15-418/15-618 Parallel Computer Architecture and Programming

Homework 4, Parallel VLSI Wire Routing via MPI

Siqi Guo(AndrewID: siqiguo) January 22, 2025

Use MPI to write a message passing version of the same application as the previous assignment. (Wire Routing under message-passing programming models (only have PRIVATE address spaces).)

- Learn how to compile and run MPI binaries on the same host and across the nodes of a cluster.
- Evaluate where optimizations are most valuable.

Notes:

Sending messages is relatively expensive, experienced MPI programmers generally try to structure their code to avoid frequent communication

At the very end, it turns out that machine 28 is a problem where `perf` is not installed. See 2024 Fall Piazza @368

1. **[40 points] Design and performance debugging for the message-passing approach.**

   i. Similar to the OpenMP implementation. After implementing the Across-Wire MPI paralleled program, we have correctness test results attached at the end of the report.

   ii. The pseudocode provided.

```
// Variable Declaration ...
MPI_Init(&argc, &argv);           // Initialize MPI
MPI_Comm_rank(MPI_COMM_WORLD, &pid); // Get process rank
MPI_Comm_size(MPI_COMM_WORLD, &nproc); // Get total number of processes

parse_arguments(...);
initialize_routing(...);

MPI_Bcast(&num_wires, 1, MPI_INT, ROOT, MPI_COMM_WORLD);
MPI_Bcast(&dim_y, 1, MPI_INT, ROOT, MPI_COMM_WORLD);
MPI_Bcast(&dim_x, 1, MPI_INT, ROOT, MPI_COMM_WORLD);
initialize_receiver_vector_size(...);

// MPI_Bcast only work on contiguous memory buffers.
MPI_Bcast(wires.data(), num_wires * sizeof(Wire), MPI_BYTE, ROOT, MPI_COMM_WORLD);
MPI_Bcast(flat_occupancy.data(), dim_y * dim_x * sizeof(int), MPI_BYTE, ROOT, MPI_COMM_WORLD);

// All processes must wait until the broadcast is complete before proceeding to the next step.
If pid is not ROOT
    For each row in occupancy grid
        Copy data from flat_occupancy to 2D occupancy array
    End For
End If

// Calculate work span for each process
Determine work span based on process rank and total wires
Adjust work span for edge cases (e.g., remainder wires)
```

```
For each simulated annealing iteration
    For wires assigned to this process (from startIndex to endIndex)
        Perform routing algorithms
    End For
End For

// send data to root process
if (pid != ROOT) {
    MPI_Send(&wires[startIndex], workSpan * sizeof(Wire), MPI_BYTE, ROOT, TAG, MPI_COMM_WORLD);
} else {
    MPI_Status status;
    for (int source = 1; source < nproc; source++) {
        MPI_Recv(&wires[source * workSpan], workSpan * sizeof(Wire), MPI_BYTE, source, TAG,
            MPI_COMM_WORLD,
                  &status);
    }
}
// This is another synchronization point where the root process must wait for all sends to
     complete before it can continue.

// Update occupancy grid on ROOT process
If pid is ROOT
    Update occupancy grid with received data
End If

// Cleanup
MPI_Finalize();
```

2. **[15 points] Experimental results from the GHC machines**

   ~~**[20 points] Experimental results from the PSC machines**~~

Correctness Tests

```
siqiguo@ghc56:~/private/15-618-Parallel-Computing/asst4/code$ ./checker.pl

--------------
Running tests on ghc56.ghc.andrew.cmu.edu
--------------

Test : easy with 1 cores
Correctness passed!
Your time : 0.6370366600
Target Time: 0.49
Your cost : 121978
Target Cost: 122050

Test : easy with 2 cores
Correctness passed!
Your time : 0.3346642290
Target Time: 0.31
Your cost : 122405
Target Cost: 122050

Test : easy with 4 cores
Correctness passed!
Your time : 0.2201978630
Target Time: 0.19
Your cost : 123904
Target Cost: 122050

Test : easy with 8 cores
Correctness passed!
Your time : 0.1405227760
Target Time: 0.13
Your cost : 125892
Target Cost: 122050

Test : medium with 1 cores
Correctness passed!
Your time : 10.9571411100
Target Time: 9.52
Your cost : 601215
Target Cost: 601213

Test : medium with 2 cores
Correctness passed!
Your time : 9.9203150420
Target Time: 6.12
Your cost : 612134
Target Cost: 601213

Test : medium with 4 cores
Correctness passed!
Your time : 8.0959574360
Target Time: 4.04
Your cost : 621736
Target Cost: 601213

Test : medium with 8 cores
Correctness passed!
Your time : 6.8566017200
Target Time: 3.41
Your cost : 641197
Target Cost: 601213

Test : hard with 1 cores
Correctness passed!
Your time : 13.4858831170
Target Time: 11.15
Your cost : 1031998
Target Cost: 1032198

Test : hard with 2 cores
Correctness passed!
Your time : 11.2784738400
Target Time: 7.08
Your cost : 1052142
Target Cost: 1032198

Test : hard with 4 cores
Correctness passed!
Your time : 10.8472393710
Target Time: 4.1
Your cost : 1063102
Target Cost: 1032198

Test : hard with 8 cores
Correctness passed!
Your time : 7.5720181200
Target Time: 3.1
Your cost : 1076051
Target Cost: 1032198
```

```
Test : extreme with 1 cores
Correctness passed!
Your time : 59.0921018140
Target Time: 46.81
Your cost : 23592484
Target Cost: 23613045

Test : extreme with 2 cores
Correctness passed!
Your time : 30.5852387770
Target Time: 28.95
Your cost : 26268192
Target Cost: 23613045

Test : extreme with 4 cores
Correctness passed!
Your time : 16.0450465490
Target Time: 15.2
Your cost : 29031434
Target Cost: 23613045

Test : extreme with 8 cores
Correctness passed!
Your time : 8.5529003170
Target Time: 11
Your cost : 30714138
Target Cost: 23613045

------------
Score table:
------------
```

Test Name	Core Num	Target Time	Your Time	Target Cost	Your Cost	Score
easy	1	0.49	0.6370366600	122050	121978	0.769186501762709
easy	2	0.31	0.3346642290	122050	122405	1
easy	4	0.19	0.2201978630	122050	123904	1
easy	8	0.13	0.1405227760	122050	125892	1
medium	1	9.52	10.9571411100	601213	601215	2
medium	2	6.12	9.9203150420	601213	612134	1.2338317833838
medium	4	4.04	8.0959574360	601213	621736	0.998028962463532
medium	8	3.41	6.8566017200	601213	641197	0.994661827900367
hard	1	11.15	13.4858831170	1032198	1031998	2.48037148993481
hard	2	7.08	11.2784738400	1032198	1052142	1.88323352089275
hard	4	4.1	10.8472393710	1032198	1063102	1.13392906520381
hard	8	3.1	7.5720181200	1032198	1076051	1.22820625262846
extreme	1	46.81	59.0921018140	23613045	23592484	3.16861296606714
extreme	2	28.95	30.5852387770	23613045	26268192	3.59568637232437
extreme	4	15.2	16.0450465490	23613045	29031434	3.25344521390159
extreme	8	11	8.5529003170	23613045	30714138	3.0752020453903
					Total score:	28.8143960018536/40