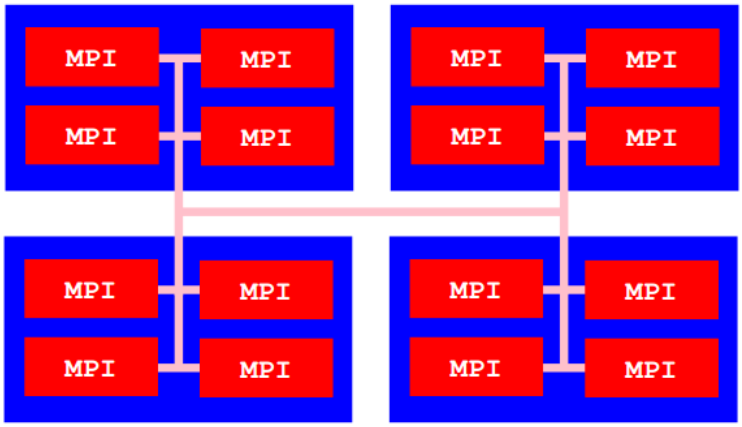
10 - дәріс. MPI және OpenMP-ді гибридті қолдану.

Жоспар:

1. Гибридті тәсілдің негізгі артықшылықтары
2. MPI және OpenMP гибридін пайдалану бойынша кеңестер
3. MPI және OpenMP гибридті қолдану тиімді болатын тапсырмалардың мысалдары
4. Қарапайым мысалды әзірлеу: массив элементтерінің қосындысын есептеу

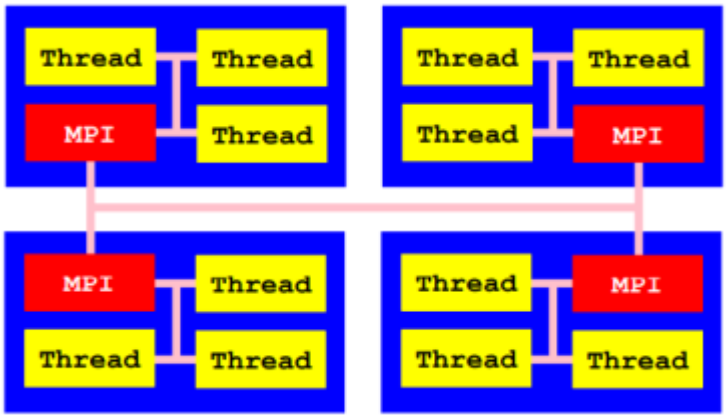
Гибридті тәсілдің негізгі артықшылықтары

Кластерде көптеген процессорлар (немесе көп ядролы процессорлар) бар, біз әр процессорда (немесе ядрода) орындалатын MPI процесін іске қосу арқылы параллель бағдарламаны орындай аламыз. Бірнеше MPI процестері бір мультипроцессорда (немесе көп ядролы процессорда) орындалуы мүмкін, бірақ бұл процестер арасындағы өзара әрекеттесу әлі де хабар алмасуға негізделген (25 - сурет).



25-сурет. Процесстердің мультипроцессорда MPI арақылы әрекеттесуі

Тағы бір тәсіл-гибридті бағдарламаны әзірлеу, онда әр мультипроцессорда (немесе көп ядролы процессорда) тек бір MPI процесі орындалады, содан кейін параллель аймақтарды орындау үшін әр машинадағы процессорлар (немесе ядролар) санына тең ағындар жиынтығы іске қосылады (26 - сурет).



26-сурет. Гибридті бағдарлама схемасы

Гибридті MPI және OpenMP параллельді есептеулерде екі модельдің де күшті жақтарын пайдалану және белгілі бір қолданба түрлері үшін жоғары өнімділікке қол жеткізу үшін қолданылады. Гибридті тәсілді қолданудың бірнеше себептері:

1. Масштабталу:

- MPI таратылған жады жүйелерінде (кластерлерде) есептеулерді параллельдеу үшін жақсы жұмыс істейді, бұл оны ауқымды параллелизмге қолайлы етеді.
- OpenMP, керісінше, түйін немесе көп ядролы процессор ішіндегі Ортақ жадпен параллель болуға арналған.

2. Жад иерархиясы:

- MPI кластерде тиімді байланыс орната отырып, бөлінген жады бар түйіндер арасындағы байланысты қамтамасыз етеді.
- OpenMP жалпы жад пен кэш консистенциясының артықшылықтарын пайдалана отырып, түйін ішіндегі параллелизмді пайдалануда тиімді.

3. Ресурстарды оңтайлы пайдалану:

- MPI жұмыс жүктемесін кластерге тарату арқылы бірнеше түйіндерді пайдалануға мүмкіндік береді.
- OpenMP ресурстарды бір машинада пайдалануды оңтайландыру арқылы әр түйіннің ішінде бірнеше ядроларды пайдалануға мүмкіндік береді.

4. Әр түрлі архитектураларға бейімделу:

- MPI және OpenMP гибридті кодтары әртүрлі аппараттық архитектураларға бейімделеді: бөлінген жады кластерлерінен бастап ортақ жады бар көп ядролы процессорларға дейін.
- Бұл бейімделу өнімділігі жоғары есептеу ортасының (HPC) әртүрлі түрлерінде қолданбаларды масштабтауды жеңілдетеді.

5. Жүктемені теңестіру:

- MPI таратылған жүйенің түйіндері арасындағы жүктемені теңестірудің табиғи әдісін ұсынады.
- OpenMP көп ядролы деңгейдегі жүктеме теңгерімсіздігін жоя отырып, әр түйіннің ішінде егжей-тегжейлі параллелизмді қамтамасыз етеді.

6. Бағдарламалаудың қарапайымдылығы:

- OpenMP прагмаға негізделген қарапайым директиваларына байланысты ортақ жадпен параллель болу үшін жиі пайдалану оңай деп саналады.
- MPI және OpenMP комбинациясы әзірлеушілерге екі модельдің де күшті жақтарын пайдалануға мүмкіндік береді, бұл таза бөлінген жад немесе ортақ жад тәсілдерімен байланысты қиындықтарды азайтады.

7. Өнімділікті оңтайландыру:

- Гибридті модельдер байланыс шығындарын Мұқият басқару және түйіндер арасындағы деректерді беруді азайту арқылы өнімділікті оңтайландыруға мүмкіндік береді.

MPI және OpenMP гибридін пайдалану бойынша кеңестер

MPI-ді OpenMP-мен біріктірудің ең оңай және қауіпсіз әдісі - ешқашан OpenMP параллель аймақтарында MPI функцияларын қолданбау. Параллель OpenMP аймақтарында MPI шақыруларын пайдалану әдетте екі параллель бағдарламалау моделі арасындағы өзара әрекеттесуге байланысты ықтимал мәселелерге байланысты ұсынылмайды. OpenMP параллель аймақтарында MPI шақыруларын пайдаланудан аулақ болудың бірнеше себептері:

- MPI және OpenMP параллельді бағдарламалаудың тәуелсіз модельдері ретінде жасалған және оларды бір уақытта қолдану параллелизмнің кірістірілген мәселелеріне әкелуі мүмкін.

- MPI өзі көбінесе ағындарды қолдана отырып жүзеге асырылады және OpenMP параллель аймағында MPI қоңырауларын қолдануға тырысу қақтығыстар мен күтпеген мінез-құлыққа әкелуі мүмкін.
- MPI іске асырулары табиғи түрде қауіпсіз болмауы мүмкін, яғни MPI қоңырауларын көп ағынды контексте, мысалы, OpenMP параллель аймағында пайдалану деректердің бұзылуына, жарыс жағдайларына немесе құлыптарға әкелуі мүмкін.
- MPI қоңыраулары OpenMP параллель аймақтарына шашыраңқы болған кезде кодты сүйемелдеудің оқылуы мен ыңғайлылығына қауіп төнуі мүмкін. Мұндай кодты түсіну және сақтау қиын болуы мүмкін.

Егер MPI және OpenMP екеуі де қажет болса, оларды әдетте кодтың бөлек бөлімдерінде пайдалану ұсынылады: түйін немесе кластер деңгейіндегі процестер арасындағы өзара әрекеттесу үшін MPI және түйін ішіндегі ортақ жадпен параллель болу үшін OpenMP. Қақтығыстарды болдырмау және тапсырмалардың нақты бөлінуін қамтамасыз ету үшін MPI шақыруларын OpenMP параллель аймақтарынан тыс жерде орындалуын қамтамасыз ету қажет.

MPI және OpenMP гибридті қолдану тиімді болатын тапсырмалардың мысалдары

MPI және OpenMP гибридті қолданылуы көбінесе тапсырма кластер түйіндері арасында таралатын (MPI көмегімен) және әр түйінде көптеген ядролар бар (OpenMP көмегімен) сценарийлерде тиімді болады. Гибридті тәсіл тиімді болуы мүмкін тапсырмалардың кейбір мысалдары:

1. Физика мен инженериядағы ауқымды есептеу:
- Мысал: жоғары энергия физикасындағы бөлшектердің өзара әрекеттесуін модельдеу.
- Көбінесе тапсырманың бөліктерін әртүрлі түйіндерге бөлу үшін MPI, ал әр түйіннің ішінде параллель орындау үшін OpenMP қажет.
2. Климатологиядағы кешенді есептеу:
- Мысал: климатты ауқымды модельдеу.
- MPI ғаламдық тордың әртүрлі бөлімдері арасындағы өзара әрекеттесу үшін қолданылады, ал OpenMP әр түйінде параллель орындау үшін қолданылады.
3. Молекулалық динамика саласындағы есептеулер:
- Мысал: молекулалардың қозғалысы мен өзара әрекеттесуін зерттеу.
- MPI молекулалық фрагменттерді әртүрлі түйіндерге тарату үшін қолданылады, ал OpenMP әр фрагменттің ішінде параллель есептеулер жүргізу үшін қолданылады.
4. Биоинформатикадағы үлкен деректерді талдау
- Мысал: геномдық деректерді өңдеу және талдау.
- MPI әртүрлі деректер үлгілерін өңдеу үшін пайдаланылуы мүмкін, ал OpenMP әрбір үлгі ішіндегі талдауды жеделдету үшін пайдаланылуы мүмкін.

Қорытындылай келе, MPI және OpenMP гибридті тәсілі-бұл әр түйінде қол жетімді ресурстарды тиімді пайдалана отырып, таратылған жад жүйелерінде тиімді масштабтауға болатын параллельді қосымшаларды әзірлеудің қуатты стратегиясы. Бұл комбинация әсіресе ауқымды үлестірілген есептеулер мен ортақ жадпен егжей-тегжейлі параллелизмді біріктіретін қолданбалар үшін пайдалы.

Қарапайым мысалды әзірлеу: массив элементтерінің қосындысын есептеу

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <mpi.h>
#include <omp.h>
```

```
// Массив үшін кездейсоқ сандарды құру функциясы
void generateRandomArray(int* array, int size) {
    for (int i = 0; i < size; ++i) {
        array[i] = rand() % 100;  // Қажет болса, диапазонды реттеңіз
    }
}
// OpenMP көмегімен массивтің қосындысын есептеу функциясы
    int calculateSumOpenMP(int* array, int size) {
    int sum = 0;

    #pragma omp parallel for reduction(+:sum)
    for (int i = 0; i < size; ++i) {
        sum += array[i];
    }
    return sum;
}

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);

    int world_rank, world_size;

    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    const int array_size = 1000000;  // Қажет болса өлшемін өзгертіңіз

    // Массивтің жергілікті бөлігіне жад бөлу
    int* local_array = new int[array_size / world_size];

    // Түбірлік процесте кездейсоқ сандарды құру
    if (world_rank == 0) {
        srand(time(NULL));
        int* global_array = new int[array_size];
        generateRandomArray(global_array, array_size);

        // Массивті барлық процестерге тарату
        MPI_Scatter(global_array, array_size / world_size, MPI_INT,
                    local_array, array_size / world_size, MPI_INT, 0, MPI_COMM_WORLD);
        delete[] global_array;
    } else {
        // Массивтің жергілікті бөлігін алу
        MPI_Scatter(NULL, array_size / world_size, MPI_INT,
                    local_array, array_size / world_size, MPI_INT, 0, MPI_COMM_WORLD);
    }
```

```
// OpenMP көмегімен локальді қосындыны есептеу
    int local_sum = calculateSumOpenMP(local_array, array_size / world_size);

    // Глобальді соманы алу үшін локальді қосындыны қысқарту.
    int global_sum;
    MPI_Reduce(&local_sum, &global_sum, 1, MPI_INT, MPI_SUM, 0,
MPI_COMM_WORLD);

    // Нәтижені түбірлік процесте шығару
    if (world_rank == 0) {
        std::cout << "Global Sum: " << global_sum << std::endl;
    }
    delete[] local_array;
    MPI_Finalize();

    return 0;
}
```

Тапсырма:

Дәрісте көрсетілген массив элементтерінің қосындысын есептейтін гибридті параллель бағдарламаны Param–Bilim суперкомпьютерінде орындаңыз. Матрицаның өлшемін бірнеше рет өзгертіп, есептеу уақытын салыстырыңыз.

Бақылау сұрақтары:

1. Гибридті тәсілді қолданудың себептерін атаңыз.
2. MPI-ді OpenMP-мен біріктірудің ең оңай және қауіпсіз әдісі қандай? Оның себептерін атаңыз.
3. Гибридті тәсіл тиімді болуы мүмкін тапсырмаларға мысал келтіріңіз