Alex Miller

CS 374

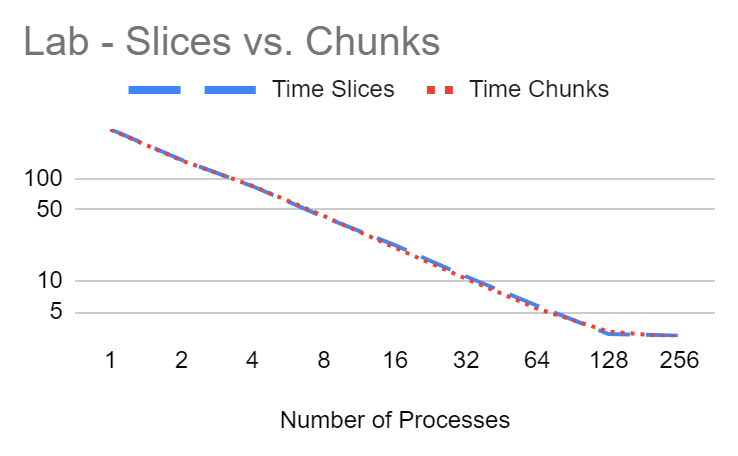
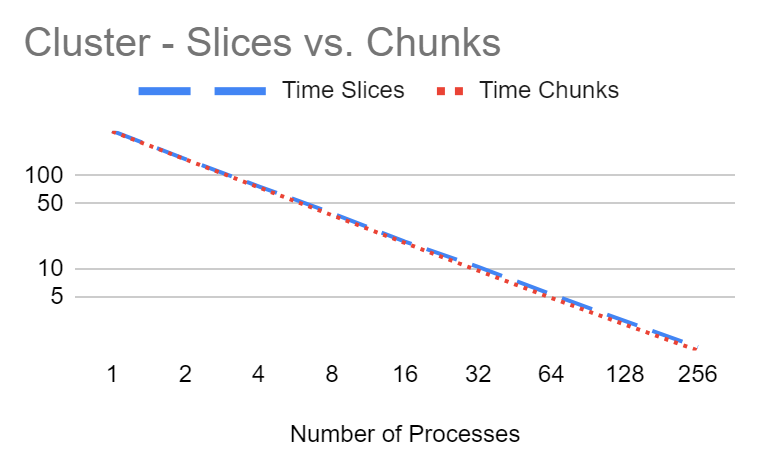
Homework 01

November 1 2023

Homework 01 - Analysis

In this analysis, we will examine the performance data of two programs, using slices and chunks, in terms of their execution time, specifically focusing on how their performances change as the number of processes used in the execution varies. We will also compare the programs' performances between running them in the lab environment and on the cluster.

We can see that using slices and chunks come with very similar execution time, it is even hard to differentiate them on the charts while using a logarithmic scale, but we can see that using chunks is usually a little bit faster. Since they are so similar in execution I believe that it doesn’t really matter that much, especially at this scale, but if you were dealing with a much larger amount of computation needed to be done then I could see the reason for using chunks as you do get the benefit of memory localization.

But we can actually see that when we are using fewer nodes that the lab environment actually has better performance than the cluster, like for a singular node the lab environment is 1-6 seconds faster, I think this is because when using the fewer nodes there is less communication overhead, especially for a singular node, and the lab computers probably have better single thread performance, but once we get to using a higher number of nodes the cluster overpasses the lab environment in execution time.  
 In summary, our analysis of two programs using slices and chunks revealed similar execution times, with a slight advantage for chunks. The lab environment outperformed the cluster when using fewer nodes due to possible reduced communication overhead and potential single-thread performance. However, for larger computations, the cluster proved more efficient. 

|  |  |  |  |
| --- | --- | --- | --- |
| Lab Environment | | | |
| Number of Processes | Time Slices | Time Chunks | Solutions |
| 1 | 300.944534 | 299.192404 | 81 |
| 2 | 151.193209 | 149.42961 | 81 |
| 4 | 83.899365 | 84.514193 | 81 |
| 8 | 42.067142 | 43.152553 | 81 |
| 16 | 22.429013 | 21.191742 | 81 |
| 32 | 11.12628 | 10.588799 | 81 |
| 64 | 5.787139 | 5.419851 | 81 |
| 128 | 3.074431 | 3.241049 | 81 |
| 256 | 2.96124 | 2.861351 | 81 |

|  |  |  |
| --- | --- | --- |
| Cluster | | |
| Number of Processes | Time Slices | Time Chunks |
| 1 | 305.750363 | 301.444183 |
| 2 | 152.069845 | 150.738517 |
| 4 | 77.330361 | 75.445363 |
| 8 | 39.818579 | 37.875253 |
| 16 | 19.623562 | 18.996477 |
| 32 | 10.501128 | 9.562237 |
| 64 | 5.318334 | 4.859265 |
| 128 | 2.756348 | 2.519087 |
| 256 | 1.440578 | 1.323184 |

Script started on 2023-11-01 00:07:40-04:00 [TERM="xterm-256color" TTY="/dev/pts/1" COLUMNS="80" LINES="24"]

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ cat circuitSatisfiability.c

/\* circuitSatifiability.c solves the Circuit Satisfiability

\* Problem using a brute-force sequential solution.

\*

\* The particular circuit being tested is "wired" into the

\* logic of function 'checkCircuit'. All combinations of

\* inputs that satisfy the circuit are printed.

\*

\* 16-bit version by Michael J. Quinn, Oregon State University, Sept 2002.

\* Extended to 32 bits by Joel C. Adams, Calvin University, Sept 2013.

\*/

#include <stdio.h> // printf()

#include <limits.h> // UINT\_MAX

#include <mpi.h>

int checkCircuit (int, long);

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

long i; // loop variable (64 bits)

int id = 0; // process id

int count = 0; // number of solutions

MPI\_Init(&argc, &argv);

printf ("\nProcess %d is checking the circuit...\n", id);

double startTime = 0.0, totalTime = 0.0;

startTime = MPI\_Wtime();

for (i = 0; i <= UINT\_MAX; ++i) {

count += checkCircuit (id, i);

}

totalTime = MPI\_Wtime() - startTime;

printf ("Process %d finished in time %f secs.\n", id, totalTime);

fflush (stdout);

printf("\nA total of %d solutions were found.\n\n", count);

MPI\_Finalize();

return 0;

}

/\* EXTRACT\_BIT is a macro that extracts the ith bit of number n.

\*

\* parameters: n, a number;

\* i, the position of the bit we want to know.

\*

\* return: 1 if 'i'th bit of 'n' is 1; 0 otherwise

\*/

#define EXTRACT\_BIT(n,i) ( (n & (1<<i) ) ? 1 : 0)

/\* checkCircuit() checks the circuit for a given input.

\* parameters: id, the id of the process checking;

\* bits, the (long) rep. of the input being checked.

\*

\* output: the binary rep. of bits if the circuit outputs 1

\* return: 1 if the circuit outputs 1; 0 otherwise.

\*/

#define SIZE 32

int checkCircuit (int id, long bits) {

int v[SIZE]; /\* Each element is one of the 32 bits \*/

int i;

for (i = 0; i < SIZE; i++) {

v[i] = EXTRACT\_BIT(bits,i);

}

if ( ( (v[0] || v[1]) && (!v[1] || !v[3]) && (v[2] || v[3])

&& (!v[3] || !v[4]) && (v[4] || !v[5])

&& (v[5] || !v[6]) && (v[5] || v[6])

&& (v[6] || !v[15]) && (v[7] || !v[8])

&& (!v[7] || !v[13]) && (v[8] || v[9])

&& (v[8] || !v[9]) && (!v[9] || !v[10])

&& (v[9] || v[11]) && (v[10] || v[11])

&& (v[12] || v[13]) && (v[13] || !v[14])

&& (v[14] || v[15]) )

&&

( (v[16] || v[17]) && (!v[17] || !v[19]) && (v[18] || v[19])

&& (!v[19] || !v[20]) && (v[20] || !v[21])

&& (v[21] || !v[22]) && (v[21] || v[22])

&& (v[22] || !v[31]) && (v[23] || !v[24])

&& (!v[23] || !v[29]) && (v[24] || v[25])

&& (v[24] || !v[25]) && (!v[25] || !v[26])

&& (v[25] || v[27]) && (v[26] || v[27])

&& (v[28] || v[29]) && (v[29] || !v[30])

&& (v[30] || v[31]) ) )

{

printf ("%d) %d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d \n", id,

v[31],v[30],v[29],v[28],v[27],v[26],v[25],v[24],v[23],v[22],

v[21],v[20],v[19],v[18],v[17],v[16],v[15],v[14],v[13],v[12],

v[11],v[10],v[9],v[8],v[7],v[6],v[5],v[4],v[3],v[2],v[1],v[0]);

fflush (stdout);

return 1;

} else {

return 0;

}

}

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ cat circuitSatisfiabilitySlices.c

/\* circuitSatifiabilitySlices.c solves the Circuit Satisfiability in slices

\* Problem using a brute-force sequential solution.

\*

\* The particular circuit being tested is "wired" into the

\* logic of function 'checkCircuit'. All combinations of

\* inputs that satisfy the circuit are printed.

\*

\* 16-bit version by Michael J. Quinn, Oregon State University, Sept 2002.

\* Extended to 32 bits by Joel C. Adams, Calvin University, Sept 2013.

\*

\* Alex Miller

\* circuitSatifiabilitySlices.c: Finds all combos of checkCircuit() in slices

\* Calvin University

\* 25 Oct 2023

\* CS 374 HW01

\*/

#include <stdio.h> // printf()

#include <limits.h> // UINT\_MAX

#include <mpi.h>

int checkCircuit (int, long);

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

int numProcesses = 0;

int totalCount = 0;

const int MASTER = 0;

long i; // loop variable (64 bits)

int id = 0; // process id

int count = 0; // number of solutions

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &id);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &numProcesses);

printf ("\nProcess %d is checking the circuit...\n", id);

double startTime = 0.0, totalTime = 0.0;

startTime = MPI\_Wtime();

for (i = id; i <= UINT\_MAX; i+= numProcesses ) {

count += checkCircuit (id, i);

}

MPI\_Reduce(&count, &totalCount, 1, MPI\_INT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

totalTime = MPI\_Wtime() - startTime;

MPI\_Finalize();

if (id == MASTER) {

printf ("Process %d finished in time %f secs.\n", id, totalTime);

printf("\nA total of %d solutions were found.\n\n", totalCount);

}

fflush (stdout);

return 0;

}

/\* EXTRACT\_BIT is a macro that extracts the ith bit of number n.

\*

\* parameters: n, a number;

\* i, the position of the bit we want to know.

\*

\* return: 1 if 'i'th bit of 'n' is 1; 0 otherwise

\*/

#define EXTRACT\_BIT(n,i) ( (n & (1<<i) ) ? 1 : 0)

/\* checkCircuit() checks the circuit for a given input.

\* parameters: id, the id of the process checking;

\* bits, the (long) rep. of the input being checked.

\*

\* output: the binary rep. of bits if the circuit outputs 1

\* return: 1 if the circuit outputs 1; 0 otherwise.

\*/

#define SIZE 32

int checkCircuit (int id, long bits) {

int v[SIZE]; /\* Each element is one of the 32 bits \*/

int i;

for (i = 0; i < SIZE; i++) {

v[i] = EXTRACT\_BIT(bits,i);

}

if ( ( (v[0] || v[1]) && (!v[1] || !v[3]) && (v[2] || v[3])

&& (!v[3] || !v[4]) && (v[4] || !v[5])

&& (v[5] || !v[6]) && (v[5] || v[6])

&& (v[6] || !v[15]) && (v[7] || !v[8])

&& (!v[7] || !v[13]) && (v[8] || v[9])

&& (v[8] || !v[9]) && (!v[9] || !v[10])

&& (v[9] || v[11]) && (v[10] || v[11])

&& (v[12] || v[13]) && (v[13] || !v[14])

&& (v[14] || v[15]) )

&&

( (v[16] || v[17]) && (!v[17] || !v[19]) && (v[18] || v[19])

&& (!v[19] || !v[20]) && (v[20] || !v[21])

&& (v[21] || !v[22]) && (v[21] || v[22])

&& (v[22] || !v[31]) && (v[23] || !v[24])

&& (!v[23] || !v[29]) && (v[24] || v[25])

&& (v[24] || !v[25]) && (!v[25] || !v[26])

&& (v[25] || v[27]) && (v[26] || v[27])

&& (v[28] || v[29]) && (v[29] || !v[30])

&& (v[30] || v[31]) ) )

{

printf ("%d) %d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d \n", id,

v[31],v[30],v[29],v[28],v[27],v[26],v[25],v[24],v[23],v[22],

v[21],v[20],v[19],v[18],v[17],v[16],v[15],v[14],v[13],v[12],

v[11],v[10],v[9],v[8],v[7],v[6],v[5],v[4],v[3],v[2],v[1],v[0]);

fflush (stdout);

return 1;

} else {

return 0;

}

}

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ cat circuitSatisfiabilityS Chunks.c

/\* circuitSatifiabilityChunks.c solves the Circuit Satisfiability in chunks

\* Problem using a brute-force sequential solution.

\*

\* The particular circuit being tested is "wired" into the

\* logic of function 'checkCircuit'. All combinations of

\* inputs that satisfy the circuit are printed.

\*

\* 16-bit version by Michael J. Quinn, Oregon State University, Sept 2002.

\* Extended to 32 bits by Joel C. Adams, Calvin University, Sept 2013.

\*

\* Alex Miller

\* circuitSatifiabilityChunks.c: Finds all combos of checkCircuit() in chunks

\* Calvin University

\* 25 Oct 2023

\* CS 374 HW01

\*/

#include <stdio.h> // printf()

#include <limits.h> // UINT\_MAX

#include <mpi.h>

#include "parallelLoopChunks.h"

int checkCircuit (int, long);

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

int numProcesses = 0;

int totalCount = 0;

const int MASTER = 0;

long i; // loop variable (64 bits)

int id = 0; // process id

int count = 0; // number of solutions

unsigned start = -1, stop = -1;

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &id);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &numProcesses);

printf ("\nProcess %d is checking the circuit...\n", id);

getChunkStartStopValues(id, numProcesses, UINT\_MAX, &start, &stop);

double startTime = 0.0, totalTime = 0.0;

startTime = MPI\_Wtime();

for (i = start; i < stop; ++i ) {

count += checkCircuit (id, i);

}

MPI\_Reduce(&count, &totalCount, 1, MPI\_INT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

totalTime = MPI\_Wtime() - startTime;

MPI\_Finalize();

if (id == MASTER) {

printf ("Process %d finished in time %f secs.\n", id, totalTime);

printf("\nA total of %d solutions were found.\n\n", totalCount);

}

fflush (stdout);

return 0;

}

/\* EXTRACT\_BIT is a macro that extracts the ith bit of number n.

\*

\* parameters: n, a number;

\* i, the position of the bit we want to know.

\*

\* return: 1 if 'i'th bit of 'n' is 1; 0 otherwise

\*/

#define EXTRACT\_BIT(n,i) ( (n & (1<<i) ) ? 1 : 0)

/\* checkCircuit() checks the circuit for a given input.

\* parameters: id, the id of the process checking;

\* bits, the (long) rep. of the input being checked.

\*

\* output: the binary rep. of bits if the circuit outputs 1

\* return: 1 if the circuit outputs 1; 0 otherwise.

\*/

#define SIZE 32

int checkCircuit (int id, long bits) {

int v[SIZE]; /\* Each element is one of the 32 bits \*/

int i;

for (i = 0; i < SIZE; i++) {

v[i] = EXTRACT\_BIT(bits,i);

}

if ( ( (v[0] || v[1]) && (!v[1] || !v[3]) && (v[2] || v[3])

&& (!v[3] || !v[4]) && (v[4] || !v[5])

&& (v[5] || !v[6]) && (v[5] || v[6])

&& (v[6] || !v[15]) && (v[7] || !v[8])

&& (!v[7] || !v[13]) && (v[8] || v[9])

&& (v[8] || !v[9]) && (!v[9] || !v[10])

&& (v[9] || v[11]) && (v[10] || v[11])

&& (v[12] || v[13]) && (v[13] || !v[14])

&& (v[14] || v[15]) )

&&

( (v[16] || v[17]) && (!v[17] || !v[19]) && (v[18] || v[19])

&& (!v[19] || !v[20]) && (v[20] || !v[21])

&& (v[21] || !v[22]) && (v[21] || v[22])

&& (v[22] || !v[31]) && (v[23] || !v[24])

&& (!v[23] || !v[29]) && (v[24] || v[25])

&& (v[24] || !v[25]) && (!v[25] || !v[26])

&& (v[25] || v[27]) && (v[26] || v[27])

&& (v[28] || v[29]) && (v[29] || !v[30])

&& (v[30] || v[31]) ) )

{

printf ("%d) %d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d \n", id,

v[31],v[30],v[29],v[28],v[27],v[26],v[25],v[24],v[23],v[22],

v[21],v[20],v[19],v[18],v[17],v[16],v[15],v[14],v[13],v[12],

v[11],v[10],v[9],v[8],v[7],v[6],v[5],v[4],v[3],v[2],v[1],v[0]);

fflush (stdout);

return 1;

} else {

return 0;

}

}

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpicc circuitSatisfiability.c -Wall -ansi -ped

dantic -std=c99 -o circuitSatisfiability

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ [Kmpicc circuitSatisfiability.c -Wall -ansi -ped

dantic -std=c99 -o circuitSatisfiabilitylitySlices[A[11Pcat circuitSatisfiabilityChunks.c

[K[A

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ cat circuitSatisfiabilityChunks.c mpicc circuitSatisfiability.c -Wall -ansi -ped

dantic -std=c99 -o circuitSatisfiabilitySlices

[A

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpicc circuitSatisfiability.c -Wall -ansi -ptyS.c -Wall -ansi -pe[1@d[A

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpicc circuitSatisfiabilitySlices.c -Wall -ans[5@i -pe[A

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpicc circuitSatisfiabilitySlices.c

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ [Kmpicc circuitSatisfiabilitySlices.c -Wall -ans

si -pedantic -std=c99 -o circuitSatisfiabilitySliceses Chunks[Asfiability.c -Wall -ansi -ped[12Pantic -std=c99 -o circuitSatisfiability[A circuitSatisfiabilitySlices.c -Wall -ansi -pedantic -std=c99 -o circuitSatisfiabilityChunks

[A

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpicc circuitSatisfiabilitySlices.c -Wall -aices.c -Wall -ansi[1P[A

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpicc circuitSatisfiabilitySlice.c -Wall -ansi [1P[A

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpicc circuitSatisfiabilitySlic.c -Wall -ansi -[1P[A

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpicc circuitSatisfiabilitySli.c -Wall -ansi -p[1P[A

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpicc circuitSatisfiabilitySl.c -Wall -ansi -pe[1P[A

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpicc circuitSatisfiabilityS.c -Wall -ansi -ped[1P[A

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpicc circuitSatisfiabilityC.c -Wall -ansi -pe[1@d[A

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpicc circuitSatisfiabilityChunks.c -Wall -ans[5@i -pe[A

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpicc circuitSatisfiabilityChunks

/bin/ld: /tmp/ccQTUGBE.o: undefined reference to symbol 'ceil@@GLIBC\_2.2.5'

/bin/ld: /lib/x86\_64-linux-gnu/libm.so.6: error adding symbols: DSO missing from command line

collect2: error: ld returned 1 exit status

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ [K

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$

[K]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ make

mpicc circuitSatisfiabilityChunks.c -Wall -ansi -pedantic -std=c99 -o circuitSatisfiabilityChunks -lm

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ cat Makefile

PROG = circuitSatisfiabilityChunks

SRC = $(PROG).c

CC = mpicc

CFLAGS = -Wall -ansi -pedantic -std=c99

LFLAGS = -o $(PROG) -lm

$(PROG): $(PROG).c parallelLoopChunks.h

$(CC) $(SRC) $(CFLAGS) $(LFLAGS)

clean:

rm -f $(PROG) \*.o \*~ \*#

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpirun -np 4 ./spmd 8 ./circuitSatisfiabilitySlices

Process 0 is checking the circuit...

Process 2 is checking the circuit...

Process 4 is checking the circuit...

Process 7 is checking the circuit...

Process 3 is checking the circuit...

Process 5 is checking the circuit...

Process 1 is checking the circuit...

Process 6 is checking the circuit...

5) 10011001111101011001100111110101

5) 10011001111101011001101111110101

5) 10011001111101011001110111110101

5) 10011001111101101001100111110101

5) 10011001111101101001101111110101

5) 10011001111101101001110111110101

5) 10011001111101111001100111110101

5) 10011001111101111001101111110101

5) 10011001111101111001110111110101

7) 10011001111101011001100111110111

7) 10011001111101011001101111110111

7) 10011001111101011001110111110111

7) 10011001111101101001100111110111

7) 10011001111101101001101111110111

7) 10011001111101101001110111110111

7) 10011001111101111001100111110111

7) 10011001111101111001101111110111

7) 10011001111101111001110111110111

6) 10011001111101011001100111110110

6) 10011001111101011001101111110110

6) 10011001111101011001110111110110

6) 10011001111101101001100111110110

6) 10011001111101101001101111110110

6) 10011001111101101001110111110110

6) 10011001111101111001100111110110

6) 10011001111101111001101111110110

6) 10011001111101111001110111110110

5) 10011011111101011001100111110101

5) 10011011111101011001101111110101

5) 10011011111101011001110111110101

5) 10011011111101101001100111110101

5) 10011011111101101001101111110101

5) 10011011111101101001110111110101

5) 10011011111101111001100111110101

5) 10011011111101111001101111110101

5) 10011011111101111001110111110101

7) 10011011111101011001100111110111

7) 10011011111101011001101111110111

7) 10011011111101011001110111110111

7) 10011011111101101001100111110111

7) 10011011111101101001101111110111

7) 10011011111101101001110111110111

7) 10011011111101111001100111110111

7) 10011011111101111001101111110111

7) 10011011111101111001110111110111

6) 10011011111101011001100111110110

6) 10011011111101011001101111110110

6) 10011011111101011001110111110110

6) 10011011111101101001100111110110

6) 10011011111101101001101111110110

6) 10011011111101101001110111110110

6) 10011011111101111001100111110110

6) 10011011111101111001101111110110

6) 10011011111101111001110111110110

5) 10011101111101011001100111110101

5) 10011101111101011001101111110101

5) 10011101111101011001110111110101

5) 10011101111101101001100111110101

5) 10011101111101101001101111110101

5) 10011101111101101001110111110101

5) 10011101111101111001100111110101

5) 10011101111101111001101111110101

5) 10011101111101111001110111110101

7) 10011101111101011001100111110111

7) 10011101111101011001101111110111

7) 10011101111101011001110111110111

7) 10011101111101101001100111110111

7) 10011101111101101001101111110111

7) 10011101111101101001110111110111

7) 10011101111101111001100111110111

7) 10011101111101111001101111110111

7) 10011101111101111001110111110111

6) 10011101111101011001100111110110

6) 10011101111101011001101111110110

6) 10011101111101011001110111110110

6) 10011101111101101001100111110110

6) 10011101111101101001101111110110

6) 10011101111101101001110111110110

6) 10011101111101111001100111110110

6) 10011101111101111001101111110110

6) 10011101111101111001110111110110

Process 0 finished in time 90.842588 secs.

A total of 81 solutions were found.

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ mpirun -np 8 ./circuitSatisfiabilitySlices8[1P[1@1[1@2 ./circuitSatisfiabilitySlices yChunks

Process 0 is checking the circuit...

Process 3 is checking the circuit...

Process 8 is checking the circuit...

Process 11 is checking the circuit...

Process 10 is checking the circuit...

Process 4 is checking the circuit...

Process 5 is checking the circuit...

Process 6 is checking the circuit...

Process 2 is checking the circuit...

Process 9 is checking the circuit...

Process 7 is checking the circuit...

Process 1 is checking the circuit...

7) 10011001111101011001100111110101

7) 10011001111101011001100111110110

7) 10011001111101011001100111110111

7) 10011001111101011001101111110101

7) 10011001111101011001101111110110

7) 10011001111101011001101111110111

7) 10011001111101011001110111110101

7) 10011001111101011001110111110110

7) 10011001111101011001110111110111

7) 10011001111101101001100111110101

7) 10011001111101101001100111110110

7) 10011001111101101001100111110111

7) 10011001111101101001101111110101

7) 10011001111101101001101111110110

7) 10011001111101101001101111110111

7) 10011001111101101001110111110101

7) 10011001111101101001110111110110

7) 10011001111101101001110111110111

7) 10011001111101111001100111110101

7) 10011001111101111001100111110110

7) 10011001111101111001100111110111

7) 10011001111101111001101111110101

7) 10011001111101111001101111110110

7) 10011001111101111001101111110111

7) 10011001111101111001110111110101

7) 10011001111101111001110111110110

7) 10011001111101111001110111110111

7) 10011011111101011001100111110101

7) 10011011111101011001100111110110

7) 10011011111101011001100111110111

7) 10011011111101011001101111110101

7) 10011011111101011001101111110110

7) 10011011111101011001101111110111

7) 10011011111101011001110111110101

7) 10011011111101011001110111110110

7) 10011011111101011001110111110111

7) 10011011111101101001100111110101

7) 10011011111101101001100111110110

7) 10011011111101101001100111110111

7) 10011011111101101001101111110101

7) 10011011111101101001101111110110

7) 10011011111101101001101111110111

7) 10011011111101101001110111110101

7) 10011011111101101001110111110110

7) 10011011111101101001110111110111

7) 10011011111101111001100111110101

7) 10011011111101111001100111110110

7) 10011011111101111001100111110111

7) 10011011111101111001101111110101

7) 10011011111101111001101111110110

7) 10011011111101111001101111110111

7) 10011011111101111001110111110101

7) 10011011111101111001110111110110

7) 10011011111101111001110111110111

7) 10011101111101011001100111110101

7) 10011101111101011001100111110110

7) 10011101111101011001100111110111

7) 10011101111101011001101111110101

7) 10011101111101011001101111110110

7) 10011101111101011001101111110111

7) 10011101111101011001110111110101

7) 10011101111101011001110111110110

7) 10011101111101011001110111110111

7) 10011101111101101001100111110101

7) 10011101111101101001100111110110

7) 10011101111101101001100111110111

7) 10011101111101101001101111110101

7) 10011101111101101001101111110110

7) 10011101111101101001101111110111

7) 10011101111101101001110111110101

7) 10011101111101101001110111110110

7) 10011101111101101001110111110111

7) 10011101111101111001100111110101

7) 10011101111101111001100111110110

7) 10011101111101111001100111110111

7) 10011101111101111001101111110101

7) 10011101111101111001101111110110

7) 10011101111101111001101111110111

7) 10011101111101111001110111110101

7) 10011101111101111001110111110110

7) 10011101111101111001110111110111

Process 0 finished in time 90.596949 secs.

A total of 81 solutions were found.

]0;ajm94@remotel2: ~/CS374/Homework01[01;32majm94@remotel2[00m:[01;34m~/CS374/Homework01[00m$ exit

Script done on 2023-11-01 00:16:52-04:00 [COMMAND\_EXIT\_CODE="0"]

SProcess 0 is checking the circuit...

Process 3 is checking the circuit...

Process 5 is checking the circuit...

Process 12 is checking the circuit...

Process 1 is checking the circuit...

Process 2 is checking the circuit...

Process 7 is checking the circuit...

Process 9 is checking the circuit...

Process 13 is checking the circuit...

Process 14 is checking the circuit...

Process 4 is checking the circuit...

Process 6 is checking the circuit...

Process 10 is checking the circuit...

Process 11 is checking the circuit...

Process 15 is checking the circuit...

Process 8 is checking the circuit...

Process 16 is checking the circuit...

Process 17 is checking the circuit...

Process 20 is checking the circuit...

Process 18 is checking the circuit...

Process 22 is checking the circuit...

Process 25 is checking the circuit...

Process 30 is checking the circuit...

Process 31 is checking the circuit...

Process 19 is checking the circuit...

Process 21 is checking the circuit...

Process 23 is checking the circuit...

Process 24 is checking the circuit...

Process 26 is checking the circuit...

Process 28 is checking the circuit...

Process 29 is checking the circuit...

Process 27 is checking the circuit...

19) 10011001111101011001100111110101

19) 10011001111101011001100111110110

19) 10011001111101011001100111110111

19) 10011001111101011001101111110101

19) 10011001111101011001101111110110

19) 10011001111101011001101111110111

19) 10011001111101011001110111110101

19) 10011001111101011001110111110110

19) 10011001111101011001110111110111

19) 10011001111101101001100111110101

19) 10011001111101101001100111110110

19) 10011001111101101001100111110111

19) 10011001111101101001101111110101

19) 10011001111101101001101111110110

19) 10011001111101101001101111110111

19) 10011001111101101001110111110101

19) 10011001111101101001110111110110

19) 10011001111101101001110111110111

19) 10011001111101111001100111110101

19) 10011001111101111001100111110110

19) 10011001111101111001100111110111

19) 10011001111101111001101111110101

19) 10011001111101111001101111110110

19) 10011001111101111001101111110111

19) 10011001111101111001110111110101

19) 10011001111101111001110111110110

19) 10011001111101111001110111110111

19) 10011011111101011001100111110101

19) 10011011111101011001100111110110

19) 10011011111101011001100111110111

19) 10011011111101011001101111110101

19) 10011011111101011001101111110110

19) 10011011111101011001101111110111

19) 10011011111101011001110111110101

19) 10011011111101011001110111110110

19) 10011011111101011001110111110111

19) 10011011111101101001100111110101

19) 10011011111101101001100111110110

19) 10011011111101101001100111110111

19) 10011011111101101001101111110101

19) 10011011111101101001101111110110

19) 10011011111101101001101111110111

19) 10011011111101101001110111110101

19) 10011011111101101001110111110110

19) 10011011111101101001110111110111

19) 10011011111101111001100111110101

19) 10011011111101111001100111110110

19) 10011011111101111001100111110111

19) 10011011111101111001101111110101

19) 10011011111101111001101111110110

19) 10011011111101111001101111110111

19) 10011011111101111001110111110101

19) 10011011111101111001110111110110

19) 10011011111101111001110111110111

19) 10011101111101011001100111110101

19) 10011101111101011001100111110110

19) 10011101111101011001100111110111

19) 10011101111101011001101111110101

19) 10011101111101011001101111110110

19) 10011101111101011001101111110111

19) 10011101111101011001110111110101

19) 10011101111101011001110111110110

19) 10011101111101011001110111110111

19) 10011101111101101001100111110101

19) 10011101111101101001100111110110

19) 10011101111101101001100111110111

19) 10011101111101101001101111110101

19) 10011101111101101001101111110110

19) 10011101111101101001101111110111

19) 10011101111101101001110111110101

19) 10011101111101101001110111110110

19) 10011101111101101001110111110111

19) 10011101111101111001100111110101

19) 10011101111101111001100111110110

19) 10011101111101111001100111110111

19) 10011101111101111001101111110101

19) 10011101111101111001101111110110

19) 10011101111101111001101111110111

19) 10011101111101111001110111110101

19) 10011101111101111001110111110110

19) 10011101111101111001110111110111

Process 0 finished in time 9.562237 secs.

A total of 81 solutions were found.

Process 2 is checking the circuit...

Process 6 is checking the circuit...

Process 10 is checking the circuit...

Process 12 is checking the circuit...

Process 0 is checking the circuit...

Process 4 is checking the circuit...

Process 7 is checking the circuit...

Process 3 is checking the circuit...

Process 13 is checking the circuit...

Process 9 is checking the circuit...

Process 11 is checking the circuit...

Process 1 is checking the circuit...

Process 5 is checking the circuit...

Process 31 is checking the circuit...

Process 16 is checking the circuit...

Process 18 is checking the circuit...

Process 8 is checking the circuit...

Process 20 is checking the circuit...

Process 22 is checking the circuit...

Process 30 is checking the circuit...

Process 17 is checking the circuit...

Process 19 is checking the circuit...

Process 21 is checking the circuit...

Process 23 is checking the circuit...

Process 24 is checking the circuit...

Process 26 is checking the circuit...

Process 27 is checking the circuit...

Process 28 is checking the circuit...

Process 29 is checking the circuit...

Process 25 is checking the circuit...

Process 14 is checking the circuit...

Process 15 is checking the circuit...

21) 10011001111101011001100111110101

21) 10011001111101011001101111110101

21) 10011001111101011001110111110101

21) 10011001111101101001100111110101

21) 10011001111101101001101111110101

21) 10011001111101101001110111110101

21) 10011001111101111001100111110101

21) 10011001111101111001101111110101

21) 10011001111101111001110111110101

22) 10011001111101011001100111110110

22) 10011001111101011001101111110110

22) 10011001111101011001110111110110

22) 10011001111101101001100111110110

22) 10011001111101101001101111110110

22) 10011001111101101001110111110110

22) 10011001111101111001100111110110

22) 10011001111101111001101111110110

22) 10011001111101111001110111110110

23) 10011001111101011001100111110111

23) 10011001111101011001101111110111

23) 10011001111101011001110111110111

23) 10011001111101101001100111110111

23) 10011001111101101001101111110111

23) 10011001111101101001110111110111

23) 10011001111101111001100111110111

23) 10011001111101111001101111110111

23) 10011001111101111001110111110111

21) 10011011111101011001100111110101

21) 10011011111101011001101111110101

21) 10011011111101011001110111110101

21) 10011011111101101001100111110101

21) 10011011111101101001101111110101

21) 10011011111101101001110111110101

21) 10011011111101111001100111110101

21) 10011011111101111001101111110101

21) 10011011111101111001110111110101

22) 10011011111101011001100111110110

22) 10011011111101011001101111110110

22) 10011011111101011001110111110110

22) 10011011111101101001100111110110

22) 10011011111101101001101111110110

22) 10011011111101101001110111110110

22) 10011011111101111001100111110110

22) 10011011111101111001101111110110

22) 10011011111101111001110111110110

23) 10011011111101011001100111110111

23) 10011011111101011001101111110111

23) 10011011111101011001110111110111

23) 10011011111101101001100111110111

23) 10011011111101101001101111110111

23) 10011011111101101001110111110111

23) 10011011111101111001100111110111

23) 10011011111101111001101111110111

23) 10011011111101111001110111110111

21) 10011101111101011001100111110101

21) 10011101111101011001101111110101

21) 10011101111101011001110111110101

21) 10011101111101101001100111110101

21) 10011101111101101001101111110101

21) 10011101111101101001110111110101

21) 10011101111101111001100111110101

21) 10011101111101111001101111110101

21) 10011101111101111001110111110101

22) 10011101111101011001100111110110

22) 10011101111101011001101111110110

22) 10011101111101011001110111110110

22) 10011101111101101001100111110110

22) 10011101111101101001101111110110

22) 10011101111101101001110111110110

22) 10011101111101111001100111110110

22) 10011101111101111001101111110110

22) 10011101111101111001110111110110

23) 10011101111101011001100111110111

23) 10011101111101011001101111110111

23) 10011101111101011001110111110111

23) 10011101111101101001100111110111

23) 10011101111101101001101111110111

23) 10011101111101101001110111110111

23) 10011101111101111001100111110111

23) 10011101111101111001101111110111

23) 10011101111101111001110111110111

Process 0 finished in time 10.501128 secs.

A total of 81 solutions were found.