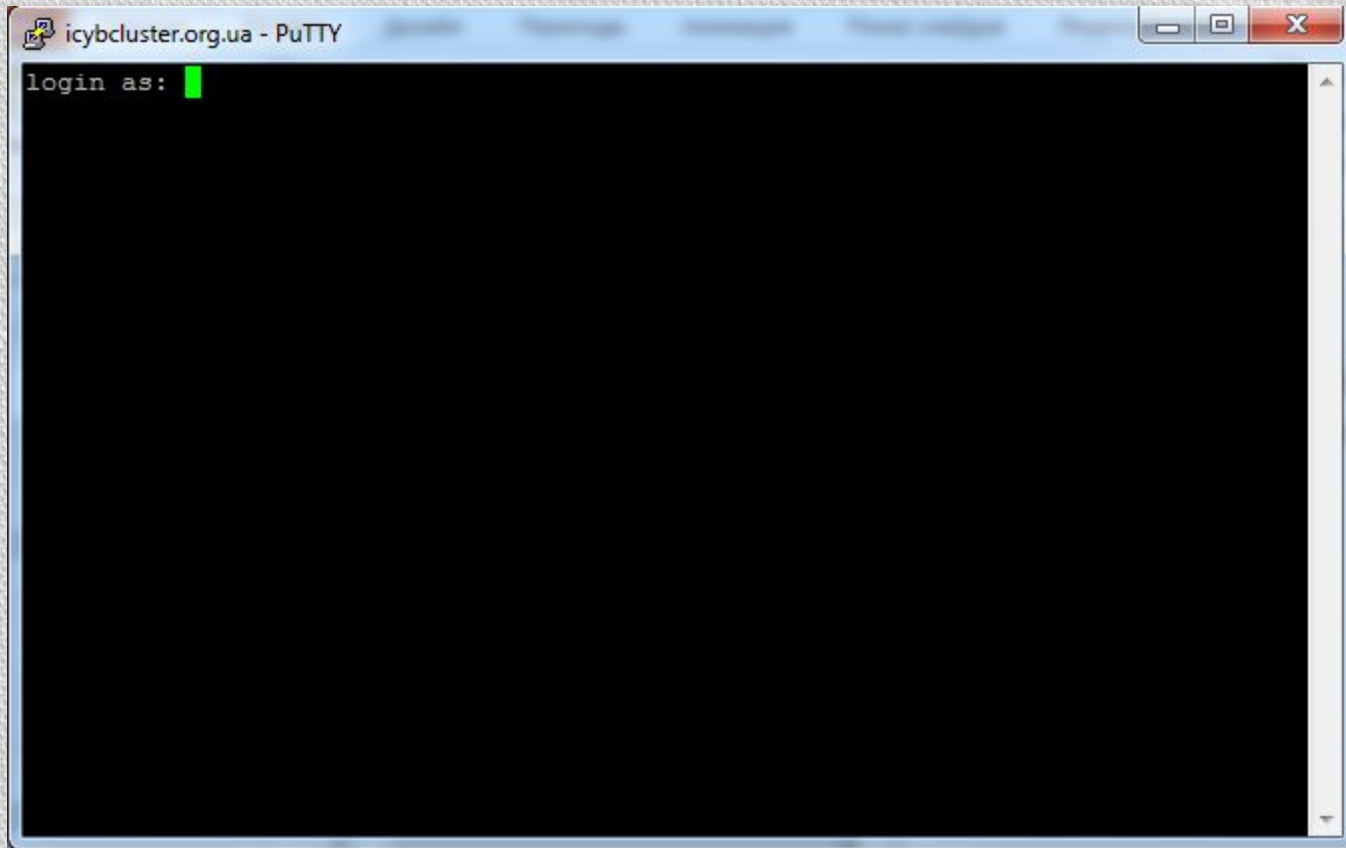
- Зверху розташоване меню, звідки можна перемикатися між різними вікнами стану суперкомп'ютера.
- На відкритій закладці «Запуск задач» розташовано деяку подібність «Провідника», форму запуску задач та журнал запусків.
- Відкривати папки і файли можна натисканням на відповідну піктограму у другому стовпчику.

- На жаль, графічний інтерфейс суперкомп'ютера недосконалий. Досить часто легше працювати з суперкомп'ютером у консольному режимі.
- Встановлення із суперкомп'ютером здійснюється через протокол SSH.
- У Linux досить виконати команду
`ssh pmchnu@icybcluster.org.ua`
- та ввести пароль для з'єднання із суперкомп'ютером.

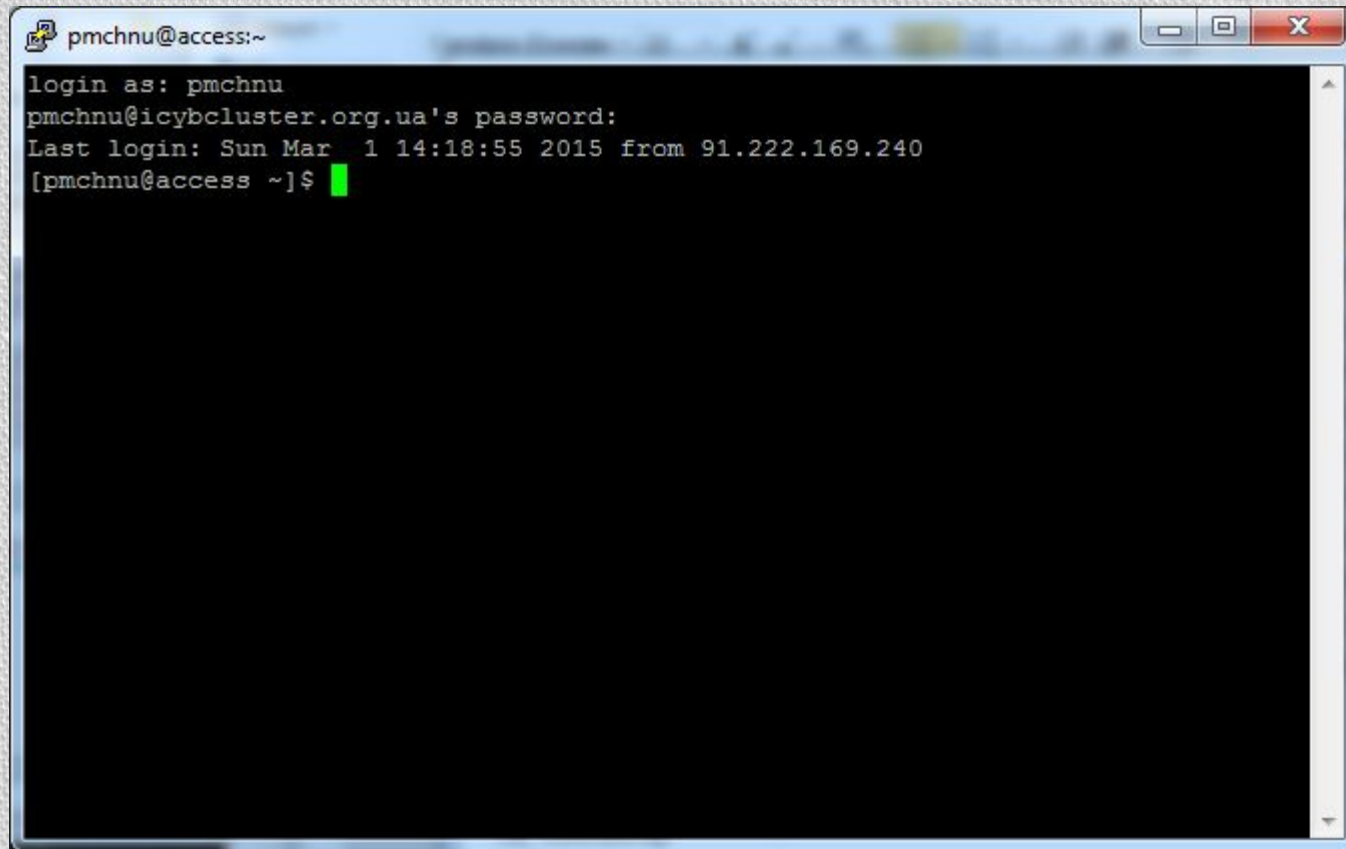
- У Windows доводиться користуватися сторонніми програмними засобами, наприклад, PuTTY
(<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>)



- У полі Host Name необхідно ввести адресу `icybcluster.org.ua` та натиснути кнопку Open, після чого відкриється наступне вікно:

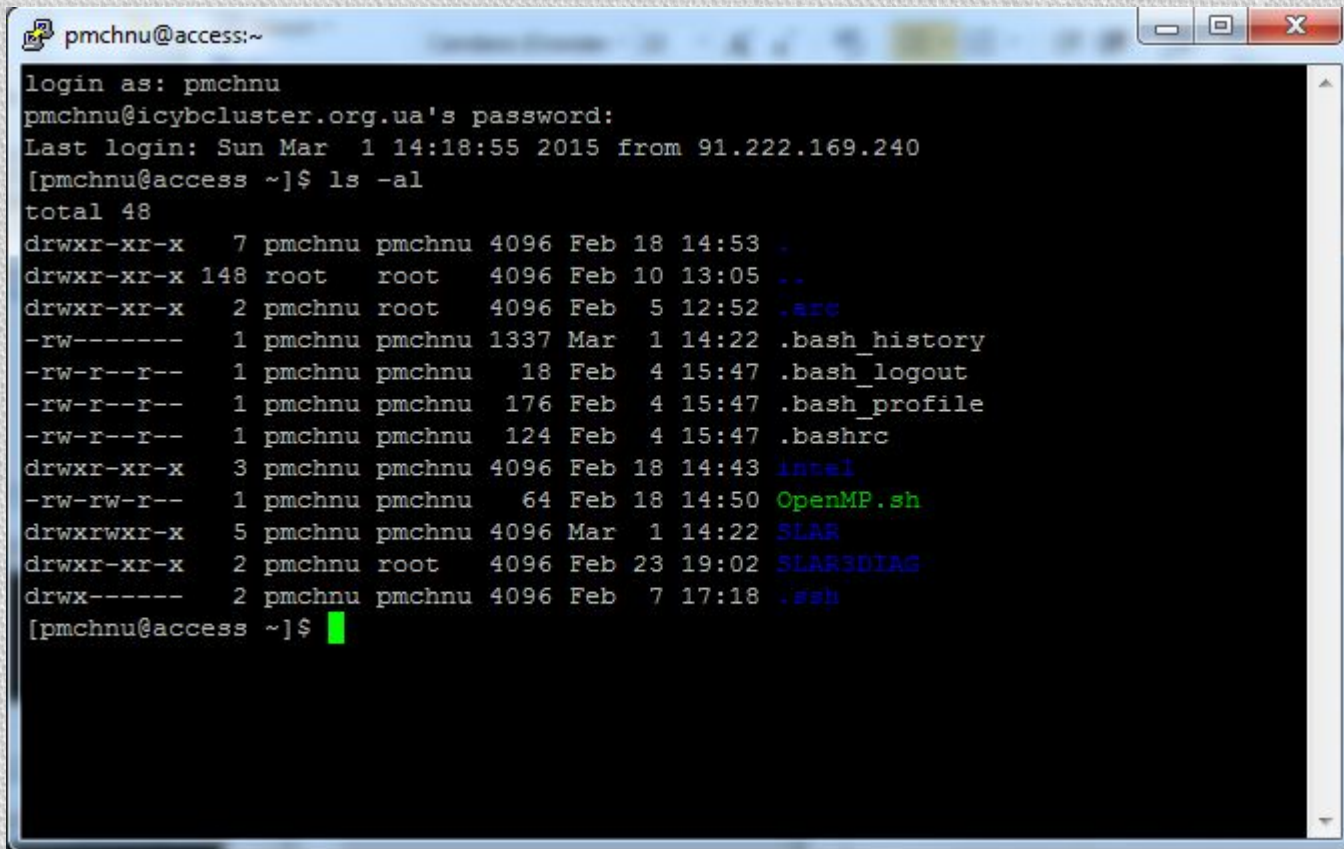


- Необхідно ввести логін та пароль, після чого можна буде працювати із суперкомп'ютером:



```
pmchnu@access:~  
login as: pmchnu  
pmchnu@icybcluster.org.ua's password:  
Last login: Sun Mar 1 14:18:55 2015 from 91.222.169.240  
[pmchnu@access ~]$
```


- На суперкомп'ютері встановлена операційна система CentOS Linux, тому у консольному режимі працюють усі стандартні команди Linux, наприклад, команда `ls -al` виведе вміст поточної папки:



```
pmchnu@access:~  
login as: pmchnu  
pmchnu@icybcluster.org.ua's password:  
Last login: Sun Mar  1 14:18:55 2015 from 91.222.169.240  
[pmchnu@access ~]$ ls -al  
total 48  
drwxr-xr-x  7 pmchnu pmchnu 4096 Feb 18 14:53 .  
drwxr-xr-x 148 root   root   4096 Feb 10 13:05 ..  
drwxr-xr-x  2 pmchnu root   4096 Feb  5 12:52 .src  
-rw-----  1 pmchnu pmchnu 1337 Mar  1 14:22 .bash_history  
-rw-r--r--  1 pmchnu pmchnu   18 Feb  4 15:47 .bash_logout  
-rw-r--r--  1 pmchnu pmchnu 176 Feb  4 15:47 .bash_profile  
-rw-r--r--  1 pmchnu pmchnu 124 Feb  4 15:47 .bashrc  
drwxr-xr-x  3 pmchnu pmchnu 4096 Feb 18 14:43 intel  
-rw-rw-r--  1 pmchnu pmchnu   64 Feb 18 14:50 OpenMP.sh  
drwxrwxr-x  5 pmchnu pmchnu 4096 Mar  1 14:22 SLAR  
drwxr-xr-x  2 pmchnu root   4096 Feb 23 19:02 SLAR3DIAG  
drwx-----  2 pmchnu pmchnu 4096 Feb  7 17:18 .ssh  
[pmchnu@access ~]$
```


- Папка `/home/users/rtchnu` є домашньою, тобто зайшовши під іменем `rtchnu`, користувач отримує повний доступ над файлами у цій папці (читання і запис), над іншими папками можна здійснювати лише операцію читання.
- До домашньої папки можна звертатися через символ `~`, як і будь-якому Linux.
- Основним програмним компонентом суперкомп'ютера є менеджер ресурсів SLURM. Саме за допомогою нього контролюється набір задач, запущених на комп'ютері, та надається необхідна для задачі кількість процесорів.
- Задачі виконуються на суперкомп'ютері за принципом черги, причому на SKIT-3 та SKIT-4 організовано по 2 черги: `lite_task` та `scit3` на SKIT-3; `scit4_It` та `scit4` на SKIT-4. Черги `scit3` та `scit4` призначені для виконання довготермінових задач. Ліміт часу виконання на них складає 21 день. Черги `lite_task` та `scit4_It` призначені для легких завдань. Ліміт часу складає 1 годину.

- Щоби подивитися на завантаженість черг у графічному режимі, треба зайти на закладку «Ресурси»:

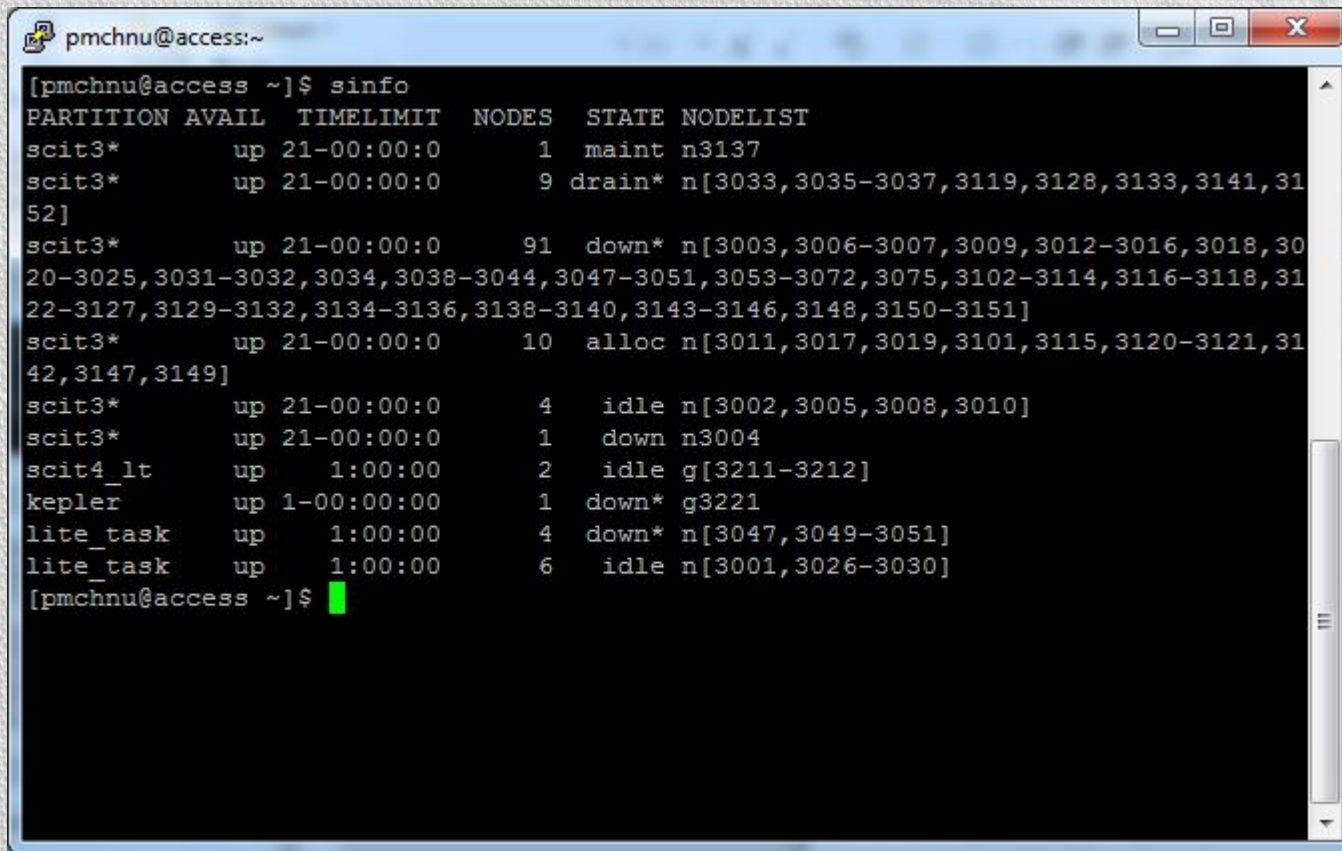
SCMS pro Ім'я користувача: pmchnu. Вихід

[Залучити задачі](#)
[Ресурси](#)
[Задачі](#)
[Мапа ґрида](#)
[Повідомлення](#)
[Налаштування](#)

[Оновити](#)
[Докладно](#)

Розділ	Стан	Вузлів	Ядер	Ліміт часу	Доступний
Кластер: icybcluster.org.ua					
scit3	maint	1	8	21 0:0:0	Доступний
scit3	drain*	9	56	21 0:0:0	Доступний
scit3	down*	91	520	21 0:0:0	Доступний
scit3	alloc	10	68	21 0:0:0	Доступний
scit3	idle	4	16	21 0:0:0	Доступний
scit3	down	1	4	21 0:0:0	Доступний
scit4_it	idle	2	24	1:0:0	Доступний
kepler	down*	1	2	1 0:0:0	Доступний
lite_task	down*	4	16	1:0:0	Доступний
lite_task	idle	6	24	1:0:0	Доступний

- У консольному режимі треба виконати команду `sinfo`:



```
pmchnu@access:~  
[pmchnu@access ~]$ sinfo  
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST  
scit3*      up 21-00:00:0      1  maint n3137  
scit3*      up 21-00:00:0      9  drain* n[3033,3035-3037,3119,3128,3133,3141,3152]  
scit3*      up 21-00:00:0     91  down* n[3003,3006-3007,3009,3012-3016,3018,3020-3025,3031-3032,3034,3038-3044,3047-3051,3053-3072,3075,3102-3114,3116-3118,3122-3127,3129-3132,3134-3136,3138-3140,3143-3146,3148,3150-3151]  
scit3*      up 21-00:00:0     10  alloc n[3011,3017,3019,3101,3115,3120-3121,3142,3147,3149]  
scit3*      up 21-00:00:0      4  idle  n[3002,3005,3008,3010]  
scit3*      up 21-00:00:0      1  down  n3004  
scit4_lt    up      1:00:00         2  idle  g[3211-3212]  
kepler      up 1-00:00:00      1  down* g3221  
lite_task   up      1:00:00         4  down* n[3047,3049-3051]  
lite_task   up      1:00:00         6  idle  n[3001,3026-3030]  
[pmchnu@access ~]$
```


- Щоби подивитися на список задач, які чекають своєї черги, треба зайти на закладку «Задачі»:

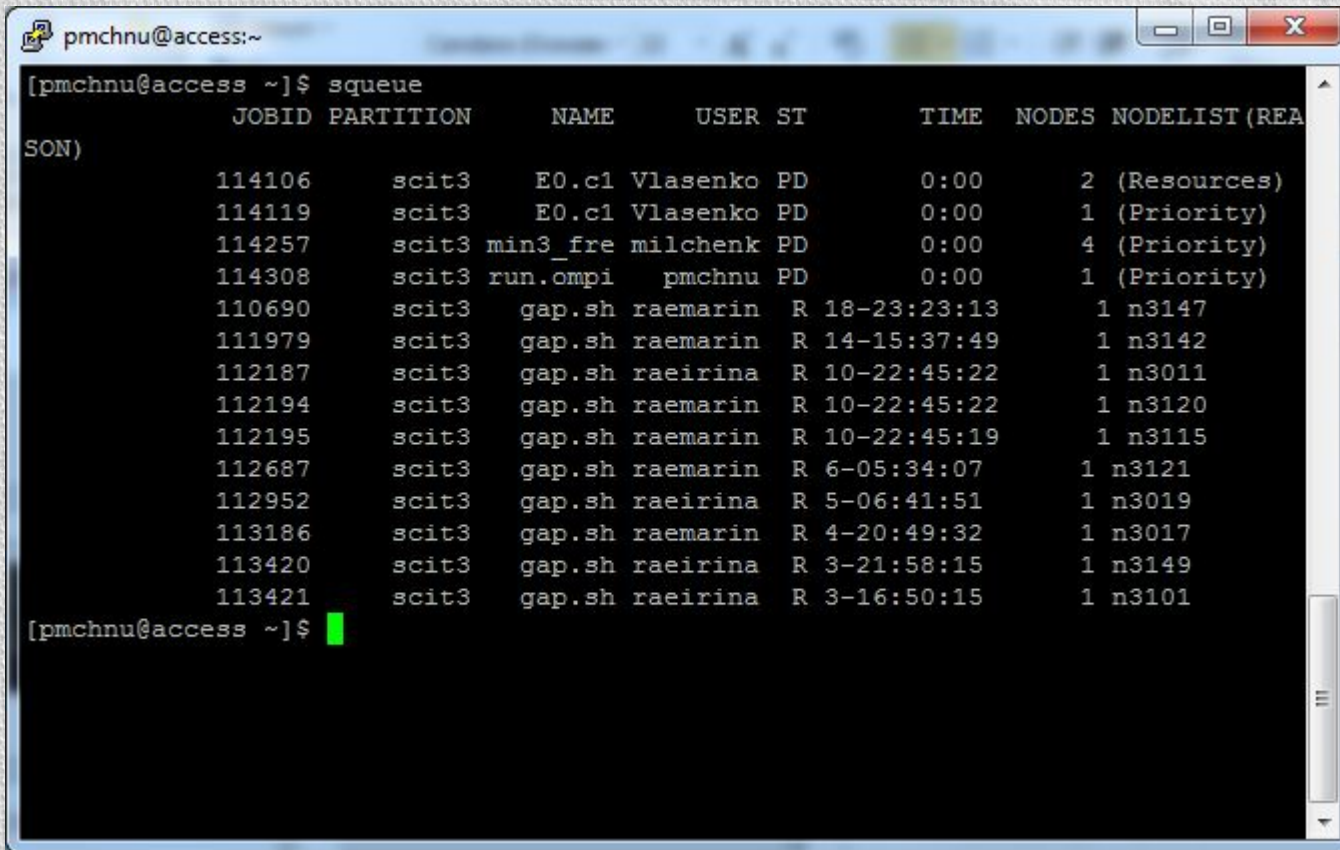
SCMS.pro Ім'я користувача: pmchnu Вихід

[Запуск задач](#)
[Ресурси](#)
[Задачі](#)
[Мапа ґрида](#)
[Повідомлення](#)
[Налаштування](#)

[Оновити](#)
[Історія запусків](#)

ID	Розділ	Ім'я задачі	Користувач	Стан	Час роботи	Ядер
Кластер: icybcluster.org.ua						
114106	scit3	E0.c1	Vlasenko	Очікує		48
114119	scit3	E0.c1	Vlasenko	Очікує		8
114257	scit3	min3_freq_cart_dp	milkhenko	Очікує		16
114308	scit3	run.OMPI	pmchnu	Очікує		16
110690	scit3	gap.sh	raemarina	Виконується	18 23:12:41	8
111979	scit3	gap.sh	raemarina	Виконується	14 15:27:17	8
112187	scit3	gap.sh	raeirina	Виконується	10 22:34:50	4
112194	scit3	gap.sh	raemarina	Виконується	10 22:34:50	8
112195	scit3	gap.sh	raemarina	Виконується	10 22:34:47	8
112687	scit3	gap.sh	raemarina	Виконується	6 5:23:35	8
112952	scit3	gap.sh	raeirina	Виконується	5 6:31:19	4
113186	scit3	gap.sh	raemarina	Виконується	4 20:39:0	4
113420	scit3	gap.sh	raeirina	Виконується	3 21:47:43	8
113421	scit3	gap.sh	raeirina	Виконується	3 16:39:43	8

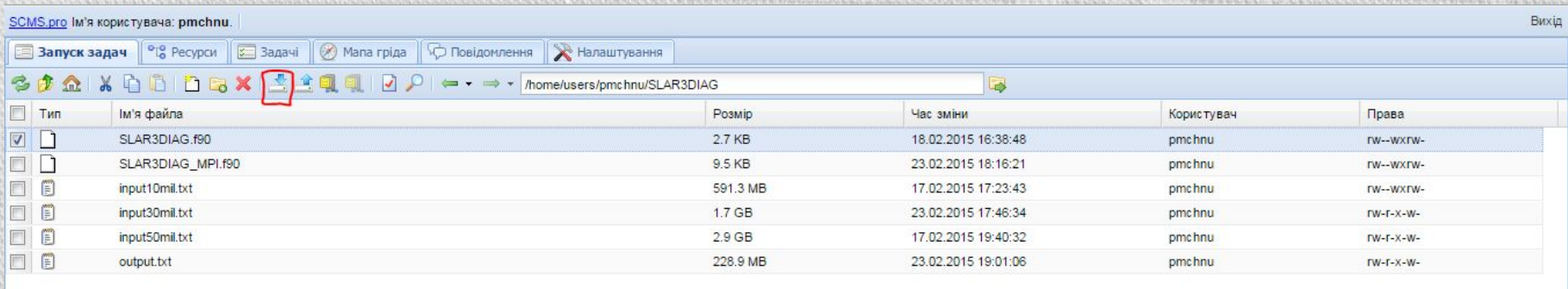
- У консольному режимі цьому відповідає команда `squeue`:



```
[pmchnu@access ~]$ squeue
JOBID PARTITION NAME USER ST TIME NODES NODELIST (REASON)
114106 scit3 E0.c1 Vlasenko PD 0:00 2 (Resources)
114119 scit3 E0.c1 Vlasenko PD 0:00 1 (Priority)
114257 scit3 min3_fre milchenk PD 0:00 4 (Priority)
114308 scit3 run.ampi pmchnu PD 0:00 1 (Priority)
110690 scit3 gap.sh raemarin R 18-23:23:13 1 n3147
111979 scit3 gap.sh raemarin R 14-15:37:49 1 n3142
112187 scit3 gap.sh raeirina R 10-22:45:22 1 n3011
112194 scit3 gap.sh raemarin R 10-22:45:22 1 n3120
112195 scit3 gap.sh raemarin R 10-22:45:19 1 n3115
112687 scit3 gap.sh raemarin R 6-05:34:07 1 n3121
112952 scit3 gap.sh raeirina R 5-06:41:51 1 n3019
113186 scit3 gap.sh raemarin R 4-20:49:32 1 n3017
113420 scit3 gap.sh raeirina R 3-21:58:15 1 n3149
113421 scit3 gap.sh raeirina R 3-16:50:15 1 n3101
[pmchnu@access ~]$
```


3.1. Обмін файлами

- Щоби завантажити деякий файл із сервера у графічному режимі, потрібно виділити необхідні файли і натиснути на кнопку «Скачати файл»:

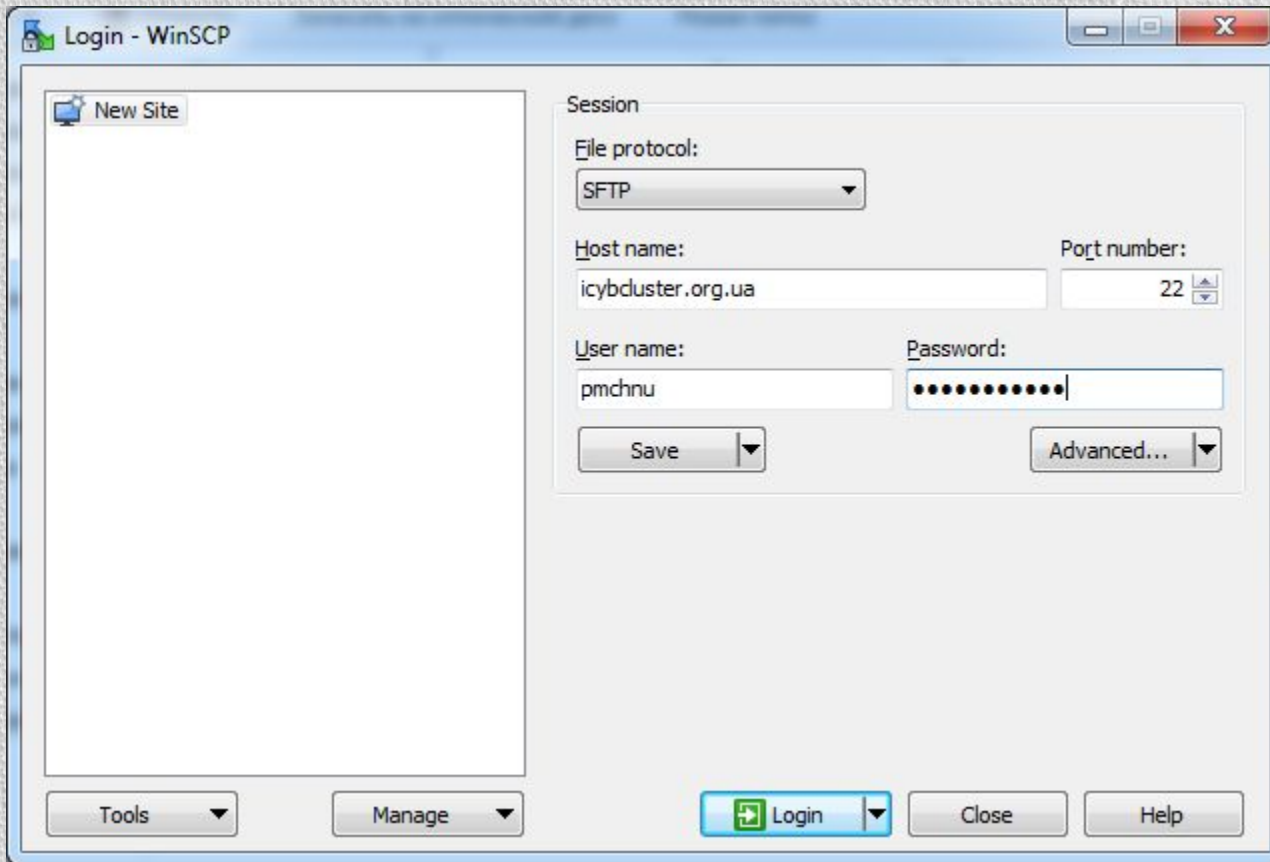


- У Linux за копіювання файлів відповідає утиліта `scp`. Так, щоби завантажити той самий файл засобами Linux, треба виконати команду:

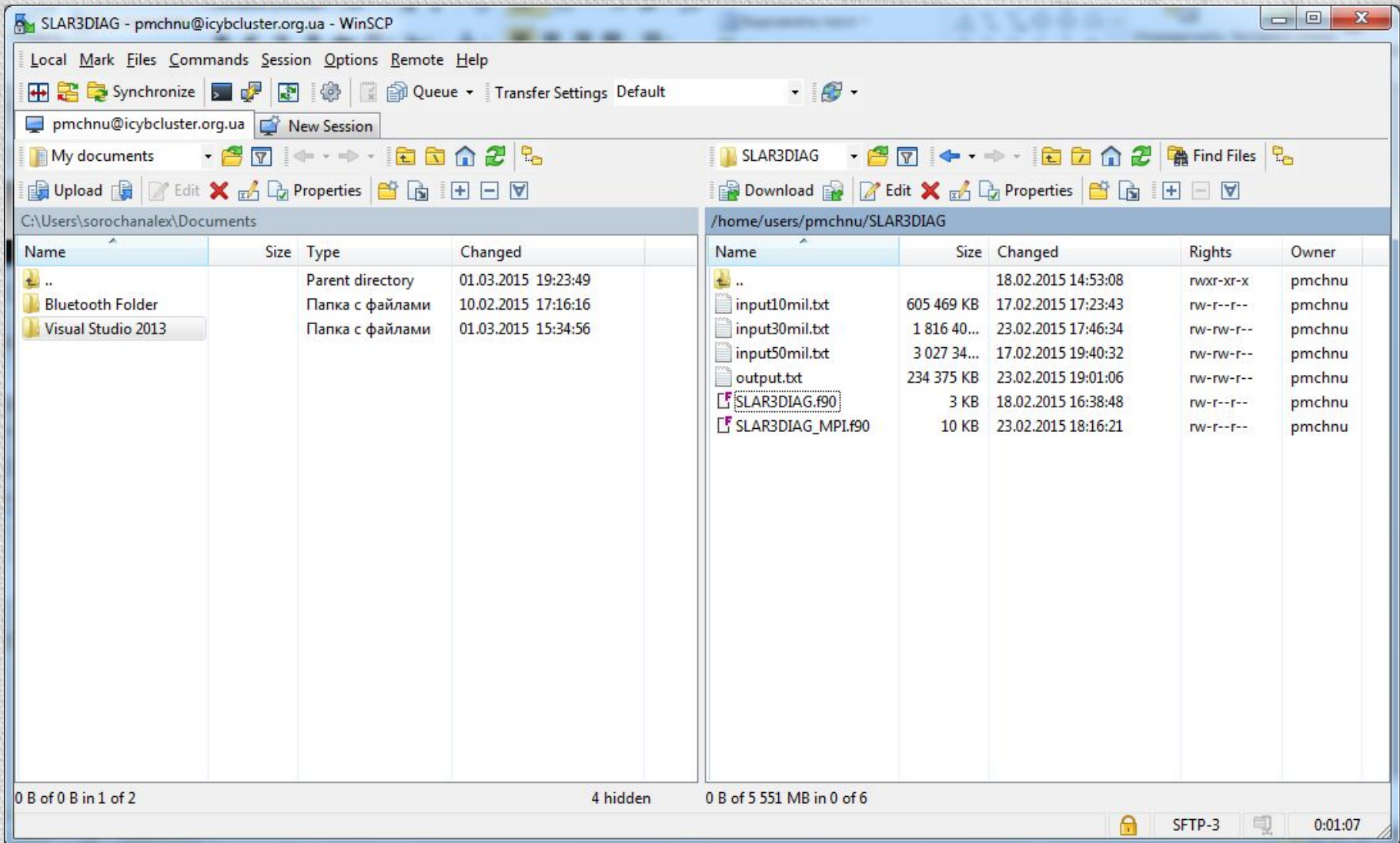
```
scp
```

```
pmchnu@icybcluster.org.ua:/home/users/pmchnu/SLAR3DIAG/SLAR3DIAG.f90 SLAR3DIAG.f90
```

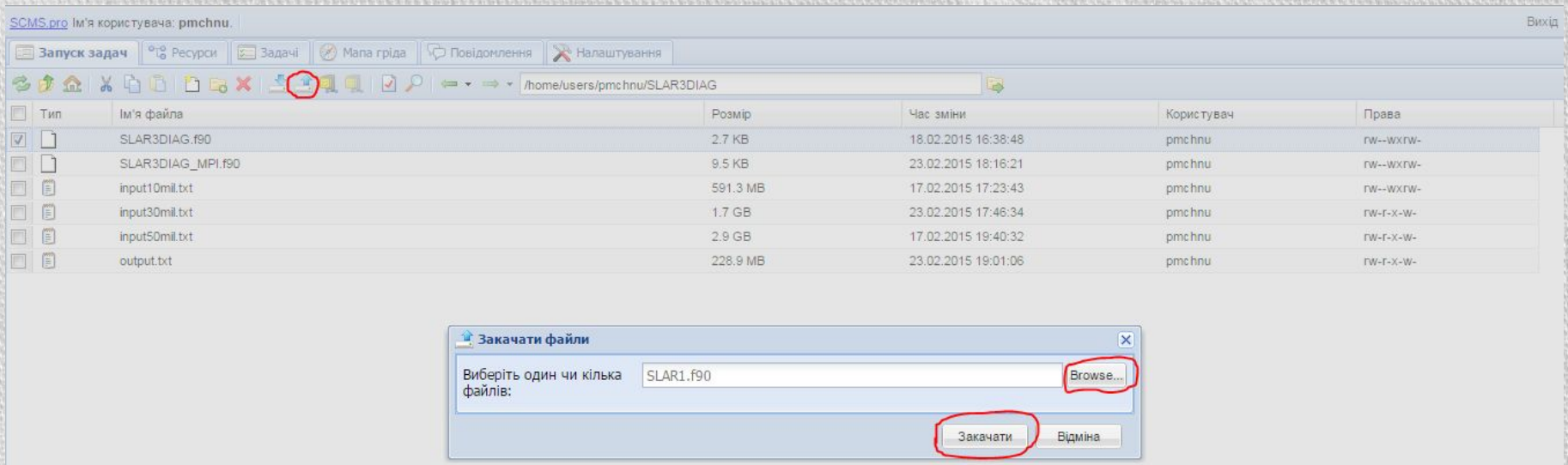

- У Windows необхідно встановити додатковий SCP-клієнт, наприклад, WinSCP:



- Відкриється наступне вікно:



- WinSCP відкриває одразу два дерева вибору файлів: на локальному комп'ютері та на суперкомп'ютері. За допомогою кнопок «Download» та «Upload» можна здійснювати обмін файлами.
- Якщо ж користуватися веб-браузером, то необхідно натиснути на кнопку «Закачування», та вибрати відповідні файли:



3.2. Запуск MPI-програм

- Щоби запускати свої завдання на суперкомп'ютері, необхідно зробити 3 речі:
 - Завантажити на сервер програмний код на мові C, C++ чи Fortran.
 - Провести компіляцію файлу у бінарний файл.
 - Запустити бінарний файл на виконання.
- Розпочнемо з запуску MPI-програм, як найбільш природного засобу для розпаралелювання на суперкомп'ютері Інституту кібернетики.
- На суперкомп'ютері встановлені 2 реалізації MPI: OpenMPI та MVAPICH2. OpenMPI вважається стандартною, а MVAPICH2 призначений для інтеграції з технологією CUDA, хоча її можна використовувати й замість OpenMPI.

- Спочатку розглянемо запуск програм у консольному режимі.
- На жаль, на керуючому вузлі, який одразу доступний для користувача, не встановлено жодних компіляторів. Компілятори встановлені безпосередньо на вузлах, на яких виконуються завдання. Тому необхідно з'єднатися із цими вузлами.
- Спочатку необхідно здійснити запит на ресурси до якої-небудь черги. Найоптимальнішим варіантом є черга `lite_task`, у якої малий ліміт часу і яка найшвидше надасть ресурси. Запит здійснюється наступною командою:

```
salloc -p lite_task bash
```

- Після виконання команди необхідно здійснити команду `squeue`, щоби дізнатися номер вузла, на якому були виділені ресурси.

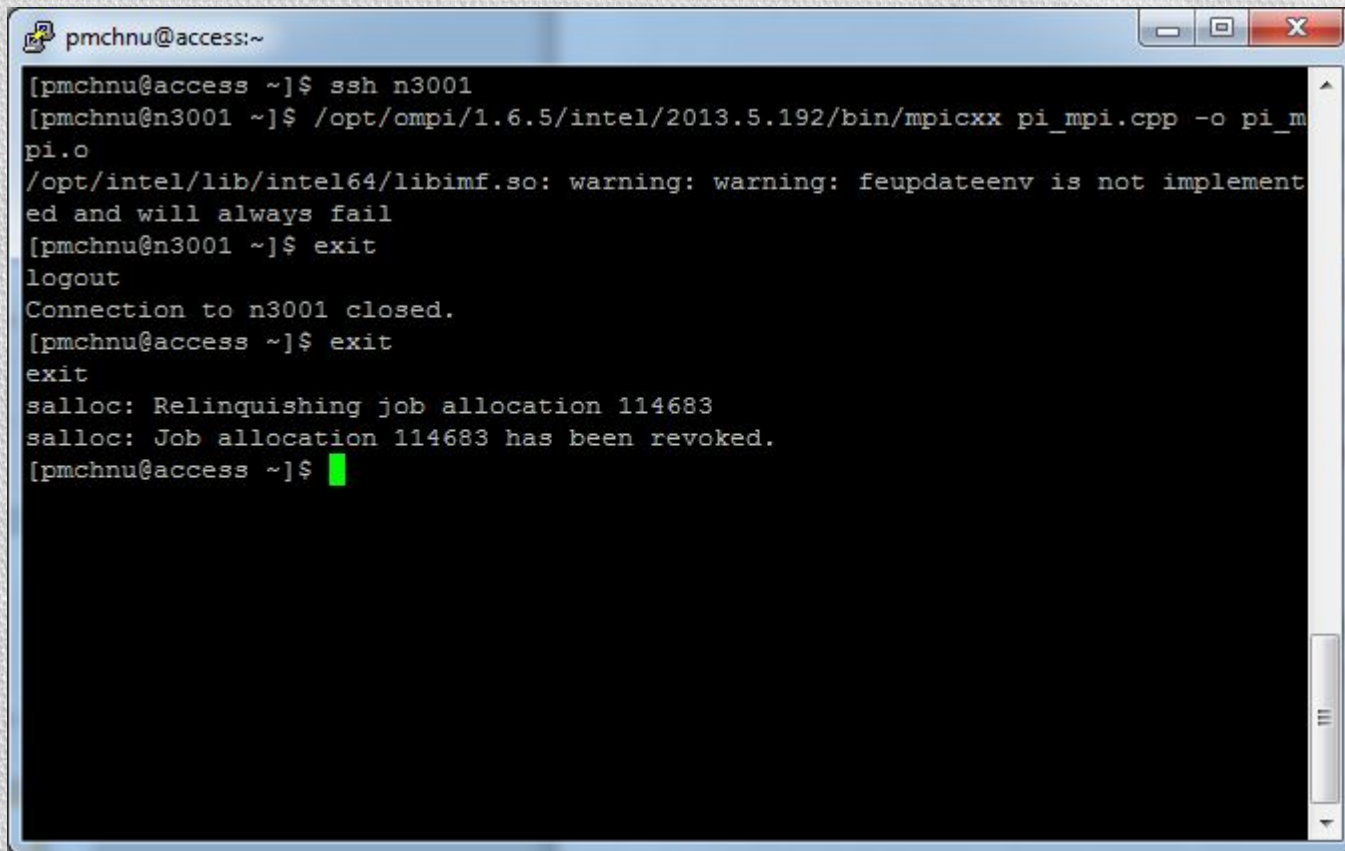

```
pmchnu@access:~  
[pmchnu@access ~]$ salloc -p lite_task bash  
salloc: Pending job allocation 114683  
salloc: job 114683 queued and waiting for resources  
salloc: job 114683 has been allocated resources  
salloc: Granted job allocation 114683  
[pmchnu@access ~]$ squeue  
JOBID PARTITION NAME USER ST TIME NODES NODELIST (REAS  
SON)  
114683 lite_task bash pmchnu R 0:05 1 n3001  
114308 scit3 run.ombi pmchnu PD 0:00 1 (Resources)  
114337 scit3 mpicxx Vlasenko PD 0:00 1 (Resources)  
114338 scit3 mpicxx Vlasenko PD 0:00 1 (Resources)  
114341 scit3 E0.c1 Vlasenko PD 0:00 1 (Resources)  
110690 scit3 gap.sh raemarin R 19-20:24:24 1 n3147  
111979 scit3 gap.sh raemarin R 15-12:39:00 1 n3142  
112187 scit3 gap.sh raeirina R 11-19:46:33 1 n3011  
112194 scit3 gap.sh raemarin R 11-19:46:33 1 n3120  
112195 scit3 gap.sh raemarin R 11-19:46:30 1 n3115  
112687 scit3 gap.sh raemarin R 7-02:35:18 1 n3121  
112952 scit3 gap.sh raeirina R 6-03:43:02 1 n3019  
113186 scit3 gap.sh raemarin R 5-17:50:43 1 n3017  
113420 scit3 gap.sh raeirina R 4-18:59:26 1 n3149  
113421 scit3 gap.sh raeirina R 4-13:51:26 1 n3101  
114257 scit3 min3_fre milchenk R 19:35:18 4 n[3002,3005,
```

- По виділенні ресурсів необхідно з'єднатися із цим вузлом за допомогою команди `ssh`. У наведеному вище випадку необхідно виконати команду:

```
ssh n3001
```


- Для того щоб підтягнути усі необхідні заголовні файли та файли бібліотек, адміністраторами суперкомп'ютера були розроблені спеціальні скрипти для компіляції MPI-програм. Для будь-яких комбінацій бібліотек MPI та компіляторів ці скрипти називаються:
 - mpicc для мови C
 - mpicxx для мови C++
 - mpif77 для мови FORTRAN 77
 - mpif90 для мови Fortran 90
- Для уникнення конфліктів між різними версіями, шляхи до папки зі скриптами формуються наступним чином:
/opt/<назва реалізації MPI>/<версія MPI>/<назва компілятора>/
<версія компілятора>/bin

- Скрипти для бібліотеки OpenMPI треба шукати у папці /opt/ompi
- Скрипти для бібліотеки MVAPICH2 треба шукати у папці /opt/mvapich2
- Замість <назва компілятора> необхідно писати gcc або intel – залежно, який саме компілятор буде використовуватися.
- Наводимо шляхи з останніми версіями бібліотек та компіляторів на момент оформлення презентації:
 - OpenMPI, GCC: /opt/ompi/1.6.5/gcc/4.4/bin
 - OpenMPI, Intel: /opt/ompi/1.6.5/intel/2013.5.192/bin
 - MVAPICH2, GCC: /opt/mvapich2/1.9a/gcc/4.4/bin
 - MVAPICH2, Intel: /opt/mvapich2/1.9a/intel/current/bin
- Компіляція відбувається наступним чином:
<шлях до скрипта> <файл з програмним кодом> -o <бінарний файл>

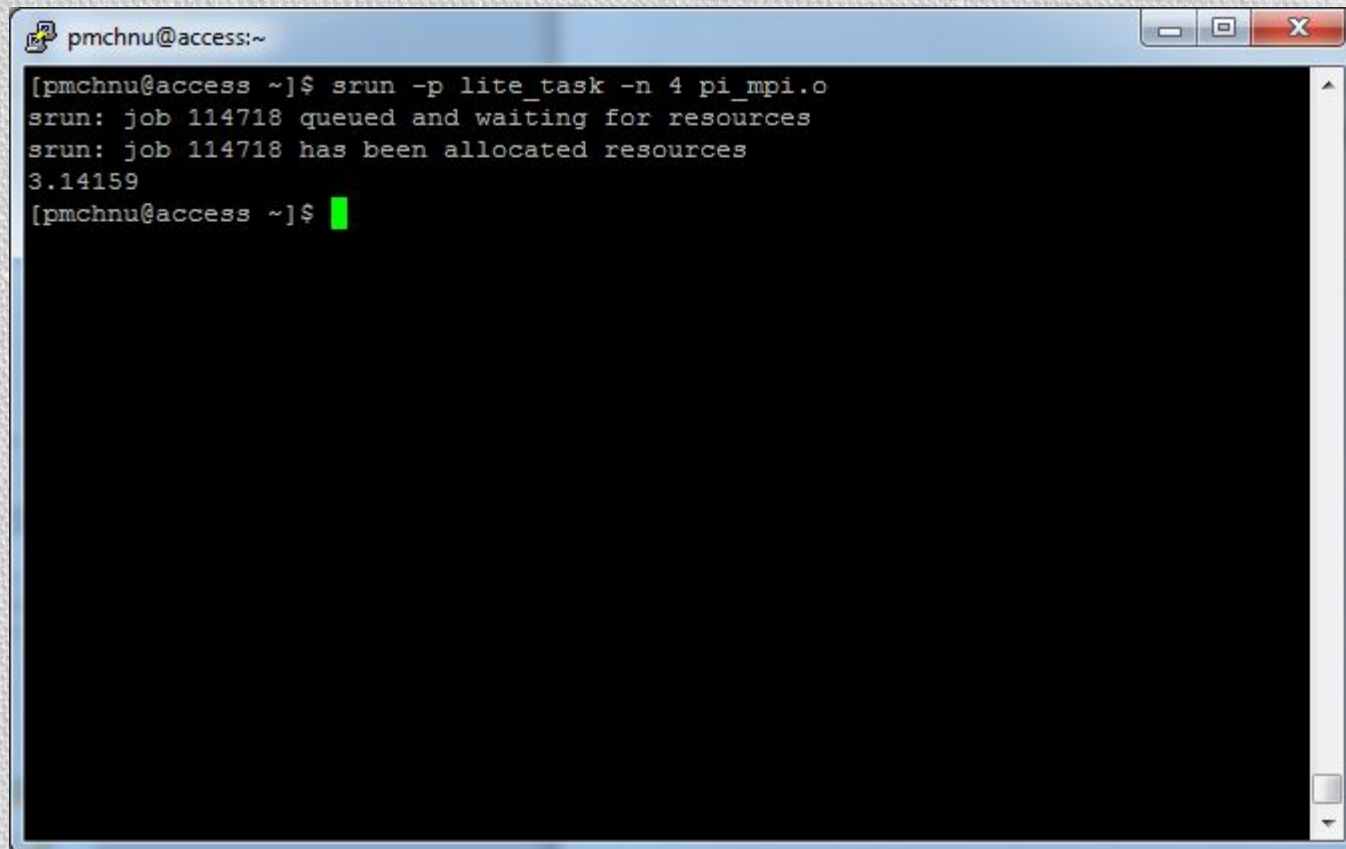


```
pmchnu@access:~  
[pmchnu@access ~]$ ssh n3001  
[pmchnu@n3001 ~]$ /opt/ompi/1.6.5/intel/2013.5.192/bin/mpicxx pi_mpi.cpp -o pi_m  
pi.o  
/opt/intel/lib/intel64/libimf.so: warning: warning: feupdateenv is not implement  
ed and will always fail  
[pmchnu@n3001 ~]$ exit  
logout  
Connection to n3001 closed.  
[pmchnu@access ~]$ exit  
exit  
salloc: Relinquishing job allocation 114683  
salloc: Job allocation 114683 has been revoked.  
[pmchnu@access ~]$ █
```

- Як бачимо, у нашому прикладі компілятор видав попередження, яке втім ніяк не позначається на подальшій роботі програми.
- Після компіляції необхідно двічі виконати команду exit, щоби звільнити ресурси.

- Скомпільований бінарний файл можна ставити у чергу виконання. Для цього можна виконати команду `sgun` з наступним синтаксисом:
`sgun -p <черга> -n <кількість ядер> <шлях до скомпільованої програми>`
- Ця команда запускає програму у діалоговому режимі. Водночас, якщо необхідних ресурсів на суперкомп'ютері немає, то очікування виконання може зайняти досить довго часу.
- Для того щоб зупинити виконання програми, необхідно натиснути комбінацію клавіш `Ctrl + C`.
- Також необхідно зазначити, що при використанні OpenMPI команда `sgun` поводитиме себе некоректно.

- Приклад запуску програми обчислення числа π у черзі lite_task з чотирма процесорами. Використано бібліотеку MVAPICH2 з компілятором Intel.



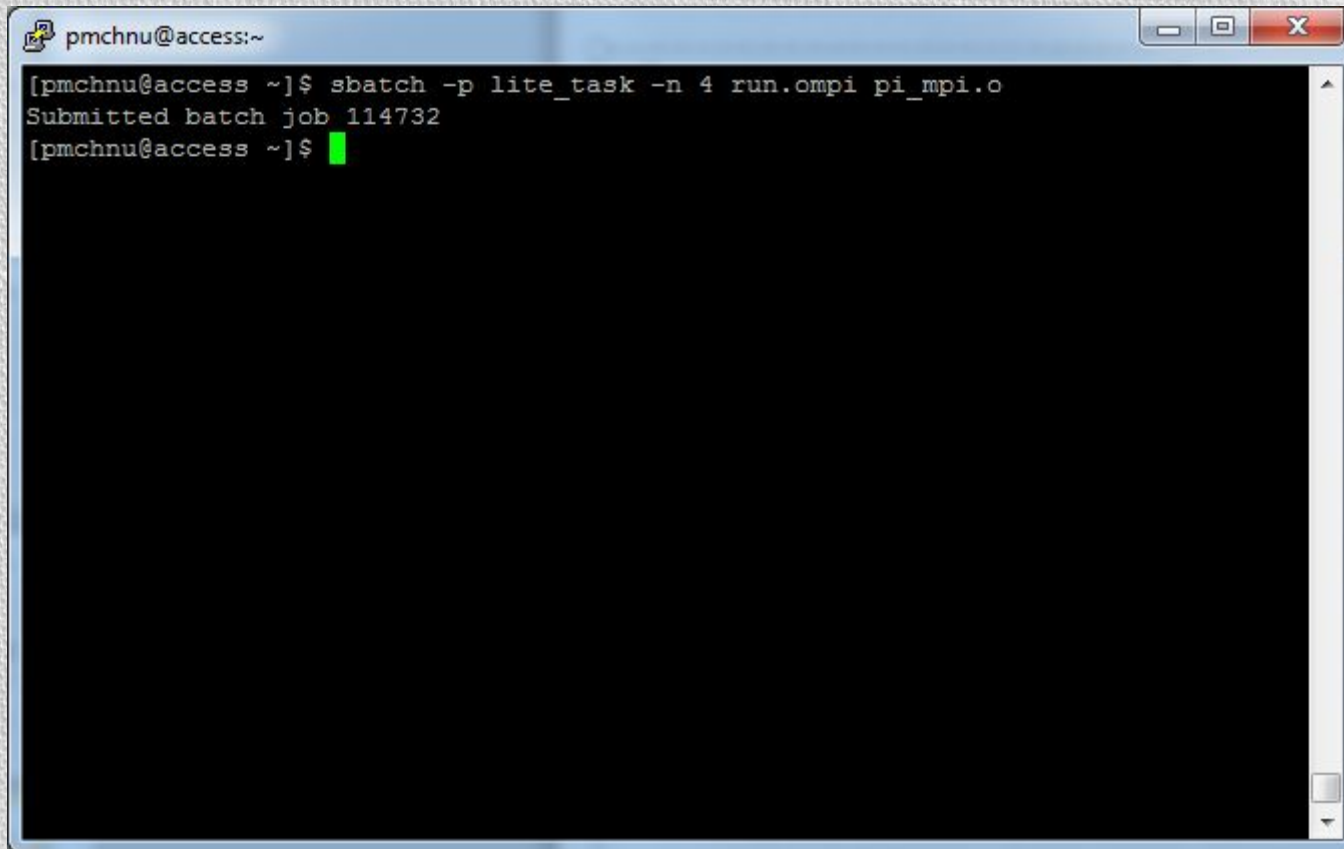
```
pmchnu@access:~  
[pmchnu@access ~]$ srun -p lite_task -n 4 pi_mpi.o  
srun: job 114718 queued and waiting for resources  
srun: job 114718 has been allocated resources  
3.14159  
[pmchnu@access ~]$
```


- Більш розповсюдженим варіантом запуску задач є використання команди `sbatch`, яка виконує задачу у фоновому режимі. Синтаксис команди:

```
sbatch -p <черга> -n <кількість ядер> <скрипт запуску>
```

- Так як задача виконується у фоновому режимі, то програма не має очікувати якихось додаткових дій від користувача, наприклад, введення значення змінних.
- На відміну від `srun`, `sbatch` потребує в якості одного з параметрів не бінарний файл програми, а `bash`-скрипт, всередині якого має бути звернення, наприклад, до команди `srun`.
- Адміністраторами суперкомп'ютера уже розроблені додаткові скрипти для запуску програм. Щоби запустити OpenMPI-програму необхідно запустити скрипт `run.ompi`, в якості параметрів якого передається бінарний файл програми з її параметрами. Для MVAPICH2 розроблено скрипт `run.mvarich2`, який, щоправда, нічого не виконує, окрім прямого запуску програми на суперкомп'ютері.

- Приклад запуску програми обчислення числа π у черзі `lite_task` з чотирма процесорами. Використано бібліотеку OpenMPI з компілятором Intel.



```
pmchnu@access:~  
[pmchnu@access ~]$ sbatch -p lite_task -n 4 run.omp pi_mpi.o  
Submitted batch job 114732  
[pmchnu@access ~]$
```


- За станом виконання задачі можна стежити, виконавши команду `squeue`:

```
pmchnu@access:~  
[pmchnu@access ~]$ sbatch -p lite_task -n 4 run.ombi pi_mpi.o  
Submitted batch job 114734  
[pmchnu@access ~]$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
<u>114734</u>	lite_task	run.ombi	pmchnu	PD	0:00	1	(Priority)
114308	scit3	run.ombi	pmchnu	PD	0:00	1	(Resources)
114337	scit3	mpicxx	Vlasenko	PD	0:00	1	(Resources)
114338	scit3	mpicxx	Vlasenko	PD	0:00	1	(Resources)
114341	scit3	E0.c1	Vlasenko	PD	0:00	1	(Resources)
110690	scit3	gap.sh	raemarin	R	19-22:22:25	1	n3147
111979	scit3	gap.sh	raemarin	R	15-14:37:01	1	n3142
112187	scit3	gap.sh	raeirina	R	11-21:44:34	1	n3011
112194	scit3	gap.sh	raemarin	R	11-21:44:34	1	n3120
112195	scit3	gap.sh	raemarin	R	11-21:44:31	1	n3115
112687	scit3	gap.sh	raemarin	R	7-04:33:19	1	n3121
112952	scit3	gap.sh	raeirina	R	6-05:41:03	1	n3019
113186	scit3	gap.sh	raemarin	R	5-19:48:44	1	n3017
113420	scit3	gap.sh	raeirina	R	4-20:57:27	1	n3149
113421	scit3	gap.sh	raeirina	R	4-15:49:27	1	n3101
114257	scit3	min3_fre	milchenk	R	21:33:19	4	n[3002,3005,3008,3010]

```
[pmchnu@access ~]$
```

- Якщо треба відмінити завдання, то виконується команда `scancel`, наприклад:
`scancel 114734`

- Також слідкувати за цим можна на сайті, попередньо натиснувши на кнопку «Оновити» у третьому зверху рядку:

SCMS.pro Ім'я користувача: pmchnu. Вихід

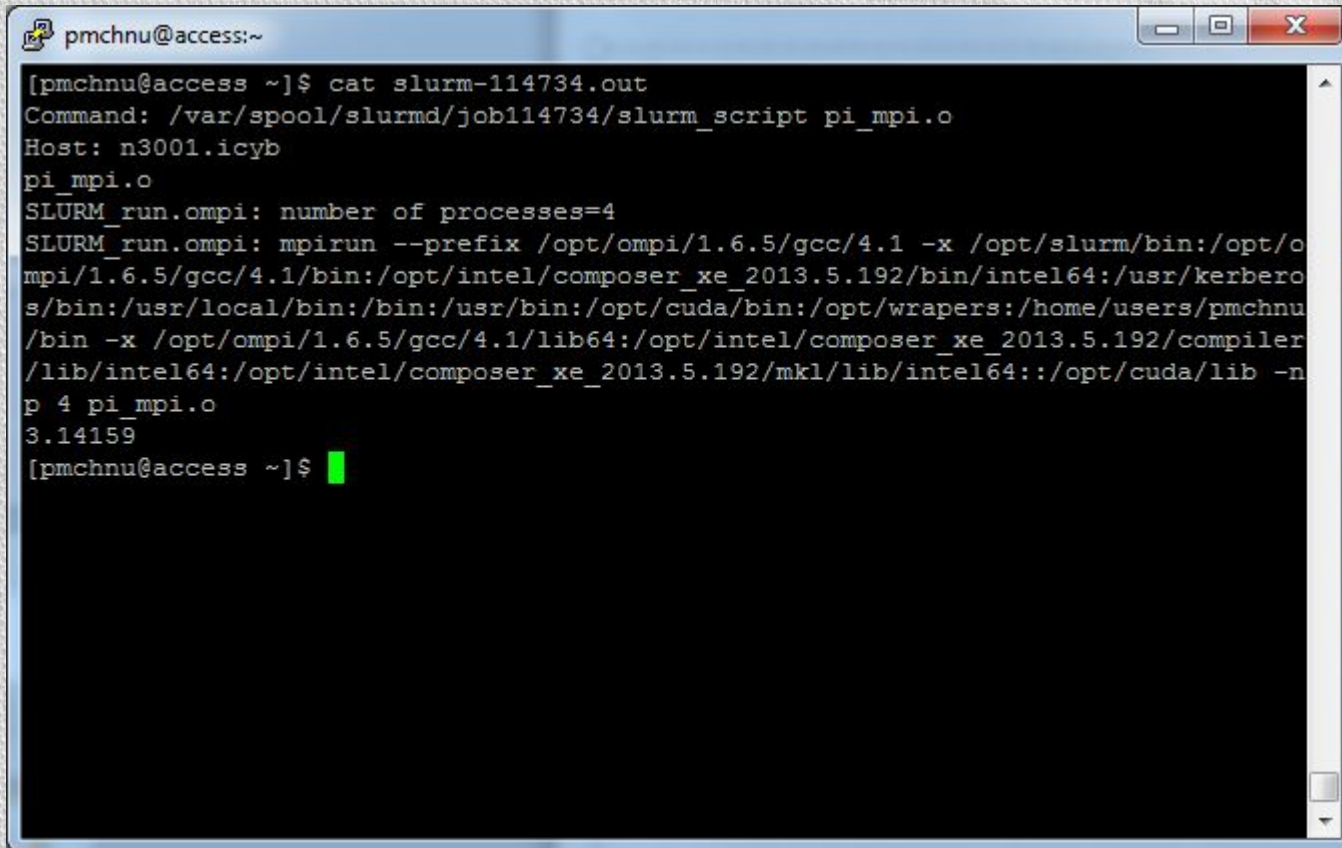
[Запуск задач](#)
[Ресурси](#)
[Задачі](#)
[Мапа ґріда](#)
[Повідомлення](#)
[Налаштування](#)

[Оновити](#)
[Історія запусків](#)

ID	Поздін	Ім'я задачі	Користувач	Стан	Час роботи	Ядер
Кластер: icybcluster.org.ua						
<u>114734</u>	lite_task	run.mpi	pmchnu	Очікує		4
114308	scit3	run.mpi	pmchnu	Очікує		16
114337	scit3	mpicxx	Vlasenko	Очікує		4
114338	scit3	mpicxx	Vlasenko	Очікує		4
114341	scit3	E0.c1	Vlasenko	Очікує		8
110690	scit3	gap.sh	raemarina	Виконується	19 22:22:48	8
111979	scit3	gap.sh	raemarina	Виконується	15 14:37:24	8
112187	scit3	gap.sh	raeirina	Виконується	11 21:44:57	4
112194	scit3	gap.sh	raemarina	Виконується	11 21:44:57	8
112195	scit3	gap.sh	raemarina	Виконується	11 21:44:54	8
112687	scit3	gap.sh	raemarina	Виконується	7 4:33:42	8
112952	scit3	gap.sh	raeirina	Виконується	6 5:41:26	4
113186	scit3	gap.sh	raemarina	Виконується	5 19:49:7	4
113420	scit3	gap.sh	raeirina	Виконується	4 20:57:50	8
113421	scit3	gap.sh	raeirina	Виконується	4 15:49:50	8
114257	scit3	min3_freq_cart_dp	milchenko	Виконується	21:33:42	16

- Після того, як задача виконається (а сама вона зникне зі списку запущених задач), у папці звідки була запущена команда sbatch, з'явиться файл з назвою slurm-<номер задачі>.out, у якому буде записано все, що мало записуватися у стандартний потік виводу.

- Дослідити вміст цього файлу можна за допомогою команди cat:



```
pmchnu@access:~  
[pmchnu@access ~]$ cat slurm-114734.out  
Command: /var/spool/slurmd/job114734/slurm_script pi_mpi.o  
Host: n3001.icyb  
pi_mpi.o  
SLURM_run.ampi: number of processes=4  
SLURM_run.ampi: mpirun --prefix /opt/mpi/1.6.5/gcc/4.1 -x /opt/slurm/bin:/opt/om  
mpi/1.6.5/gcc/4.1/bin:/opt/intel/composer_xe_2013.5.192/bin/intel64:/usr/kerbero  
s/bin:/usr/local/bin:/bin:/usr/bin:/opt/cuda/bin:/opt/wrapers:/home/users/pmchnu  
/bin -x /opt/mpi/1.6.5/gcc/4.1/lib64:/opt/intel/composer_xe_2013.5.192/compiler  
/lib/intel64:/opt/intel/composer_xe_2013.5.192/mkl/lib/intel64::/opt/cuda/lib -n  
p 4 pi_mpi.o  
3.14159  
[pmchnu@access ~]$
```


- З сайту також можна подивитись на вміст цього файлу, попередньо натиснувши кнопку «Оновити»

The screenshot displays the SCMS_pro web interface for user pmchnu. The main content area shows a file viewer for 'slurm-114734.out'. The file viewer window contains the following text:

```
Command: /var/spool/slurmd/job114734/slurm_script pi_mpi.o
Host: n3001.icyb
pi_mpi.o
SLURM_run.OMPI: number of processes=4
SLURM_run.OMPI: mpirun --prefix /opt/ompi/1.6.5/gcc/4.1 -x
/opt/slurm/bin:/opt/ompi/1.6.5/gcc/4.1/bin:/opt/intel/composer_xe_2013.5.192/bin/intel64:/usr/kerberos/bin:/usr
/local/bin:/bin:/usr/bin:/opt/cuda/bin:/opt/wrappers:/home/users/pmchnu/bin -x
/opt/ompi/1.6.5/gcc/4.1/lib64:/opt/intel/composer_xe_2013.5.192/compiler/lib/intel64:/opt/intel/composer_xe_201
3.5.192/mkl/lib/intel64:/opt/cuda/lib -np 4 pi_mpi.o
3.14159
```

The interface also includes a 'Форма запуску' (Launch Form) section with the following fields:

- Кластер та черга: [empty]
- Робоча тека: /home/users/pmchnu
- Скрипт запуску: MPI
- Файл для виконання: [empty]
- Параметри задачі: [empty]

On the right side, there is a 'Права' (Permissions) table and a 'Профілі' (Profiles) section.

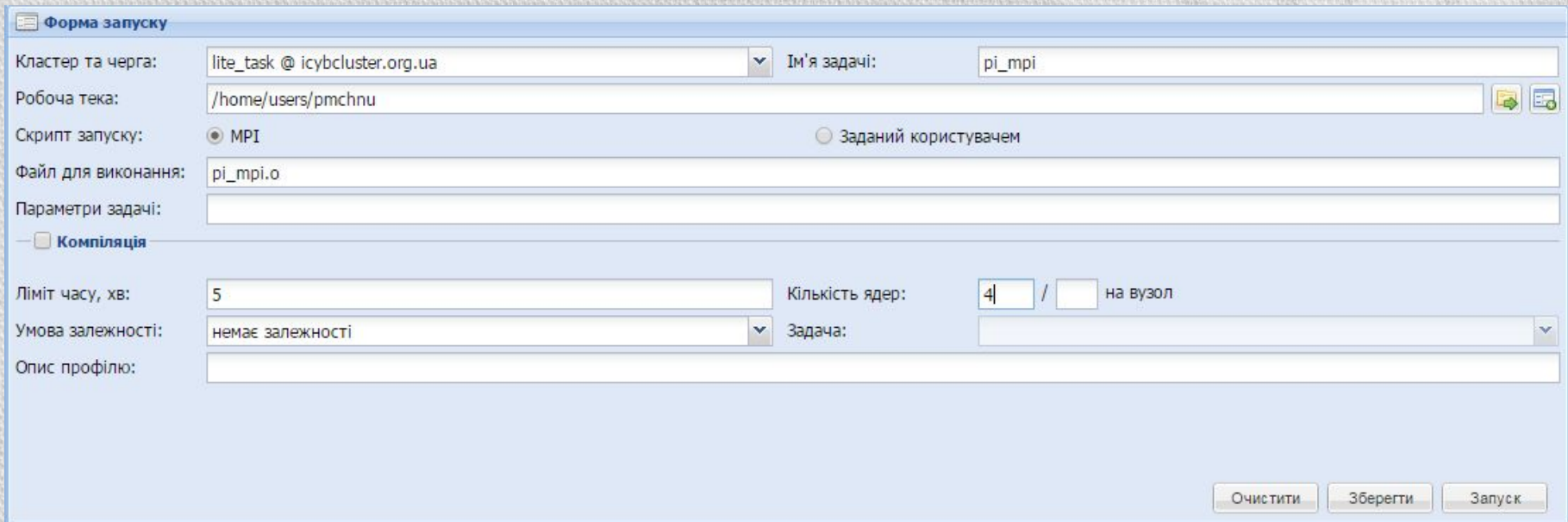
Права
rw-r-x-w-
rwXrw--x
rwXr--r-x
rwXr--r-x
rw--wxrw-
rwXr--r-x
rw-r-x-w-

The 'Профілі' section lists the following profiles:

- gameSS-2009-scit3
- grid-hostname
- grid-sleep
- hostname
- sleep

- Запускати задачу можна і з веб-інтерфейсу. Форма запуску на закладці «Запуск задач» є оболонкою для команди `sbatch`. Форма дозволяє автоматично застосувати скрипт `run.ompi` або задати свій скрипт. Серед недоліків форми є обов'язкове заповнення майже усіх полів форми.
- Компіляція файлів формою передбачено, але потребує наявності файлу `Makefile` для Linux-івської утиліти `make` та виконується одночасно з запуском задачі, що далеко не завжди є прийнятним через неузгодженість між параметрами `SLURM` для цих двох задач.

- Запуск з форми відбувається таким чином:



The screenshot shows a web-based form for submitting a job to a cluster. The form is titled "Форма запуску" and contains the following fields and controls:

- Кластер та черга:** A dropdown menu with the value "lite_task @ icybcluster.org.ua".
- Ім'я задачі:** A text input field containing "pi_mpi".
- Робоча тека:** A text input field containing "/home/users/pmchnu".
- Скрипт запуску:** Two radio buttons: "MPI" (selected) and "Заданий користувачем".
- Файл для виконання:** A text input field containing "pi_mpi.o".
- Параметри задачі:** An empty text input field.
- Компіляція:** A collapsed section with a minus sign and the label "Компіляція".
- Ліміт часу, хв:** A text input field containing "5".
- Кількість ядер:** A text input field containing "4" followed by a slash and another empty text input field, with the label "на вузол".
- Умова залежності:** A dropdown menu with the value "немає залежності".
- Задача:** A dropdown menu.
- Опис профілю:** An empty text input field.

At the bottom right of the form, there are three buttons: "Очистити", "Зберегти", and "Запуск".

- При цьому у каталозі, звідки було запущено задачу, створюється папка <ім'я задачі>.out, де буде розташовано 2 файли: файл з розширенням .err для відображення помилок і .out з вмістом стандартного потоку виводу.

3.3. Запуск OpenMP-програм

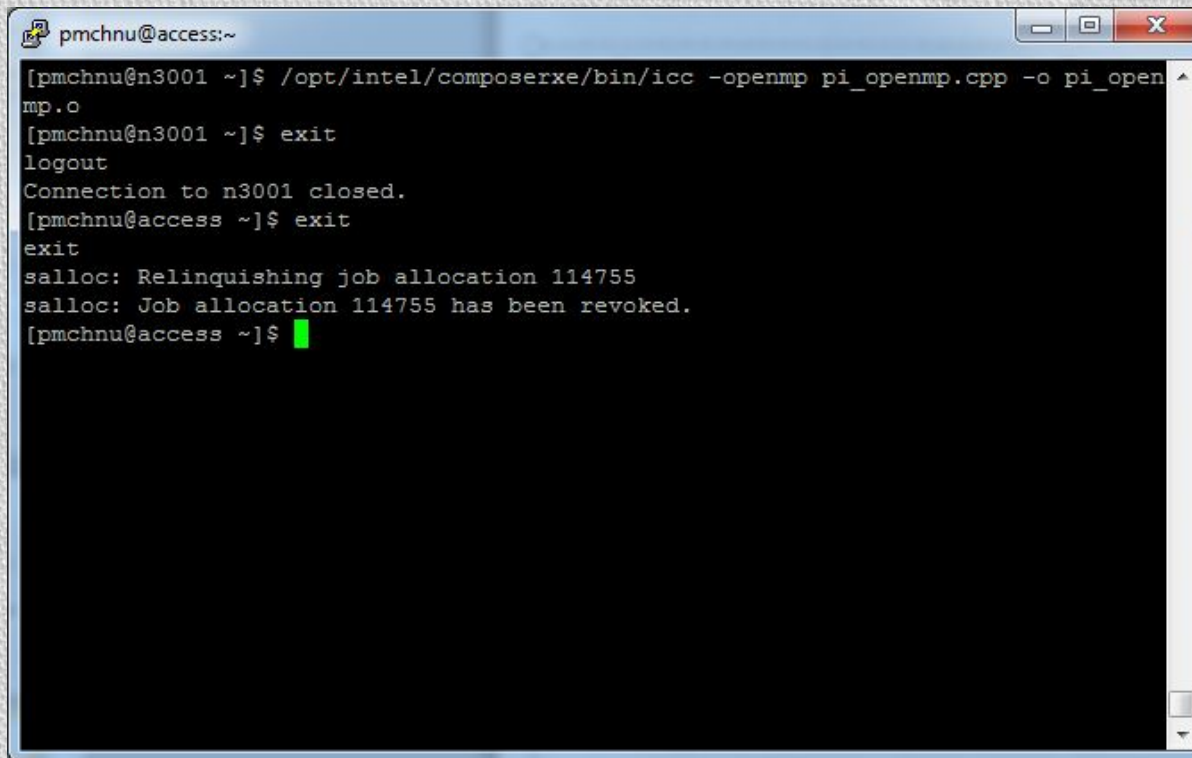
- Щоби скомпілювати OpenMP-програму, як і у випадку з MPI, необхідно з'єднатися з додатковим вузлом.
- Компіляція здійснюється наступним чином:

<шлях до компілятора> -fopenmp <файл з програмним кодом> -o <бінарний файл>

для компіляторів GCC з заміною fopenmp на openmp у компіляторах Intel.

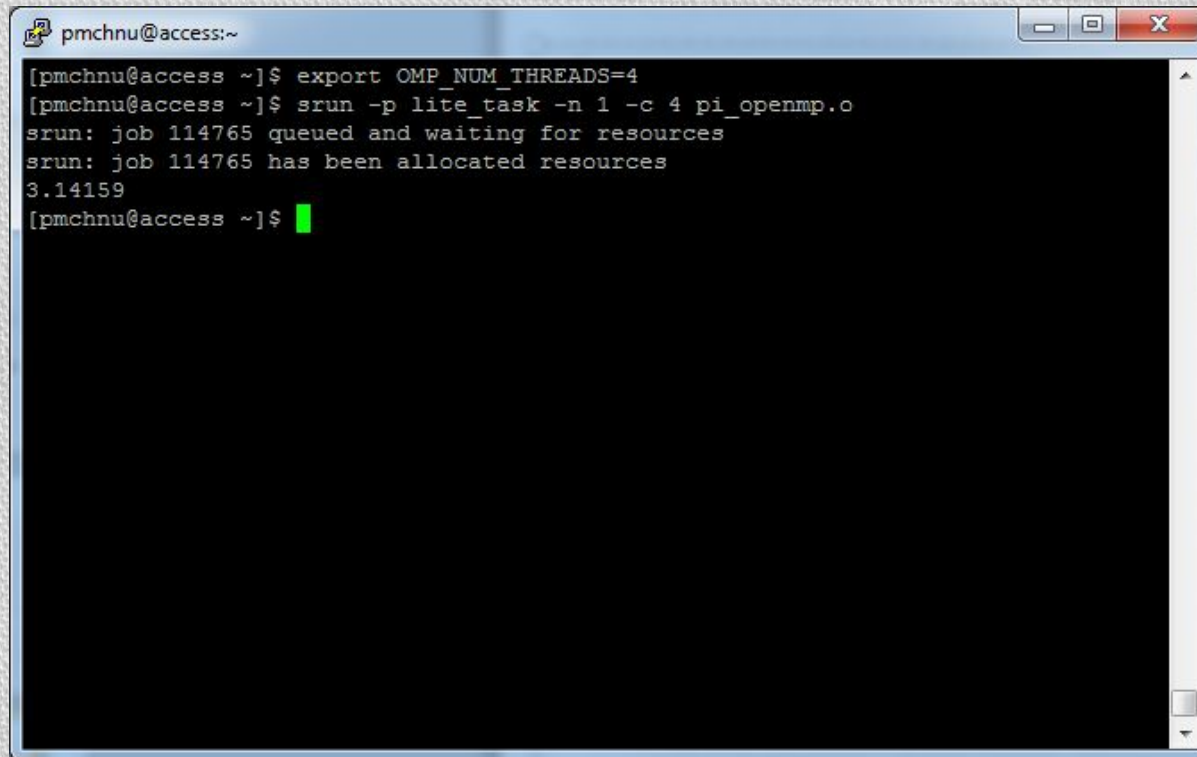
- Компілятори GCC можна викликати без вказання їхнього повного шляху, їхні найновіші версії мають такі назви:
 - gcc44 для мови C
 - g++44 для мови C++
 - g77 для мови FORTRAN 77
 - gfortran44 для мови Fortran 90

- Найновіші компілятори Intel зберігаються у папці `/opt/intel/composerxe/bin` і мають такі назви:
 - `icc` для мов C++ та C
 - `ifort` для мови Fortran
- Приклад компіляції:

A terminal window titled 'pmchnu@access:~' with standard window controls. The terminal shows a sequence of commands and system messages: a compilation command using 'icc', an 'exit' command leading to 'logout' and 'Connection to n3001 closed.', another 'exit' command leading to 'exit', and 'salloc' messages indicating job allocation revocation. The prompt returns to 'pmchnu@access ~]' with a green cursor.

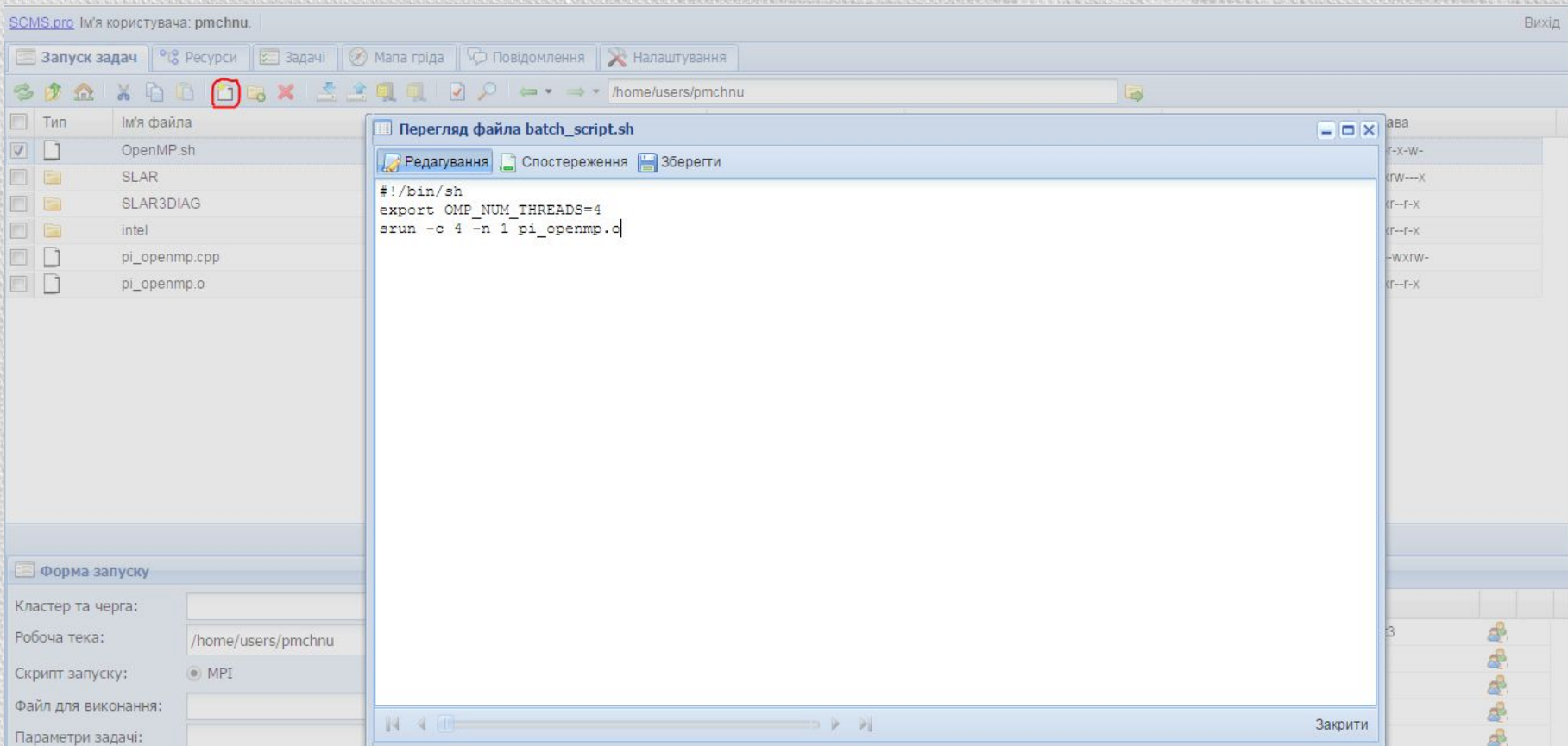
```
pmchnu@access:~  
[pmchnu@n3001 ~]$ /opt/intel/composerxe/bin/icc -openmp pi_openmp.cpp -o pi_openmp.o  
[pmchnu@n3001 ~]$ exit  
logout  
Connection to n3001 closed.  
[pmchnu@access ~]$ exit  
exit  
salloc: Relinquishing job allocation 114755  
salloc: Job allocation 114755 has been revoked.  
[pmchnu@access ~]$
```


- Виконання за допомогою команди `srun` виконується в 2 етапи:
 - Спочатку встановлюється значення змінної середовища:
`export OMP_NUM_THREADS=<кількість ядер>`
 - Потім виконується команда наступного вигляду:
`srun -p <черга> -n 1 -c <кількість ядер> <шлях до скомпільованої програми>`



```
pmchnu@access:~  
[pmchnu@access ~]$ export OMP_NUM_THREADS=4  
[pmchnu@access ~]$ srun -p lite_task -n 1 -c 4 pi_openmp.o  
srun: job 114765 queued and waiting for resources  
srun: job 114765 has been allocated resources  
3.14159  
[pmchnu@access ~]$ █
```

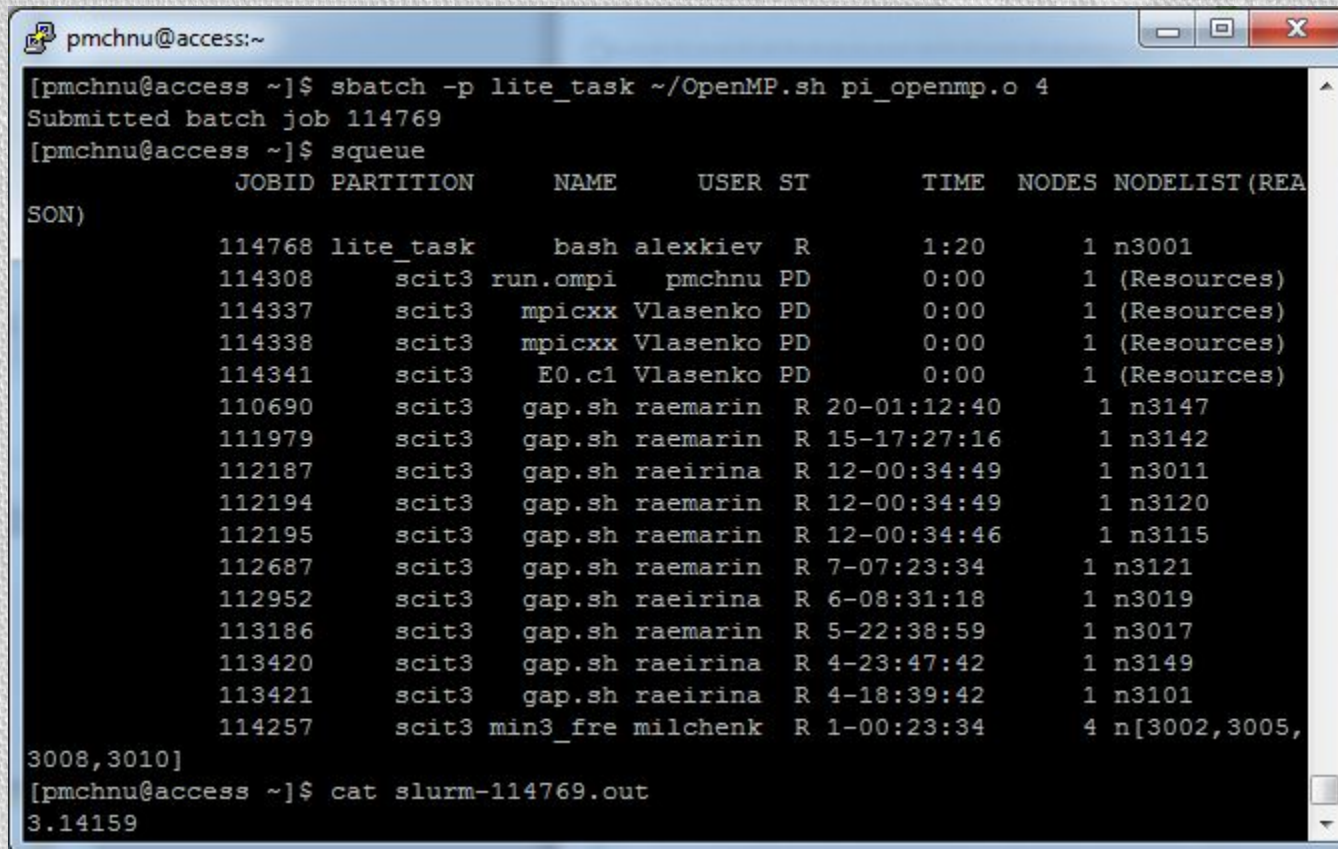

- Виконання за допомогою команди sbatch потребує скрипта, який треба написати вручну. Найпростіше це зробити з веб-інтерфейсу. Треба натиснути на кнопку «Створити файл», ввести ім'я файлу, записати ті ж команди, що потребувала команда srun, і зберегти:




```
pmchnu@access:~  
[pmchnu@access ~]$ sbatch -p lite_task batch_script.sh  
Submitted batch job 114767  
[pmchnu@access ~]$ squeue  
JOBID PARTITION NAME USER ST TIME NODES NODELIST (RESO  
SON)  
114308 scit3 run.omp pmchnu PD 0:00 1 (Resources)  
114337 scit3 mpicxx Vlasenko PD 0:00 1 (Resources)  
114338 scit3 mpicxx Vlasenko PD 0:00 1 (Resources)  
114341 scit3 E0.c1 Vlasenko PD 0:00 1 (Resources)  
110690 scit3 gap.sh raemarin R 20-01:03:49 1 n3147  
111979 scit3 gap.sh raemarin R 15-17:18:25 1 n3142  
112187 scit3 gap.sh raeirina R 12-00:25:58 1 n3011  
112194 scit3 gap.sh raemarin R 12-00:25:58 1 n3120  
112195 scit3 gap.sh raemarin R 12-00:25:55 1 n3115  
112687 scit3 gap.sh raemarin R 7-07:14:43 1 n3121  
112952 scit3 gap.sh raeirina R 6-08:22:27 1 n3019  
113186 scit3 gap.sh raemarin R 5-22:30:08 1 n3017  
113420 scit3 gap.sh raeirina R 4-23:38:51 1 n3149  
113421 scit3 gap.sh raeirina R 4-18:30:51 1 n3101  
114257 scit3 min3_fre milchenk R 1-00:14:43 4 n[3002,3005,  
3008,3010]  
[pmchnu@access ~]$ cat slurm-114767.out  
3.14159  
[pmchnu@access ~]$
```

- Параметри `s` і `n` визначені у скрипті, тому залишилося вказати лише чергу завдань.

- Для полегшення запуску OpenMP-програм можна написати нижченаведений скрипт. Наразі він зберігається у домашньому каталозі. Тоді виклик здійснюватиметься наступним чином:
`sbatch -p <черга> ~/OpenMP.sh <шлях до скомпільованої програми> <кількість ядер>`



```
pmchnu@access:~  
[pmchnu@access ~]$ sbatch -p lite_task ~/OpenMP.sh pi_openmp.o 4  
Submitted batch job 114769  
[pmchnu@access ~]$ squeue  
JOBID PARTITION NAME USER ST TIME NODES NODELIST (RESO  
N)
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (RESO
114768	lite_task	bash	alexkiev	R	1:20	1	n3001
114308	scit3	run.ombi	pmchnu	PD	0:00	1	(Resources)
114337	scit3	mpicxx	Vlasenko	PD	0:00	1	(Resources)
114338	scit3	mpicxx	Vlasenko	PD	0:00	1	(Resources)
114341	scit3	E0.c1	Vlasenko	PD	0:00	1	(Resources)
110690	scit3	gap.sh	raemarin	R	20-01:12:40	1	n3147
111979	scit3	gap.sh	raemarin	R	15-17:27:16	1	n3142
112187	scit3	gap.sh	raeirina	R	12-00:34:49	1	n3011
112194	scit3	gap.sh	raemarin	R	12-00:34:49	1	n3120
112195	scit3	gap.sh	raemarin	R	12-00:34:46	1	n3115
112687	scit3	gap.sh	raemarin	R	7-07:23:34	1	n3121
112952	scit3	gap.sh	raeirina	R	6-08:31:18	1	n3019
113186	scit3	gap.sh	raemarin	R	5-22:38:59	1	n3017
113420	scit3	gap.sh	raeirina	R	4-23:47:42	1	n3149
113421	scit3	gap.sh	raeirina	R	4-18:39:42	1	n3101
114257	scit3	min3_fre	milchenk	R	1-00:23:34	4	n[3002,3005,

```
3008,3010]  
[pmchnu@access ~]$ cat slurm-114769.out  
3.14159
```


- Відповідним чином заповнюється і форма запуску:

Форма запуску

Кластер та черга: Ім'я задачі:

Робоча тека:

Скрипт запуску: MPI Заданий користувачем

Файл скрипта:

Параметри задачі:

Компіляція

Ліміт часу, хв: Кількість ядер: / на вузол

Умова залежності: Задача:

Опис профілю:

4. Деякі результати тестування

- Тестування проводились на комп'ютері з 4-ядерним процесором Intel Core i3-2310M та на суперкомп'ютері Інституту кібернетики.
- Програми були написані на мові Fortran.
- На звичайному комп'ютері використовувався компілятор Intel з продукту Intel Parallel Studio XE 2015 Cluster Edition та, відповідно, бібліотека Intel MPI 5.
- На суперкомп'ютері теж використовувався компілятор Intel з бібліотекою OpenMPI.
- Замірявся лише «корисний» час – сам час виконання алгоритму.
- Час замірявся у секундах.
- Тестування проводилися на випадково згенерованих даних.

- Програма розв'язування СЛАР з квадратною матрицею із застосуванням паралельного алгоритму LU-розкладу.

		Звичайний комп'ютер			
		1000*1000		2000*2000	
К-ть потоків	Р-ть матриць	OpenMP	MPI	OpenMP	MPI
1		13,45	10,28	105,06	80,45
2		7,55*	5,43	61,37**	41,25
3		8,92	7,13	65,6	54,32
4		7,21	5,61	54,38	41,5
*зустрічалися дані з проміжку [7,19; 8,18]			**зустрічалися дані з проміжку [57,51; 67,89]		
		Суперкомп'ютер			
		1000*1000		2000*2000	
К-ть потоків	Р-ть матриць	OpenMP	MPI	OpenMP	MPI
1		0,82	0,86	12,04	12,07
2		0,17	0,22	9,99	10,13
3		0,27	0,32	9,87	10,12
4		0,11	0,16	9,94	10,28

- Програма розв'язування СЛАР з тридіагональною матрицею із застосуванням паралельного алгоритму редукції. Використовувалась бібліотека OpenMP.

		Звичайний комп'ютер			
		Р-ть матриць	10 млн.		30 млн.
К-ть потоків					
1			1,61		4,74
2			1,1		3,23
3			1,06		3,13
4			0,94		2,75
		Суперкомп'ютер			
		Р-ть матриць	10 млн.		30 млн.
К-ть потоків					
1			1,69		5,06
2			1,67		5,03
3			1,67		5,02
4			1,67		5,02

- Як видно, цей алгоритм не дав виграшу при запуску на суперкомп'ютері.

- Програма розв'язування СЛАР з тридіагональною матрицею із застосуванням паралельного алгоритму прогонки. Використовувалась бібліотека MPI.

		Звичайний комп'ютер			
		10 млн.			30 млн.
К-ть потоків					
1		1,41			4,49
2		0,73			2,33
3		0,83			2,64
4		0,63			2
		Суперкомп'ютер			
		10 млн.			30 млн.
К-ть потоків					
1		0,54			1,58
2		0,42			1,25
3		0,36			1,1
4		0,35			1,06

- Програма розв'язування СЛАР з квадратною матрицею із застосуванням паралельного алгоритму LU-розкладу. Більша кількість тестів на суперкомп'ютері з бібліотекою MPI.

К-ть потоків	P-ть матриць	1000*1000	2000*2000	3000*3000	4000*4000	5000*5000
1		0,86	12,07	42,12	100,78	196,34
2		0,22	10,13	39,77	97,6	191,78
3		0,32	10,12	39,74	97,62	192,34
4		0,16	10,28	40,09	98,03	192,51
5		0,19	4,8	22,38	57,16	114,07
6		0,21	3,54	17,97	46,99	94,56
7		0,23	4,54	21,51	54,78	109
8		0,26	3,64	18,36	47,65	95,33
9		...	1,88	10,71	29,98	61,94
10			2,44	13,75	37,18	75,19
11			2,18	12,29	33,63	68,56
12			1,97	11,17	30,96	63,4
13			1,74	10,02	28,19	58,03
14			1,77	9,27	26,28	54,37
15			1,78	8,69	24,59	51,1
16			1,98	8,24	23,19	48,3
17			...	7,37	21,17	44,46
18				6,97	20,1	42,3
19				6,7	19,35	40,27
20				6,4	18,12	38,19
21				5,94	16,86	35,76
22				5,8	16,23	34,2
23				5,74	15,56	32,86
24				5,86	15,25	31,97



Дякую за увагу!