KLASTERLI MUHITDA DASTURLASH TILLARINI O'QITISH METODIKASI

Mamatova Shirin Faxriyevna
Osiyo texnologiyalari universiteti magistranti
Normamatov Xayriddin Mengniyevich
Osiyo texnologiyalari universiteti magistranti

Annotatsiya

Klasterli muhitda dasturlash tillarini o‘qitish – zamonaviy yuqori unumli hisoblash (HPC) va katta ma’lumotlar tahlili sohalariga mutaxassislar tayyorlashning asosiy omilidir. Ushbu maqolada klaster arxitekturasi (bosh tugun, hisoblash tugunlari, SLURM, MPI) asosida dasturlashni o‘qitishning *metodik jihatlari* tahlil qilinadi. Parallel dasturlash tillari (MPI, OpenMP, Python kutubxonalari) va ularni o‘zlashtirishda qo‘llaniladigan *bosqichma-bosqich yondashuv* korsatilgan: klaster tuzilishini tushunishdan tortib, algoritmlarni parallelashtirishgacha. Maqolada *amaliy misollar* orqali:

1. MPI yordamida massiv yig‘indisini hisoblash (C),
2. OpenMP bilan matritsalarni ko‘paytirish (C),
3. mpi4py va Monte-Karlo usuli orqali π hisoblash (Python)

kabi real vazifalar batafsil yoritiladi. Ta’lim jarayonini tashkil etishda *laboratoriya mashg‘ulotlari, modulli loyihalar, virtual muhitlar*ning ahamiyati vurg‘ulanadi. Qiyinchiliklar (debugging, resurslar boshqaruvi) va ularning yechimlari keltirilgan, shuningdek, klasterli muhit boshqa platformalar (oddiy PC, bulut xizmatlar) bilan *taqqoslanadi*. Xulosada, ushbu metodikaning HPC mutaxassislarini tayyorlashdagi strategik roli ta’kidlangan.

Kalit so‘zlar: Klasterli muhit, Parallel dasturlash, MPI (Message Passing Interface), OpenMP, Dasturlash tillarini o‘qitish metodikasi, Yuqori unumli hisoblash (HPC), Distributiv algoritmlar

Kirish

Zamonaviy ilmiy, muhandislik va biznes muammolari masshtabli hisoblash resurslarini talab qiladi. Klasterli muhitlar – bir nechta mustaqil serverlar (tugunlar) tez aloqa tarmog'i (odatda InfiniBand yoki yuqori tezlikdagi Ethernet) orqali bog'langan arxitektura – bu talabni qondirishning samarali yo'li. Dasturlashni o'qitishda klasterli muhitni joriy etish nafaqat parallel va distributiv hisoblash tamoyillarini o'zlashtirish, balki talabalarni HPC (High-Performance Computing) va katta ma'lumotlar (Big Data) sohalariga tayyorlash imkonini beradi. Ushbu maqola klasterli muhitda dasturlash

tillarini o'qitishning metodik jihatlarini, afzalliklarini, qiyinchiliklarini va namunaviy yondashuvlarini tahlil qiladi.

1. Klasterli muhit va uning o‘qitishdagi ahamiyati

- **Tushuncha:** Klaster – umumiy maqsad uchun birlashtirilgan, bitta tizim sifatida ishlaydigan kompyuterlar guruhi. Tarkibida: bosh tugun (head node), hisoblash tugunlari (compute nodes), tezkor tarmoq, parallel fayl tizimi (masalan, Lustre, NFS), klaster boshqaruv dasturi (masalan, SLURM, PBS Pro).
- **O‘qitishdagi ahamiyati:**
  - **Real dunyo tajribasi:** Talabalar sanoat va ilmiy tadqiqotlarda keng qo‘llaniladigan infrastrukturada ishlash tajribasini olishadi.
  - **Murakkablikni tushunish:** Dastur ishlashiga protsessor soni, tarmoq kechikishi, ma'lumotlar almashinuvining ta'siri kabi omillarni tushunish.
  - **Parallel paradigma:** Dasturlarni parallel va distributiv tarzda tuzish, ishlatish va optimallashtirish ko‘nikmalarini rivojlantirish.
  - **Resurslardan samarali foydalanish:** Cheklangan resurslarni (vaqt, protsessorlar) taqsimlash va boshqarish tamoyillari.
  - **Karyera imkoniyatlari:** HPC, ma'lumotlar tahlili, sun'iy intellekt sohalarida talab ortib bormoqda.

2. Klasterga mos dasturlash tillari va texnologiyalari

Asosiy e'tibor parallel dasturlash interfeyslari (API) va ularni qo‘llab-quvvatlovchi tillarga qaratiladi:

- **MPI (Message Passing Interface):** Standart, ko‘p platformali, ko‘p tilni qo‘llab-quvvatlovchi (C, C++, Fortran, Python). Tugunlararo aloqa uchun (MPI_Send, MPI_Recv, MPI_Bcast, MPI_Reduce). **Afzallik:** Juda katta klasterlarda samarali, nazorat darajasi yuqori. **Kamchilik:** Dasturlash nisbatan murakkab, xatolik topish qiyin.
- **OpenMP (Open Multi-Processing):** Bitta tugundagi ko‘p yadroli protsessorlar uchun. Direktivalar asosida (#pragma omp parallel). **Afzallik:** Oddiy, seriyali dasturni parallelashtirish oson. **Kamchilik:** Faqat bir tugun (shared memory).
- **Python Parallel Kutubxonalari:** mpi4py (MPI uchun), multiprocessing (bitta tugundagi ko‘p jarayonlar), concurrent.futures, Dask (katta ma'lumotlar uchun distributiv hisoblash). **Afzallik:** Kirish osonligi, keng qo‘llanilishi. **Kamchilik:** Interpretatsiya tufayli ba'zi hollarda C/Fortrandan sekinroq.
- **Boshqalar:** Chapel, UPC (Partitioned Global Address Space modellari), Spark (katta ma'lumotlar uchun Java/Scala asosida).

3. O‘qitish metodikasining asosiy tamoyillari

- **Bosqichma-bosqich yondashuv:**

1. **Klaster tuzilishi va interfeys:** Tugunlar, boshqaruvchi, hisob yuritish tizimi (SLURM skriptlari), fayl tizimi tushunchalari.

2. **Parallel dasturlash asoslari:** Jarayonlar/potoklar, sinxronizatsiya, aloqa, ma'lumotlar bo'linishi.

3. **MPI/OpenMP asosiy operatsiyalari:** Xabarlar uzatish, kollektiv operatsiyalar, parallel region, potoklar sinxronizatsiyasi.

4. **Algoritmlarni parallelashtirish:** Domain Decomposition, Master-Worker, Loop Parallelism.

5. **Ishlashni tahlil qilish va optimallashtirish:** Profayllash (gprof, perf, mpiP), tor joylarni aniqlash, balanssizlik muammolari.

- **Amaliyotga urg‘u:** Nazariyadan ko‘ra amaliy topshiriqlar, kichik loyihalar, tajriba o‘tkazish ustuvor.
- **Modullashtirilgan dasturlash:** Dastur alohida funktsiyalarga (modullarga) bo‘linadi, ularni parallel ishlatish mumkin.
- **Xatoliklar va tuzatish:** Klasterdagi xatoliklarning o‘ziga xosligi (tarmoq uzilishi, tugunlardagi nosozliklar), gdb/ddd kabi debuggerlarni parallel rejimda ishlatish, keng log yozish.
- **Haqiqiy muammolarni yechish:** Ilmiy hisoblash, ma'lumotlarni qayta ishlash, simulyatsiya kabi sohalardan misollar.

4. Amaliy misollar va metodik izohlar

- **4.1. Oddiy Parallel Yig‘indi (MPI - C)**
- **Muammo:** Massiv elementlari yig‘indisini N ta protsessorga taqsimlab hisoblash.
- **Metodik maqsad:** MPI_Init, MPI_Comm_size, MPI_Comm_rank, MPI_Scatter, MPI_Reduce kabi asosiy funksiyalarni o‘rganish, ma'lumotlarni bo‘lish va natijani birlashtirish.

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);
    int world_size, world_rank;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    // Bosh protsess massiv yaratadi va to'ldiradi
    int *data = NULL;
    int data_count = 1000;
    if (world_rank == 0) {
        data = (int*)malloc(data_count * sizeof(int));
```

```
        for (int i = 0; i < data_count; i++) data[i] = i + 1;
    }

    // Har bir protsessga bo'linadigan qism hajmini aniqlash
    int chunk_size = data_count / world_size;
    int *local_data = (int*)malloc(chunk_size * sizeof(int));

    // Bosh protsess ma'lumotlarni bo'lib yuboradi (Scatter)
    MPI_Scatter(data, chunk_size, MPI_INT, local_data, chunk_size, MPI_INT,
0, MPI_COMM_WORLD);

    // Har bir protsess o'z qismining yig'indisini hisoblaydi
    int local_sum = 0;
    for (int i = 0; i < chunk_size; i++) local_sum += local_data[i];
    printf("Protsess %d: Lokal yig'indi = %d\n", world_rank, local_sum);

    // Lokal yig'indilarni yig'ib, bosh protsessga yuborish (Reduce)
    int global_sum;
    MPI_Reduce(&local_sum, &global_sum, 1, MPI_INT, MPI_SUM, 0,
MPI_COMM_WORLD);

    // Bosh protsess yakuniy natijani chiqaradi
    if (world_rank == 0) {
        printf("Global yig'indi = %d\n", global_sum);
        free(data);
    }
    free(local_data);
    MPI_Finalize();
    return 0;
}
```

Ishga tushirish (SLURM skripti): sbatch -N 4 -n 16 ./my_mpi_sum_program

- **4.2. Matritsalarni Ko‘paytirish (OpenMP - C)**
- **Muammo:** Ikkita katta matritsani ko‘paytirishni bir tugundagi ko‘p yadrolar orqali tezlashtirish.
- **Metodik maqsad:** Loop Parallelism, parallel for direktivasi, private/shared o‘zgaruvchilar, ish yukini muvozanatlash (schedule).

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
```

```
#define N 1024

int main() {
    int **A = (int**)malloc(N * sizeof(int*));
    int **B = (int**)malloc(N * sizeof(int*));
    int **C = (int**)malloc(N * sizeof(int*));
    for (int i = 0; i < N; i++) {
        A[i] = (int*)malloc(N * sizeof(int));
        B[i] = (int*)malloc(N * sizeof(int));
        C[i] = (int*)malloc(N * sizeof(int));
        // Matritsalarni to'ldirish (misol uchun)
        for (int j = 0; j < N; j++) {
            A[i][j] = rand() % 10;
            B[i][j] = rand() % 10;
        }
    }

    // Asosiy ko'paytirish tsikli (Parallel qilingan)
    #pragma omp parallel for shared(A, B, C) private(i, j, k) schedule(dynamic)
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            C[i][j] = 0;
            for (int k = 0; k < N; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }

    // Xotirani tozalash...
    return 0;
}
```

Ishga tushirish: export OMP_NUM_THREADS=8; ./my_omp_matrix_mult

- **4.3. Monte-Karlo usuli bilan π hisoblash (MPI + Python - mpi4py)**
- **Muammo:** π sonini tasodifiy nuqtalar yordamida hisoblash (klasterda).
- **Metodik maqsad:** mpi4py dan foydalanish, Master-Worker modeli, natijalarni yig‘ish from mpi4py import MPI

```
import numpy as np
import sys
```

```
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()

# Parametrlar
n_points = 1000000  # Umumiy nuqtalar
points_per_proc = n_points // size  # Har bir protsessga

# Har bir protsess o'ziga tasodifiy nuqtalar yaratadi va hisoblaydi
np.random.seed(rank)  # Har bir protsess uchun boshqa urug'
local_inside = 0
for i in range(points_per_proc):
    x, y = np.random.random(2)  # [0, 1) oralig'ida
    if x**2 + y**2 <= 1.0:
        local_inside += 1

# Barcha protsesslardagi 'local_inside' larni yig'amiz (0- protsessga)
total_inside = comm.reduce(local_inside, op=MPI.SUM, root=0)

# 0-protsess natijani hisoblab chiqaradi
if rank == 0:
    pi_estimate = 4.0 * total_inside / n_points
    print(f"Taxminiy π qiymati: {pi_estimate} (Nuqtalar: {n_points})")
```

Ishga tushirish: mpiexec -n 4 python pi_monte_carlo_mpi.py

5. O‘quv jarayonini tashkil etish metodikasi

- **Laboratoriya mashg‘ulotlari:** Har bir nazariy mavzu kichik amaliy topshiriqlar bilan mustahkamlanadi (masalan, har bir MPI funksiyasi uchun alohida misol).
- **Proyektlar:** Guruh (2-4 talaba) loyihalari. Masalan: klasterda ishlaydigan oddiy qidiruv tizimi, katta matritsalar bilan ishlash, tabiiy tilni qayta ishlash algoritmi.
- **Baholash:**
  - **Dastur ishlashi:** To‘g‘rilik, parallel samaradorlik (tezlanish - Speedup, samaradorlik - Efficiency), xotira sarfi.
  - **Kod sifat:** Aniqlik, modullashtirish, izohlar.
  - **Hisobot:** Muammo tavsifi, parallel yechim, natijalar tahlili (grafiklar), xulosalar.
  - **Demostratsiya va himoya:** Guruh ishini namoyish qilish va savollarga javob berish.

- **Virtual laboratoriya:** Talabalarga masofadan kirish imkoniyati (OpenStack, Kubernetes klasterlari, bulut xizmatlari - AWS ParallelCluster, GCP Dataproc).
- **Raqobatbardoshlik:** "Eng tez" dastur, "Eng samarali" algoritm tanlovi.

6. Qiyinchiliklar va yechimlar

- **Qiyinchiliklar:**
  - **Murakkablik:** Parallel dasturlash seriyali dasturlashdan murakkab.
  - **Dasturiy ta'minot va resurslar:** Klasterni sozlash, boshqarish, dasturiy ta'minot o‘rnatish talab qiladi.
  - **Xatoliklarni topish:** Parallel dasturlarda xatolik topish (race conditions, deadlocks) juda qiyin.
  - **Tezligi pasaytiruvchi omillar:** Tarmoq kechikishi, balanssiz yuk, sinxronizatsiya narxi.
  - **Talabalarning tayyorgarligi:** Algoritmik tafakkur, tizim dasturlash bilimi talab etiladi.
- **Yechimlar:**
  - **Bosqichma-bosqich o‘rganish:** Oddiy misollardan murakkablarga.
  - **Simulyatorlar:** Klaster muhitini simulyatsiya qiluvchi dasturlar (MPICH ni o‘rnatmasdan ishlatish).
  - **Kuchli debuggerlar:** TotalView, DDT (Arm Forge), gdb bilan mpirun.
  - **Vizualizatsiya:** Parallel dastur jarayonini grafik tasvirlash (Paraver, Vampir).
  - **Namunalar va shablonlar:** To‘g‘ri ishlaydigan kod namunalari, SLURM skript shablonlari.
  - **Faol jamoaviy ishlash:** Masalalarni muhokama qilish, tajriba almashish.

7. Klaster vs Boshqa Muhitlar (Taqqoslash)

- **Oddiy Kompyuter (PC):**
  - **Afzallik:** Oddiy, kirish oson, shaxsiy.
  - **Kamchilik:** Parallel imkoniyatlar cheklangan (faqat OpenMP, bir necha yadro).
  - **O‘qitishda:** Asosiy tushunchalar, OpenMP, multiprocessing uchun yaxshi.
- **Bulutli Muhit (AWS, Azure, GCP):**
  - **Afzallik:** Cheksiz masshtablash, turli xil konfiguratsiyalar, tayyor HPC xizmatlari.
  - **Kamchilik:** Xarajat (tajriba uchun), tarmoq kechikishi kattaroq bo‘lishi mumkin, konfiguratsiya murakkab.
  - **O‘qitishda:** Haqiqiy tarqatilgan tizimlar, katta ma'lumotlar bilan ishlash.
- **Klaster (On-Premise):**

- **Afzallik:** Yuqori ishonchlilik, juda tez aloqa (InfiniBand), to‘g‘ridan-to‘g‘ri nazorat.
- **Kamchilik:** Yuqori narx, texnik xizmat ko‘rsatish talabi, fizik joy.
- **O‘qitishda:** "Toza" HPC tajribasi, ekstremal parallel dasturlash, tizim darajasidagi boshqaruv.

Xulosa

Klasterli muhitda dasturlashni o‘qitish – talabalarni zamonaviy texnologik talablarga tayyorlashning muhim bosqichi. U parallel va distributiv hisoblashning chuqur tamoyillarini, HPC texnologiyalarini, katta ma'lumotlarni qayta ishlash usullarini o‘zlashtirish imkonini beradi. Muvaffaqiyatli metodika bosqichma-bosqich yondashuv, kuchli amaliy komponent, real dunyo muammolariga e'tibor va klaster muhitining o‘ziga xos qiyinchiliklarini bartaraf etish usullarini o‘z ichiga oladi. Quyidagilarga e'tibor qaratish kerak:

1. **Tayanch tushunchalar:** Klaster arxitekturasi, MPI/OpenMP asoslari, parallel algoritmlar.
2. **Amaliyot:** Kichik laboratoriya topshiriqlaridan tortib to katta guruh loyihalarigacha.
3. **Tahlil:** Dastur ishlashini o‘lchash, optimallashtirish, xatoliklarni tuzatish.
4. **Zamonaviy vositalar:** mpi4py, Dask, kuchli debugger va profayllar.
5. **Tashkil etish:** Virtual muhitlar, aniq baholash mezonlari, faol o‘rganish muhiti.

Klasterli muhitda dasturlashni o‘zlashtirgan mutaxassislar ilmiy tadqiqotlar, ma'lumotlar tahlili, AI va murakkab modellashtirish sohalarida muhim rol o‘ynaydi. O‘quv dasturlariga ushbu mavzuni integratsiyalash kelajak mutaxassislarini tayyorlashda strategik ahamiyatga ega.

Adabiyotlar

1. Normamatov, X. (2025). IMPROVING THE METHODOLOGY OF TEACHING PROGRAMMING LANGUAGES BASED ON NETWORK TECHNOLOGIES. *International Journal of Artificial Intelligence*, *1*(2), 656-662.
2. Normamatov, X. (2025). APPLYING INTERNATIONAL EXPERIENCES IN TEACHING PROGRAMMING TO HIGHER EDUCATION SPECIALIST STUDENTS: CHALLENGES AND SOLUTIONS. *International Journal of Artificial Intelligence*, *1*(2), 648-650.
3. Normamatov, X. (2025). CHALLENGES AND SOLUTIONS IN TEACHING PROGRAMMING: AN EXPLORATION OF GLOBAL AND LOCAL PERSPECTIVES. *International Journal of Artificial Intelligence*, *1*(2), 651-655.
4. Menginiyevich, N. X., & Bahodir o‘g‘li, N. B. (2025). IQTISODIY MASALALARDA CHIZIQLI DASTURLASH MASALALARINI YECHISHDA SIMPLEKS USUL ALGORITMI VA UNING TAHLILI. *Pedagogs*, *79*(1), 133-136.

6. Mengniyevich, N. H., & Abdirashid o‘g, O. R. A. (2025). OB’EKTLARNING KESISHISH NUQTALARI VA OPTIMIZATSIYA MASALALARINI ALGEBRAIK VA TRANSSENDENT TENGLAMALARNI TAQRIBIY YECHISH USULLARI BILAN HAL QILISH. *Pedagogs*, *79*(1), 148-150.
7. Mengniyevich, N. X., & Farxod o‘g‘li, X. D. (2025). MA’LUMOTLARNI INTELLEKTUAL TAHLIL QILISH VA MASHINALI O ‘QITISH: MUAMMO VA YECHIMLARI. *Pedagogs*, *79*(1), 137-147.
8. Mengniyevich, N. X., & Farhod o‘g, X. J. E. (2025). JAMIYAT TARAQQIYOTIDA ROBOTOTEXNIKA, AVTOMATLASHTIRISH VA SANOAT INTELLEKTUAL TIZIMLARI KIRIB KELISHINING SALBIY VA IJOBIY TOMONLARI. *Pedagogs*, *79*(1), 128-132.
9. Нормаматов, Х. М., & Абдуллаева, С. У. (2015). ЭФФЕКТИВНОСТЬ ПРИМЕНЕНИЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ" Э-БОЛЬНИЦА". In *Инновации в технологиях и образовании* (pp. 117-119).
10. Нормаматов, Х. М. (2014). ЛИНЕЙНЫЕ СИСТЕМЫ В ЦИФРОВОЙ ОБРАБОТКЕ СИГНАЛОВ. In *Инновации в строительстве глазами молодых специалистов* (pp. 239-241).
11. Шеров, Ж. Э., & Нормаматов, Х. М. (2015). АВТОМАТИЗАЦИЯ УПРАВЛЕНИЯ ВЫСШЕГО УЧЕБНОГО ЗАВЕДЕНИЯ. In *Инновации в технологиях и образовании* (pp. 178-182).
12. Ruziev, Z. I., Kadirov, L. K., Ostonova, M. E., Baratov, B. S., & Ortiq, S. (2020). The role of income tax individuals in replenishing state budget revenues. *Journal of Advanced Research in Dynamical and Control Systems*, *12*(7 Special Issue), 2033-2037.
13. Ibragimovna, I. F. (2024). THE POSITION OF WOMEN-GIRLS IN THE SOCIAL-ECONOMIC LIFE OF UZBEKISTAN (1941-1945). *International journal of advanced research in education, technology and management*, *3*(4), 272-275.
14. Nazarov, R. (2022). THE INFLUENCE OF SOCIAL NETWORKS ON THE SPIRITUAL IMAGE OF THE INDIVIDUAL. *INTERNATIONAL JOURNAL OF SOCIAL SCIENCE & INTERDISCIPLINARY RESEARCH ISSN: 2277-3630 Impact factor: 8.036*, *11*(01), 58-60.
15. Hamraeva, N. (2022). The History of the Khiva Khanate's Relations Between Neighboring Countries in" Shajaraii Turk". *Central Asian Journal of Social Sciences and History*, *3*(1), 16-18.
16. Akhmadova, N. A. Q. (2021). ESTABLISHING RELATIONS OF UZBEKISTAN WITH THE UNO AND ITS SPECIALIZED AGENCIES. *Current Research Journal of History*, *2*(06), 76-81.
17. Ochilova, O. R. (2024). XALQ TABOBATI VAKILLARI VA ULARNING MUOLAJA USULLARI. «Ёш олимлар ахборотномаси»–«Вестник молодых ученых», (Спецвыпуск), 71-74.