Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №8

Паралельне програмування в бібліотеці MPI.

Виконав:

студент групи ІП-22

Сочка О. О.

Перевірив:

Корочкін О. В.

Київ - 2015

**Технічне завдання**

Математична задача:

**Структура ПКС ЛП**



Рис. 3.1 Структура ПКС із локальною пам’яттю

**Етап 1. Побудова паралельного алгоритму**

Константи, що використовуються:

*  – розмірність векторів і матриць;
* Q – кількість ядер на першому ярусі;
* K – кількість розгалужень на другому ярусі
* P (сумарна кількість процесів) = 1 + Q + Q\*K
* .

**Етап 2. Розробка алгоритмів процесів (задач)**

**Задача Т0:**

1. Введення MB, MC, a, MO
2. Передати в Т1 …TQ.
3. Прийняти MAH від Т1 …TQ.
4. Обчислення
5. Вивід МА.

**Задача Ti (i = 1..Q)**

1. Прийняти від Т0
2. Передати в
3. Обчислення
4. Прийняти MAHH від
5. Передати результати MAH в T0.

**Задача Т11 …TQK.**

1. Прийняти від Т0.
2. Обчислення
3. Передати результати MAHH в Ti.

## Етап 3. Схема взаємодії задач

# Етап 4. Розробка програми

1 #include <mpi.h>

2 #include <iostream>

3 #include <vector>

4

5 using namespace std;

6

7 const int N = 1600;

8 const int Q = 3;

9 const int K = 4;

10 const int P = Q \* K + Q + 1;

11 const int HH = N / P;

12 const int H = (N - HH) / Q;

13 static\_assert((N - HH) % Q == 0, "N should be divisible by Q");

14 static\_assert(N % P == 0, "N should be divisible by P");

15

16 void calc(int\* MBh, int\* MC, int a, int\* MOh, int\* MAh)

17 {

18 for (int r = 0; r < HH; ++r)

19 {

20 for (int c = 0; c < N; ++c)

21 {

22 int tmp = 0;

23 for (int k = 0; k < N; ++k)

24 {

25 tmp += MBh[r\*N + k] \* MC[k\*N + c];

26 }

27 MAh[r\*N + c] = tmp + a \* MOh[r \* N + c];

28 }

29 }

30 }

31

32 int main(int argc, char\* argv[])

33 {

34 MPI\_Init(&argc, &argv);

35 int rank, size;

36 MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

37 MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

38

39 if (size != P && rank == 0)

40 {

41 cerr << "MPI\_Comm\_size not corresponding to constants in program!" << endl;

42 exit(1);

43 }

44

45 cout << "Starting process " << rank << endl;

46

47 if (rank == 0)

48 {

49 // flat 2d arrays

50 vector<int>

51 MB(N \* N, 1),

52 MC(N \* N, 1),

53 MO(N \* N, 1),

54 MA(N \* N);

55

56 int a = 1;

57

58 for (int i = 1; i <= Q; ++i)

59 {

60 MPI\_Send(MB.data() + HH \* N + (i - 1) \* H \* N, H \* N, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

61 MPI\_Send(MC.data(), N \* N, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

62 MPI\_Send(MO.data() + HH \* N + (i - 1) \* H \* N, H \* N, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

63 MPI\_Send(&a, 1, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

64 }

65

66 calc(MB.data(), MC.data(), a, MO.data(), MA.data());

67

68

69 for (int i = 1; i <= Q; ++i)

70 {

71 MPI\_Recv(MA.data() + HH \* N + (i - 1) \* H \* N, H \* N, MPI\_INT, i, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

72 }

73

74 if (N <= 50)

75 {

76 for (int i = 0; i < N; ++i)

77 {

78 for (int j = 0; j < N; ++j)

79 {

80 cout << MA[i \* N + j] << ' ';

81 }

82 cout << endl;

83 }

84 }

85 }

86 else if (rank <= Q)

87 {

88 // flat 2d arrays

89 vector<int>

90 MB(H \* N),

91 MC(N \* N),

92 MO(H \* N),

93 MA(H \* N);

94

95 int a;

96

97 MPI\_Recv(MB.data(), MB.size(), MPI\_INT, 0, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

98 MPI\_Recv(MC.data(), MC.size(), MPI\_INT, 0, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

99 MPI\_Recv(MO.data(), MO.size(), MPI\_INT, 0, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

100 MPI\_Recv(&a, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

101

102 for (int i = (rank - 1) \* K + Q + 1; i <= (rank - 1) \* K + Q + K; ++i)

103 {

104 int m = (i - Q - 1) % K + 1;

105 MPI\_Send(MB.data() + m \* HH \* N, HH \* N, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

106 MPI\_Send(MC.data(), N \* N, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

107 MPI\_Send(MO.data() + m \* HH \* N, HH \* N, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

108 MPI\_Send(&a, 1, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

109 }

110

111 calc(MB.data(), MC.data(), a, MO.data(), MA.data());

112

113 for (int i = (rank - 1) \* K + Q + 1; i <= (rank - 1) \* K + Q + K; ++i)

114 {

115 int m = (i - Q - 1) % K + 1;

116 MPI\_Recv(MA.data() + m \* HH \* N, HH \* N, MPI\_INT, i, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

117 }

118

119 MPI\_Send(MA.data(), MA.size(), MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

120 }

121 else

122 {

123 // flat 2d arrays

124 vector<int>

125 MB(HH \* N),

126 MC(N \* N),

127 MO(HH \* N),

128 MA(HH \* N);

129

130 int a = 1;

131

132 const int parent = (rank - Q - 1) / K + 1;

133 MPI\_Recv(MB.data(), MB.size(), MPI\_INT, parent, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

134 MPI\_Recv(MC.data(), MC.size(), MPI\_INT, parent, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

135 MPI\_Recv(MO.data(), MO.size(), MPI\_INT, parent, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

136 MPI\_Recv(&a, 1, MPI\_INT, parent, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

137

138 calc(MB.data(), MC.data(), a, MO.data(), MA.data());

139 MPI\_Send(MA.data(), MA.size(), MPI\_INT, parent, 0, MPI\_COMM\_WORLD);

140 }

141

142 cout << "Finishing process " << rank << endl;

143 MPI\_Finalize();

144 }