

Computação Paralela - WA1

Mestrado em Engenharia Informática

Afonso Marques

Departamento de Informática
Universidade do Minho
Mirandela, Portugal
pg53601

Renato Gomes

Departamento de Informática
Universidade do Minho
Porto, Portugal
pg54174

Abstract—The program to analyse and optimise is part of a simple molecular dynamics simulation code applied to atoms of argon gas (original code version in Foley-Lab/MolecularDynamics: Simple Molecular Dynamics). The code follows a generic approach to simulate particle movements over time steps, using Newton laws of motion. In this report we shall refer to the program as MD.

The code uses Lennard Jones potential to describe the interactions among two particles (force/potential energy). The Verlet integration method is used to calculate the particles trajectories over time. Boundary conditions (i.e., particles moving outside the simulation space) are managed by elastic walls.

Index Terms—algorithm, performance, single thread, optimization, metrics

I. INTRODUCTION

The main goal of this project was to evaluate the benefits of optimization of code in a single thread, using C++, by making changes to certain algorithms present in the *MD.cpp* file.

We started off by removing complex calculations from the inside of *loops* plus replacing the use of the *pow* function from the *math.h* library with an uglier but better performing alternative. During this process, we took the care not to interfere with the calculations being made as to make sure the final output would not be changed in any way. Furthermore, we performed tests using the *perf* metrics command to see how well our version of the code was performing.

We also deconstructed smaller loops that had an average of three to five iterations, which in turn removed additional instructions and improved the performance, even if so slightly.

Lastly, the obtained results were analysed and compared with the expected results.

II. IMPLEMENTATION

The original version of the MD code consists of four main steps.

- 1) Put all the atoms in simple crystal lattice and give them random velocities that corresponds to the initial temperature we have specified, using the *initialize* function;

- 2) Based on the positions of the atoms, calculate the initial inter molecular forces. The accelerations of each particle will be defined from the forces and their mass, and this will allow to update their positions via Newton's law. The calculus is made in the *computeAccelerations* function.
- 3) Calculate various things like instantaneous mean velocity squared, temperature, pressure, potential and kinetic energy.
- 4) Print the final results to the terminal and additional results to the *cp_average.txt* and *cp_output.txt*. The *cp_output.txt* contains all the values of temperature (in Kelvin), pressure (in Pascal), kinetic, potential and total energy (in natural units n.u). The *cp_average.txt* contains all of the values that were printed on the terminal.

III. ALGORITHMIC COMPLEXITY

- 1) *Potential()*: The potential function involves three nested loops. The outer two loops iterate over all particles in the system, resulting in N^2 iterations. Inside the innermost loop, there are several mathematical operations like addition, subtraction, multiplication, and power calculations. These operations are not nested within the loops and are generally considered to be constant time. So, the time complexity of the *Potential()* function is $O(N^2)$.
- 2) *computeAcceleitaions()*: This function also involves nested loops. The outer loop iterates over all particles, and the two inner loops iterate over the components for each pair of particles. The most computationally intensive part is the calculation of forces (f) and updating the accelerations. Inside the inner loops, there are constant time operations as well as calculations that depend on the number of dimensions (3 in this case). Overall, the time complexity of this function is also $O(N^2)$.

IV. OPTIMIZATIONS AND METRICS

Before looking at the code, the group decided it was best to use the *gprof* tool to access what parts of the code used more time and computation power.

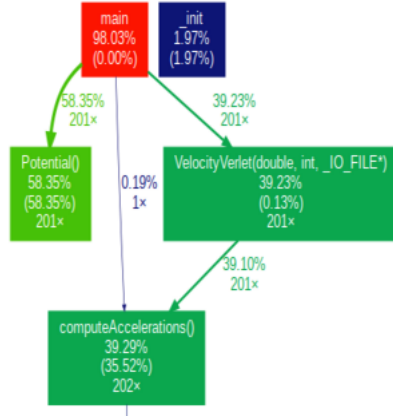


Fig. 1. *gprof* graph from the original code

Looking at the graph it's easy to see that the functions that take most of the time to execute are *Potencial()* and *computeAccelerations()*, making them the best candidates for optimization.

A. Optimizations

1) *Potencial()*:

- We started by removing the *sqr()* function replacing it with a multiplication. The *quot* variable became *sigma/r2*, which is the original *sigma/rnorm* squared.
- Next we removed the *pow()* functions, by replacing them with multiplications. As *quot* is now the original's *quot* squared, *term2* instead of multiplying *quot* 6 times, we can do it only 3 times. And *term1* is *term2* * *term2*.
- In the original code there's an *if (j!=i)* condition to ensure that a particle is not interacting with itself. By removing that condition and starting the loops from *i+1*, we can avoid self-interaction, and that way we have a big improvement in performance.

2) *computeAccelarations()*:

- Similarly as in the *Potencial()* function, we started by trying to optimize the calculations, starting with the removal of the *pow* functions. The introduction of two new variables, *invrSqd* and *invrSqd3*, where *invrSqd* is defined as the reciprocal of the original *rSqd*, and *invrSqd3* is the cube of *invrSqd*. This change simplifies the code by avoiding repetitive division operations when dealing

with the reciprocal. The calculation $2 * pow(rSqd, -7)$ in the original is transformed into $2.0 * invrSqd3$. This change replaces the use of the *pow* function with direct multiplication. The second part of the calculation $- pow(rSqd, -4)$ in the original is transformed into -1.0 , because $invrSqd3 * invrSqd$ is already used to represent the $1/r^4$ term.

- In the original *computeAccelerations()* function, the loops over *k* are nested within the loops over *i* and *j*. This means that for each pair of particles *i* and *j*, there's an inner loop over *k* (*k*=0, *k*=1, *k*=2). In the optimized version, the inner loop over *k* has been eliminated. Instead, each element is computed separately without an explicit loop. This eliminates the need for an inner loop and can lead to slightly improved performance and code readability.

B. Metrics

The code was tested for performance and reliability through 3 independent runs on the university's cluster (3 was the best we could do because otherwise the cluster would crash). To ensure robustness, the results of each run were carefully examined, and an average was computed to assess the code's behavior. This thorough testing approach helped verify the code's stability and performance across multiple executions.

TABLE I
TABLE WITH EXECUTION TIME FOR 3 ITERATIONS OF ORIGINAL VERSION OF MD

Iteration	Texte
1	236,95 sec
2	236,79 sec
3	236,92 sec

TABLE II
TABLE WITH EXECUTION TIME FOR 3 ITERATIONS OF OPTIMIZED VERSION OF MD

Iteration	Texte
1	6.82 sec
2	6.43 sec
3	6.65 sec

TABLE III
TABLE WITH METRICS FOR EACH VERSION OF MD

Version MD.cpp	#I	Average #CC	Average CPI	Average Texte
Original	1.243.580.424.482	780.836.157.356	0,6	236,8770 sec
Optimized	35.346.319.145	21.426.058.674	0,6	6,654 sec

REFERENCES

- [1] <https://iopscience.iop.org/article/10.1088/1742-6596/1491/1/012022/pdf>
- [2] https://pt.wikipedia.org/wiki/Potencial_de_Lennard-Jones
- [3] <https://chem.libretexts.org/Bookshelves>

V. ANNEXES

```

Terminal - pg53601@search7edu:~$testar-cluster
Ficheiro  Editor  Ver  Terminal  Separadores  Ajuda

ENTER THE INITIAL TEMPERATURE OF YOUR GAS IN KELVIN

ENTER THE NUMBER DENSITY IN moles/m^3
FOR REFERENCE, NUMBER DENSITY OF AN IDEAL GAS AT STP IS ABOUT 40 moles/m^3
NUMBER DENSITY OF LIQUID ARGON AT 1 ATM AND 87 K IS ABOUT 35000 moles/m^3
PERCENTAGE OF CALCULATION COMPLETE:
[ 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 ]

TO ANIMATE YOUR SIMULATION, OPEN THE FILE
'cp_traj.xyz' WITH VMD AFTER THE SIMULATION COMPLETES

TO ANALYZE INSTANTANEOUS DATA ABOUT YOUR MOLECULE, OPEN THE FILE
'cp_output.txt' WITH YOUR FAVORITE TEXT EDITOR OR IMPORT THE DATA INTO EXCEL

THE FOLLOWING THERMODYNAMIC AVERAGES WILL BE COMPUTED AND WRITTEN TO THE FILE
'cp_average.txt':

AVERAGE TEMPERATURE (K):      88.70310
AVERAGE PRESSURE (Pa):        23075480.34071
PV/NT (J * mol^-1 K^-1):      7.43265
PERCENT ERROR OF PV/NT AND GAS CONSTANT:  10.60569
THE COMPRESSIBILITY (unitless): 0.89394
TOTAL VOLUME (m^3):           1.02479e-25
NUMBER OF PARTICLES (unitless): 2160
Time taken: 236.92s

Performance counter stats for 'make run' (3 runs):
1243592074035      inst_retired.any          #    0,6 CPI
780854077830
236,935719347 seconds time elapsed
236,931430000 seconds user
0,009764000 seconds sys

pg53601@search7edu testar-cluster14

```

Fig. 2. Example of the original MD.cpp result in the Search cluster

```

Terminal - pg53601@search7edu:~$testar-cluster
Ficheiro  Editor  Ver  Terminal  Separadores  Ajuda

ENTER THE INITIAL TEMPERATURE OF YOUR GAS IN KELVIN

ENTER THE NUMBER DENSITY IN moles/m^3
FOR REFERENCE, NUMBER DENSITY OF AN IDEAL GAS AT STP IS ABOUT 40 moles/m^3
NUMBER DENSITY OF LIQUID ARGON AT 1 ATM AND 87 K IS ABOUT 35000 moles/m^3
PERCENTAGE OF CALCULATION COMPLETE:
[ 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 ]

TO ANIMATE YOUR SIMULATION, OPEN THE FILE
'cp_traj.xyz' WITH VMD AFTER THE SIMULATION COMPLETES

TO ANALYZE INSTANTANEOUS DATA ABOUT YOUR MOLECULE, OPEN THE FILE
'cp_output.txt' WITH YOUR FAVORITE TEXT EDITOR OR IMPORT THE DATA INTO EXCEL

THE FOLLOWING THERMODYNAMIC AVERAGES WILL BE COMPUTED AND WRITTEN TO THE FILE
'cp_average.txt':

AVERAGE TEMPERATURE (K):      88.70310
AVERAGE PRESSURE (Pa):        23075480.34071
PV/NT (J * mol^-1 K^-1):      7.43265
PERCENT ERROR OF PV/NT AND GAS CONSTANT:  10.60569
THE COMPRESSIBILITY (unitless): 0.89394
TOTAL VOLUME (m^3):           1.02479e-25
NUMBER OF PARTICLES (unitless): 2160
Time taken: 6.07s

Performance counter stats for 'make run':
3534683973      inst_retired.any          #    0,6 CPI
21161279891
0,004168456 seconds time elapsed
0,682198800 seconds user
0,002370800 seconds sys

pg53601@search7edu testar-cluster16

```

Fig. 3. Example of the optimized MD.cpp result in the Search cluster

```

Terminal - pg53601@search7edu:~$testar-cluster
Ficheiro  Editor  Ver  Terminal  Separadores  Ajuda

YOU WILL NOW ENTER A FEW SIMULATION PARAMETERS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ENTER THE INITIAL TEMPERATURE OF YOUR GAS IN KELVIN

ENTER THE NUMBER DENSITY IN moles/m^3
FOR REFERENCE, NUMBER DENSITY OF AN IDEAL GAS AT STP IS ABOUT 40 moles/m^3
NUMBER DENSITY OF LIQUID ARGON AT 1 ATM AND 87 K IS ABOUT 35000 moles/m^3
PERCENTAGE OF CALCULATION COMPLETE:
[ 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 ]

TO ANIMATE YOUR SIMULATION, OPEN THE FILE
'cp_traj.xyz' WITH VMD AFTER THE SIMULATION COMPLETES

TO ANALYZE INSTANTANEOUS DATA ABOUT YOUR MOLECULE, OPEN THE FILE
'cp_output.txt' WITH YOUR FAVORITE TEXT EDITOR OR IMPORT THE DATA INTO EXCEL

THE FOLLOWING THERMODYNAMIC AVERAGES WILL BE COMPUTED AND WRITTEN TO THE FILE
'cp_average.txt':

AVERAGE TEMPERATURE (K):      88.70310
AVERAGE PRESSURE (Pa):        23075480.34071
PV/NT (J * mol^-1 K^-1):      7.43265
PERCENT ERROR OF PV/NT AND GAS CONSTANT:  10.60569
THE COMPRESSIBILITY (unitless): 0.89394
TOTAL VOLUME (m^3):           1.02479e-25
NUMBER OF PARTICLES (unitless): 2160
Time taken: 236.79s

Performance counter stats for 'make run' (3 runs):
1243588424482      inst_retired.any          #    0,6 CPI
780830157256
236,8770 +- 0,6770 seconds time elapsed (+ - 0,03%)

pg53601@search7edu testar-cluster14

```

Fig. 4. Example of the result of 3 iterations of the original MD.cpp in the Search cluster

```

Terminal - pg53601@search7edu:~$testar-cluster
Ficheiro  Editor  Ver  Terminal  Separadores  Ajuda

YOU WILL NOW ENTER A FEW SIMULATION PARAMETERS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ENTER THE INITIAL TEMPERATURE OF YOUR GAS IN KELVIN

ENTER THE NUMBER DENSITY IN moles/m^3
FOR REFERENCE, NUMBER DENSITY OF AN IDEAL GAS AT STP IS ABOUT 40 moles/m^3
NUMBER DENSITY OF LIQUID ARGON AT 1 ATM AND 87 K IS ABOUT 35000 moles/m^3
PERCENTAGE OF CALCULATION COMPLETE:
[ 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 ]

TO ANIMATE YOUR SIMULATION, OPEN THE FILE
'cp_traj.xyz' WITH VMD AFTER THE SIMULATION COMPLETES

TO ANALYZE INSTANTANEOUS DATA ABOUT YOUR MOLECULE, OPEN THE FILE
'cp_output.txt' WITH YOUR FAVORITE TEXT EDITOR OR IMPORT THE DATA INTO EXCEL

THE FOLLOWING THERMODYNAMIC AVERAGES WILL BE COMPUTED AND WRITTEN TO THE FILE
'cp_average.txt':

AVERAGE TEMPERATURE (K):      88.70310
AVERAGE PRESSURE (Pa):        23075480.34071
PV/NT (J * mol^-1 K^-1):      7.43265
PERCENT ERROR OF PV/NT AND GAS CONSTANT:  10.60569
THE COMPRESSIBILITY (unitless): 0.89394
TOTAL VOLUME (m^3):           1.02479e-25
NUMBER OF PARTICLES (unitless): 2160
Time taken: 6.65s

Performance counter stats for 'make run' (3 runs):
35346319145      inst_retired.any          #    0,6 CPI
21420950674
6,654 +- 0,112 seconds time elapsed (+ - 1,68%)

pg53601@search7edu testar-cluster15

```

Fig. 5. Example of the result of 3 iterations of the optimized MD.cpp in the Search cluster

```

Terminal - pg53601@search7edu:~$testar-cluster
Ficheiro  Editor  Ver  Terminal  Separadores  Ajuda

Time (s)      T(t) (K)      P(t) (Pa)      Kinetic En. (n.u.)      Potential En. (n.u.)      Total En. (n.u.)
0.0000e+00    89.82306818    0.00000000    2276.12602954    -21120.90135872    -18848.85331893
5.0000e-15    89.43086643    0.00000000    2287.16925478    -21153.06528702    -18865.89683233
1.0000e-14    88.77596388    0.00000000    2252.25261323    -21123.22989494    -18878.97708615
5.0000e-14    87.86214191    0.00000000    2211.43167653    -21061.55650991    -18850.12513338
2.0000e-14    86.09426692    0.00000000    2284.78152338    -21028.20514180    -18823.49625803
5.0000e-14    85.27547768    0.00000000    2172.43704078    -20903.52750404    -18731.13503314
3.0000e-14    83.11339318    0.00000000    2134.54486899    -20887.76883666    -18753.22394457
5.0000e-14    81.71815331    0.00000000    2001.15431918    -20803.34646851    -18799.87248292
4.0000e-14    89.46125653    0.00000000    2843.86481894    -20784.74769972    -18661.46388878
4.0000e-14    87.28021724    0.00000000    2696.11274127    -20588.01859666    -18688.69236889
5.0000e-14    84.77287539    0.00000000    1832.85680142    -20484.42555900    -18651.46895738
5.0000e-14    82.10974968    0.00000000    1872.12907580    -20382.87688837    -18609.55818287
6.0000e-14    79.30497268    0.00000000    1888.93737931    -20234.91661524    -18426.13264183
6.0000e-14    78.40831815    0.00000000    1742.23481935    -20102.73432440    -18368.47964254
7.0000e-14    73.45226064    0.00000000    1674.87687468    -19907.87788271    -18233.00308803
7.0000e-14    78.48674794    0.00000000    1687.21352959    -19832.42898609    -18225.20675759
8.0000e-14    67.53931228    0.00000000    1548.33586634    -19658.51671178    -18150.18065544
8.0000e-14    64.76457608    0.00000000    1475.37099963    -19568.45562174    -18093.08471211
8.0000e-14    61.80952322    0.00000000    1412.46661093    -19444.51807120    -18032.05092927
9.0000e-14    59.45718184    0.00000000    1355.71981888    -19328.89182913    -17975.17212133
2.0000e-13    57.15127358    0.00000000    1293.13959138    -19223.63181472    -17929.46655585
1.0000e-13    55.10987848    0.00000000    1256.59086053    -19130.44558126    -17873.84970873
1.1000e-13    53.06420885    0.00000000    1218.79182385    -18960.77432435    -17833.98258338
1.1000e-13    51.15860653    0.00000000    1184.22530888    -18808.51888155    -17783.32508807
1.2000e-13    58.83812528    0.00000000    1159.19382482    -18935.53248251    -17776.33857799
1.2500e-13    58.67364513    0.00000000    1141.78177931    -18906.68470334    -17758.92529644
1.3000e-13    49.37614818    0.00000000    1131.79958848    -18880.79994810    -17749.00044803
1.3500e-13    49.11343864    0.00000000    1118.88745057    -18875.24697115    -17746.24592557
1.4000e-13    49.68246884    0.00000000    1132.84388447    -18883.02411888    -17758.18025441
1.4500e-13    50.11272698    0.00000000    1142.79349971    -18902.38189182    -17768.19274681
1.5000e-13    50.78694778    0.00000000    1158.62684452    -18933.57029332    -17775.54424888
1.5000e-13    51.07421517    0.00000000    1177.61694648    -18970.36589824    -17795.40188886
1.5000e-13    52.85739097    0.00000000    1201.56529493    -19029.71493504    -17819.21882595
1.5500e-13    53.80974886    0.00000000    1228.11358424    -19074.03497071    -17845.92139547
1.7000e-13    52.65844452    0.00000000    1256.33599254    -19111.74512166    -17874.91315572
1.7500e-13    56.45120757    0.00000000    1287.18277572    -19192.34228833    -17895.15951261
1.8000e-13    57.83268480    0.00000000    1318.23453025    -19258.49542354    -17930.27270208
1.8500e-13    59.18451884    0.00000000    1349.43456118    -19316.97863535    -17967.54487525
1.9000e-13    60.34243772    0.00000000    1388.20991813    -19378.72227142    -17996.44231811
1.9500e-13    61.85147952    0.00000000    1418.31558918    -19438.83151997    -18028.51593888
2.0000e-13    62.11718837    0.00000000    1439.12538682    -19486.58821436    -18057.46281785
2.0500e-13    64.31909323    0.00000000    1466.56124537    -19551.41163815    -18084.83064777
2.1000e-13    65.44864973    0.00000000    1492.33895734    -19602.91292380    -18118.59593886
2.1500e-13    66.50054859    0.00000000    1516.32788863    -19650.98578354    -18134.56299331
2.2000e-13    67.47775411    0.00000000    1538.48998085    -19695.23333333    -18156.74345248
2.2500e-13    68.50250578    0.00000000    1558.82645371    -19735.94463805    -18177.86234844
2.3000e-13    69.18256663    0.00000000    1577.47644798    -19773.17557082    -18195.69892284

cp_output.txt: 2020, 240506

```

Fig. 6. cp_output.txt file obtained with optimized MD