Виконання завдань розподіленої обробки зображень під управлінням системи CONDOR

Юрій Шийка

науковий керівник:

Роман Шувар

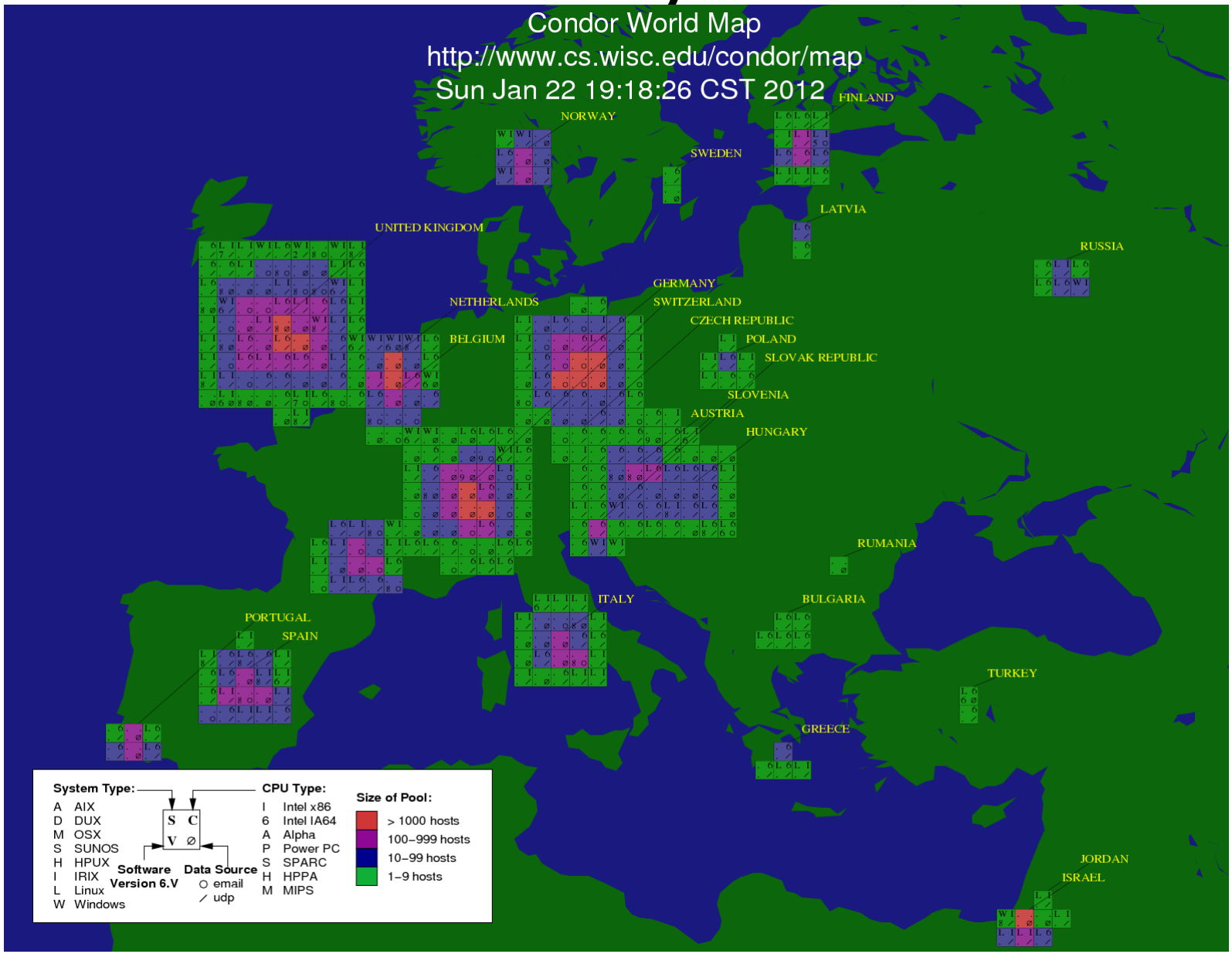
- Condor - спеціалізована система управління навантаженням обчислювальних вузлів для ресурсномістких задач.

Condor включає механізми:

- черга завдань;
- політика планування та схема пріоритетів завдань;
- система моніторингу та управління ресурсами.

Користувачі відправляють на виконання завдання розподілених або паралельних обчислень, Condor поміщає їх у чергу, обирає, коли і на яких обчислювальних вузлах слід їх виконати згідно визначеної політики, ретельно стежить за їх виконанням і інформує користувача після їх завершення.

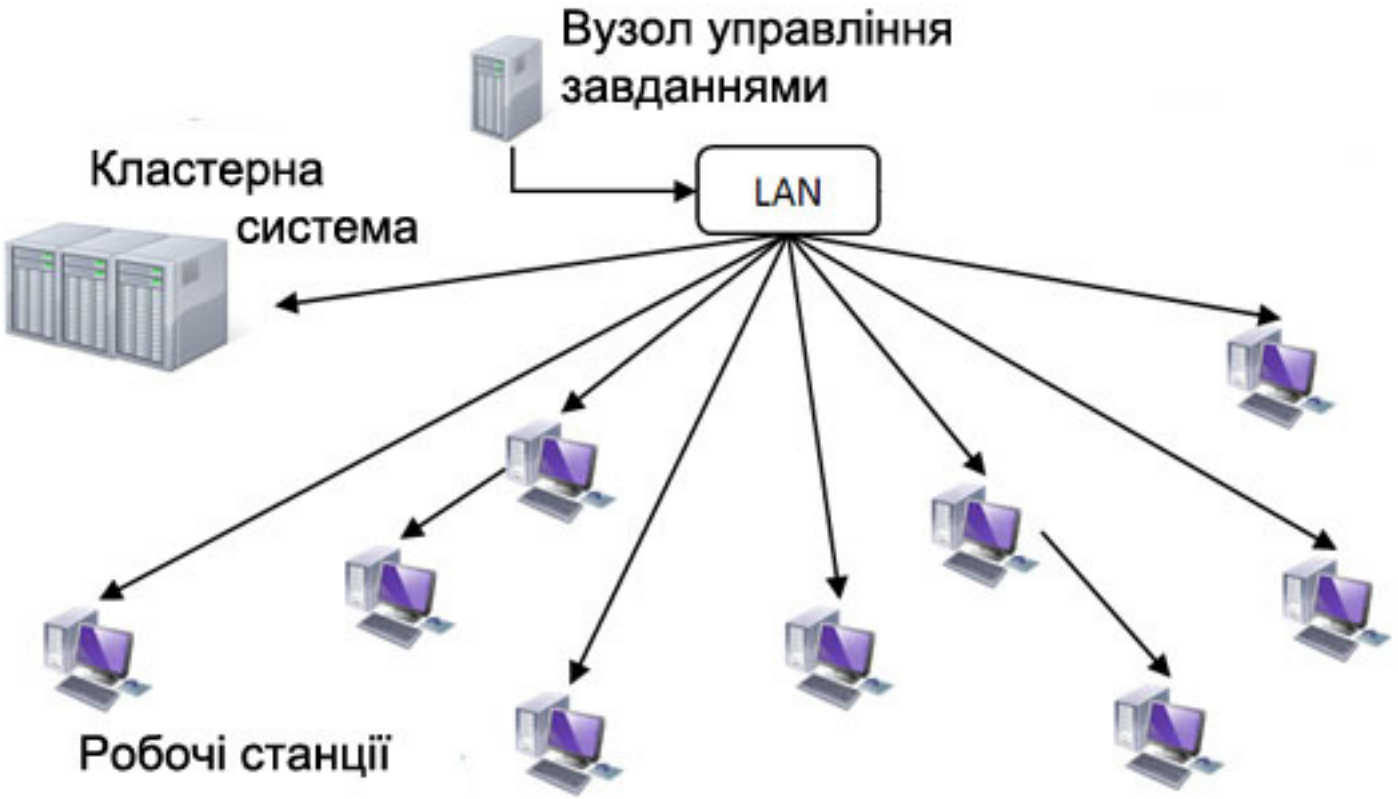
Condor у світі



Переваги Condor

- Порівняно легка адаптація існуючих програм, можливість виконання їх у різних середовищах.
- Стійкість до збоїв. Збій окремої системи не приводить до зупинки обчислень чи втрати результатів.
- Гнучкість і універсальність. Condor дозволяє виконувати як паралельні так і розподілені обчислення в гомогенних, гетерогенних і змішаних системах.

Схема використання Condor



Виконання завдання під управлінням системи Condor

- Підготувати вхідні дані для окремих завдань (вручну чи за допомогою програми генератора завдань).
- Вибрати середовище (набір умов) в якому повинні виконуватись завдання. Забезпечити доступ до вхідних і вихідних файлів завдань тим чи іншим способом (NFS, Condor FileTransfer, …).
- Створити виконавчий файл програми, який повинен виконуватися на робочих вузлах відповідно до вимог системи Condor та встановленої операційної системи.
- Створити файл опису завдання Condor з параметрами виконання, вказаними вхідними і вихідними файлами і т.д.
- Додати завдання в чергу Condor за допомогою команди condor_submit.
- Після виконання завдань отримати вихідні файли (відповідно до способу виконання завдань) та при потребі виконати програму обробки результатів.

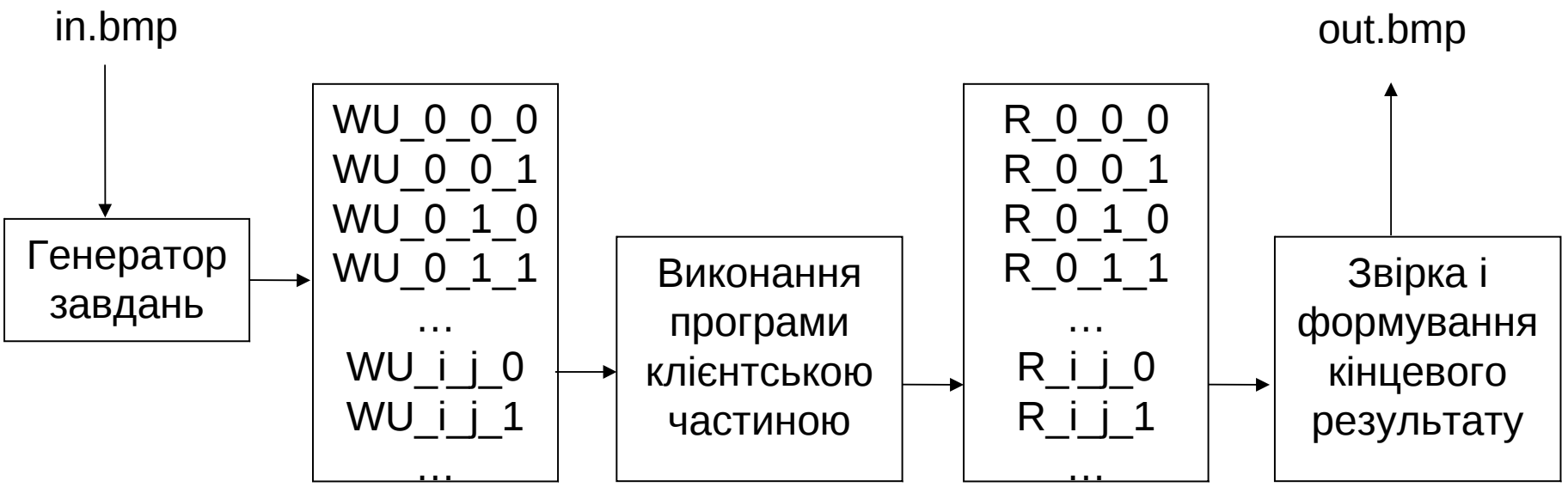
Приклад файлу опису завдань

```
Executable     =       condor_segment
Universe       =       vanilla

Error   =       log/err.segment.$(Process)
Input   =       in/$(Process)_1
Output  =       out_$(Process)_1
Log     =       log/condor_segment.log

Arguments      =       $(Process)

Should_transfer_files   =     YES
When_to_transfer_output =     ON_EXIT
Transfer_input_files    =     in/$(Process)_0, in/$(Process)_1

Queue 162
```

Розподілена обробка растрових зображень

- При обробці растрових зображень, зокрема даних ДЗЗ виникає проблема обробки величезних обсягів зображень. Вихід – формування незалежних підзадач, які можна виконувати розподілено.
- Незалежно від типу обробки (фільтрація, контрастування, сегментація і т.д.) процеси розбиття великої задачі на незалежні підзадачі та формування кінцевого результату є однаковими в більшості випадків. Ці процеси полягають у розбитті зображення на менші шляхом нарізання, та об’єднання результатів підзадач в кінцевий результат за принципом мозаїки.

Схема роботи проекту обробки растрових зображень



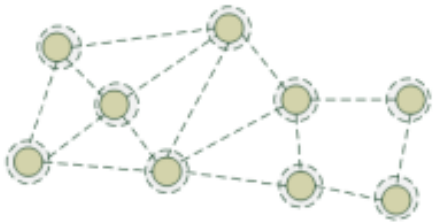
Сегментація зображень дистанційного зондування Землі

Використання в якості вхідних даних зображень високої роздільної здатності дає можливість виділення об’єктів не тільки за характеристиками кольору, а й за складнішими характеристиками – текстура, форма і т.д.

Для поділу зображення на сегменти (класи) в роботі використано алгоритм Краскала, який будує мінімальне остовне дерево даного графа. Тобто об’єднує всі вершини графа в одну область (існує шлях між всіма вершинами) мінімальною сумарною довжиною ребер.

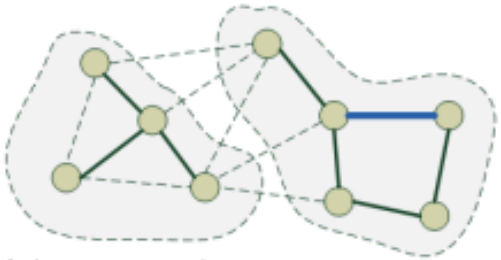
Алгоритм Краскала

1) В початковому графі кожна вершина знаходиться в окремому сегменті

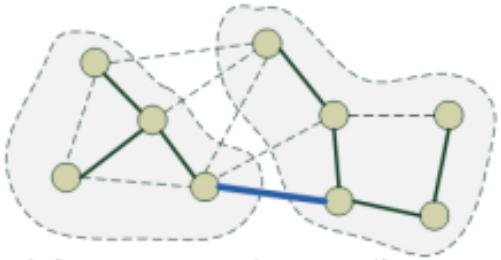


2) Формується список ребер і сортується по зростанню

3) Проходимо по вершинах в зростаючому порядку



4а) таке ребро пропускаємо, бо воно сполучає вершини одного сегмента



4б) таким ребром об'єднуєм сегменти

Використання алгоритму для класифікації

- Будується граф, в якому кожен піксель належить окрмій області і з’єднаний з сусідніми (4-ма чи 8-ми) ребрами з довжиною, яка визначається як різниця властивостей (характеристик);
- Сегменти (класи) об’єднуються до тих пір, поки не виконується умова розрізнення сегментів.

Умова розрізнення сегментів

- максимальний перепад в межах сегмента

$$Int(C) = \max_{e \in C} w(e)$$

- мінімальне ребро, що сполучає різні сегменти

$$Diff(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2} w(v_i, v_j)$$

- признак чи слід об’єднувати сегменти

$$D(C_1, C_2) = \begin{cases} true, Diff(C_1, C_2) > \min(Int(C_1), Int(C_2)) \\ false, \text{інакше } (\text{об’єднати сегменти}) \end{cases}$$

- тобто для того щоб на певному кроці сегменти об’єдналися в один, необхідно щоб перепад на їх границі був меншим за максимальний перепад всередині кожного з цих сегментів

Характеристики для сегментації

В якості характеристик для сегментації використано:

- Різницю кольорів пікселів

$$Dist_{colour} = \sqrt{\left(r_{i,j} - r_{m,n}\right)^2 + \left(g_{i,j} - g_{m,n}\right)^2 + \left(b_{i,j} - b_{m,n}\right)^2}$$

- Текстурні характеристики на основі моментів
- Текстурні характеристики ASM (Angular Second-Moment)

Текстурні характеристики на основі моментів

- Моменти порядку p+q обчислюються за формулою: $$M_{pq} = \sum_{-W/2}^{W/2} \sum_{-W/2}^{W/2} I(m,n) x_m{}^p y_n{}^q$$

де W-розмір вікна, x_m,y_n – координати відносно центрального пікселя

- В роботі використано моменти до другого порядку і вікно розміром 3. Таким чином 6 моментів обчислювались як згортка відповідних масок з зображенням

Текстурні характеристики на основі моментів

- $M_{k=1..6} = m_k \otimes I$

$$m_1 = m_{00} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}; m_2 = m_{10} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}; m_3 = m_{01} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix};$$

$$m_4 = m_{20} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}; m_5 = m_{11} = \begin{bmatrix} 1 & 1 & -1 \\ 0 & 0 & 0 \\ -1 & 1 & 1 \end{bmatrix}; m_6 = m_{02} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix};$$

- Для покращення ефективності характеристик використано нелінійну функцію tanh. Таким чином характеристики обчислювалися за формулою:

$$F_k(i,j) = \frac{1}{L^2} \sum_{(a,b) \in W_{ij}} \left| \tanh\left(0.01 \cdot \left(M_k(a,b) - \overline{M}\right)\right) \right|$$

$W_{i,j}$ – *вікно розміру* L *з центром* (i, j)

$\overline{M}$ – *середнє значення моменту*

Текстурна характеристика ASM

$$ASM = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \left(\frac{P(i,j)}{R} \right)^2$$

де P(i,j) – матриця відносних частот,

R – кількість пар пікселів,

Ng – кількість відліків значень пікселів;

$$P_{\Delta x, \Delta y}(i,j) = \sum_{p=1}^{n} \sum_{q=1}^{m} \begin{cases} 1, I(p,q) = i \wedge I(p+\Delta x, q+\Delta y) = j \\ 0, \textit{інакше} \end{cases}$$

Висновки

- Реалізовано проект розподіленої обробки растрових зображень на основі платформи BOINC. Встановлено та налаштовано серверну частину проекту.
- Реалізовано виконавчу програму сегментації растрових зображень за допомогою алгоритму Краскала адаптовану до розподілених обчислень.
- В якості характеристик для сегментації використано різницю кольорів та текстурні характеристики на основі моментів та характеристику ASM.
- Проведено дослідження ефективності роботи проекту. Встановлено оптимальну стратегію формування та розподілу підзадач.

Приклади результатів класифікації

Приклади результатів класифікації

Приклади результатів класифікації