# Εργασία 2

**Ονοματεπώνυμο: Νικόλαος Θεοδώρου**
**Αριθμός Ταυτότητας: 1030496**

---

**Ερώτημα:**

**1) Συναρτήσεις για `n_body_omp_static`. (10% Κώδικας + 5% Σχόλια) (`Courier New 10`):**

```
void n_body_omp_static(int threads)
{
SimulationTime++;
omp_set_num_threads(threads);
char* execution_type = "static";

computeAccelerations(execution_type);
computePositions(execution_type);
computeVelocities(execution_type);
resolveCollisions(execution_type);
}

void computeAccelerations(char* exec_type)
{
int i,j;
if (strcmp(exec_type, "static") == 0){
#pragma omp parallel private(i, j) default(none) shared(accelerations, positions, masses,
GravConstant, threads, bodies)
{
#pragma omp for schedule(static,bodies/threads)
for (i = 0; i < bodies; i++)
{
accelerations[i].x = 0;
accelerations[i].y = 0;
accelerations[i].z = 0;
// #pragma omp parallel for schedule(static)
for (j = 0; j < bodies; j++)
{
if (i != j)
{
// accelerations[i] =
addVectors(accelerations[i],scaleVector(GravConstant*masses[j]/pow(mod(subtractVectors(positio
ns[i],positions[j])),3),subtractVectors(positions[j],positions[i])));
vector sij = {positions[i].x - positions[j].x, positions[i].y - positions[j].y, positions[i].z -
positions[j].z};
vector sji = {positions[j].x - positions[i].x, positions[j].y - positions[i].y, positions[j].z -
positions[i].z};
double mod = sqrt(sij.x * sij.x + sij.y * sij.y + sij.z * sij.z);
double mod3 = mod * mod * mod;
double s = GravConstant * masses[j] / mod3;
```

```
vector S = {s * sji.x, s * sji.y, s * sji.z};
accelerations[i].x += S.x;
accelerations[i].y += S.y;
accelerations[i].z += S.z;
}
}
}
}
}
```

**Η n_body_omp_static δέχεται ως argument των αριθμό των threads που δόθηκαν από το command line και θέτει τη ρύθμηση του openmp για το με πόσα threads να τρέξει. Στη συνέχει παραλληλοποιήσα τη Compute Accelerations, αφού είναι η πιο χρονοβόρα μέθοδος του προγράμματος. Η Compute Accelerations δέχεται ως argument τον τύπο παράλληλης εκτέλεσης σε αυτή τη περίπτωση είναι static, άρα θα εκτελέσουμε το branch του static scheduling. Το κάθε thread εκτελεί bodies/threads επαναλήψεις για να υπολογήσει τις επιταχύνσεις των σωματηδίων.**

**2) Συναρτήσεις για n_body_omp_dynamic. (10% Κώδικας + 5% Σχόλια) (Courier New 10):**

```
void n_body_omp_dynamic(int threads)
{
SimulationTime++;
omp_set_num_threads(threads);
char* execution_type = "dynamic";

computeAccelerations(execution_type);
computePositions(execution_type);
computeVelocities(execution_type);
resolveCollisions(execution_type);
}
void computeAccelerations(char* exec_type)
{
int i,j;
if (strcmp(exec_type, "dynamic") == 0) {
#pragma omp parallel private(i, j) default(none) shared(accelerations, positions, masses,
GravConstant, bodies)
{
#pragma omp for schedule(dynamic,
bodies/omp_get_num_threads()*log(omp_get_num_threads()))
for (i = 0; i < bodies; i++)
{
accelerations[i].x = 0;
accelerations[i].y = 0;
accelerations[i].z = 0;
```

```
// #pragma omp parallel for schedule(static)
for (j = 0; j < bodies; j++)
{
if (i != j)
{
// accelerations[i] =
addVectors(accelerations[i],scaleVector(GravConstant*masses[j]/pow(mod(subtractVectors(positions[i],positions[j])),3),subtractVectors(positions[j],positions[i])));
vector sij = {positions[i].x - positions[j].x, positions[i].y - positions[j].y, positions[i].z -
positions[j].z};
vector sji = {positions[j].x - positions[i].x, positions[j].y - positions[i].y, positions[j].z -
positions[i].z};
double mod = sqrt(sij.x * sij.x + sij.y * sij.y + sij.z * sij.z);
double mod3 = mod * mod * mod;
double s = GravConstant * masses[j] / mod3;
vector S = {s * sji.x, s * sji.y, s * sji.z};
accelerations[i].x += S.x;
accelerations[i].y += S.y;
accelerations[i].z += S.z;
}
}
}
}
}
```

**H n_body_omp_dynamic δέχεται ως argument τον αριθμό των threads που δόθηκαν από το command line και θέτει τη ρύθμηση του openmp για το πόσα threads να τρέξει. Στη συνέχεια παραλληλοποιούμε την Compute Accelerations, αυτή τη φορά με dynamic scheduling. H Compute Accelerations δέχεται ως argument τον τύπο παράλληλης εκτέλεσης, σε αυτήν τη περίπτωση είναι dynamic. Εκτελούμε το branch του dynamic scheduling, το οποίο χρησιμοποιεί μια δυναμική ουρά για να διανείμει τις επαναλήψεις στα threads. Αντί για να διανέμονται ισόποσα στα threads, κάθε thread αναλαμβάνει ένα μικρότερο αριθμό επαναλήψεων. Αυτό μπορεί να είναι χρήσιμο όταν υπάρχει μεγάλος αριθμός επαναλήψεων και το σύστημα μπορεί να διαχειριστεί καλύτερα την κατανομή τους στα threads. H κατανομή γίνεται με τύπο bodies/threads*log(threads) που είναι rule of thumb για dynamic scheduling workload distribution.**

**3) Συναρτήσεις για n_body_omp_guided. (10% Κώδικας + 5% Σχόλια) (Courier New 10):**

```
void n_body_omp_guided(int threads)
{
SimulationTime++;
omp_set_num_threads(threads);
char* execution_type = "guided";

computeAccelerations(execution_type);
computePositions(execution_type);
computeVelocities(execution_type);
resolveCollisions(execution_type);
}
```

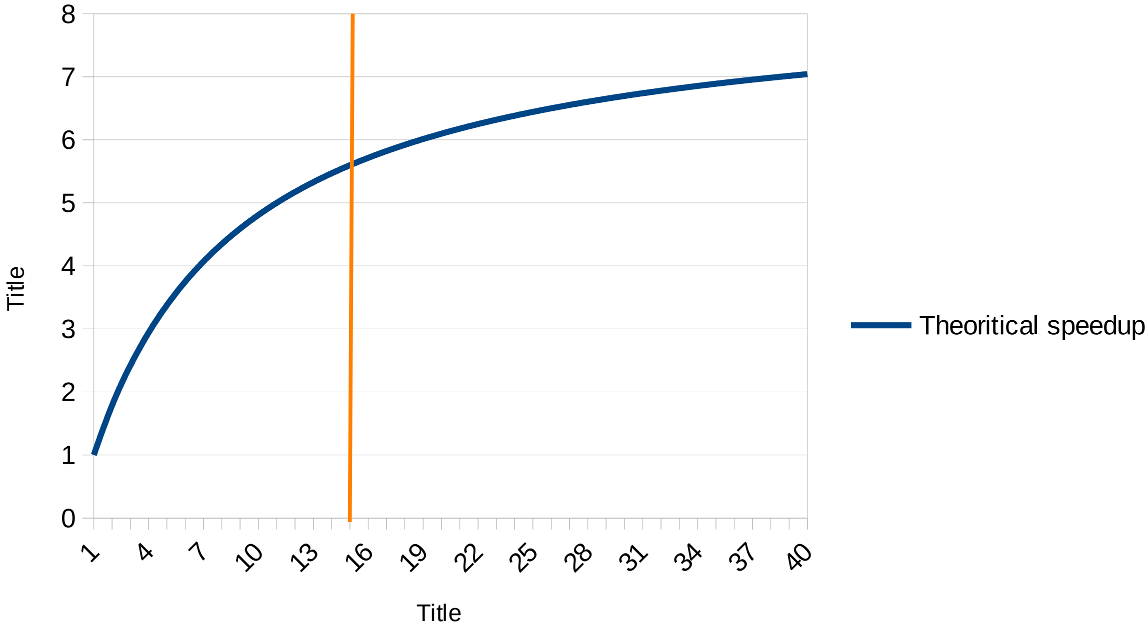
```
void computeAccelerations(char* exec_type)
{
int i,j;
if (strcmp(exec_type, "guided") == 0){
#pragma omp parallel private(i, j) default(none) shared(accelerations, positions, masses,
GravConstant, bodies)
{
#pragma omp for schedule(guided)
for (i = 0; i < bodies; i++)
{
accelerations[i].x = 0;
accelerations[i].y = 0;
accelerations[i].z = 0;
// #pragma omp parallel for schedule(static)
for (j = 0; j < bodies; j++)
{
if (i != j)
{
// accelerations[i] =
addVectors(accelerations[i],scaleVector(GravConstant*masses[j]/pow(mod(subtractVectors(positio
ns[i],positions[j])),3),subtractVectors(positions[j],positions[i])));
vector sij = {positions[i].x - positions[j].x, positions[i].y - positions[j].y, positions[i].z -
positions[j].z};
vector sji = {positions[j].x - positions[i].x, positions[j].y - positions[i].y, positions[j].z -
positions[i].z};
double mod = sqrt(sij.x * sij.x + sij.y * sij.y + sij.z * sij.z);
double mod3 = mod * mod * mod;
double s = GravConstant * masses[j] / mod3;
vector S = {s * sji.x, s * sji.y, s * sji.z};
accelerations[i].x += S.x;
accelerations[i].y += S.y;
accelerations[i].z += S.z;
}
}
}
}

}
```

**Η n_body_omp_guided δέχεται ως argument τον αριθμό των threads που δόθηκαν από το command line και θέτει τη ρύθμηση του openmp για το πόσα threads να τρέξει. Στη συνέχεια παραλληλοποιούμε την Compute Accelerations, αυτή τη φορά με guided scheduling. Η Compute Accelerations δέχεται ως argument τον τύπο παράλληλης εκτέλεσης, σε αυτήν τη περίπτωση είναι guided. Εκτελούμε το branch του guided scheduling, το οποίο χρησιμοποιεί μια δυναμική ουρά για να διανείμει τις επαναλήψεις στα threads. Αντί για να διανέμονται ισόποσα στα threads, κάθε thread αναλαμβάνει ένα μικρότερο αριθμό επαναλήψεων. Επίσης όσο προχωρά η εκτέλεση το guided scheduling εξυπηρετά μικρότερα chunk sizes.**

**4) Ο ιδανικός αριθμός threads. (5%)**

**Σύμφωνα με την προηγούμενη εργασία η συνάρτηση Compute Accelerations απασχολούσε το πρόγραμμα για 88% του χρόνου. Επίσης, σύμφωνα με το νόμο του Amdahl η θεωρητική μέγιστη επιτάχυνση που μπορεί να επιτευχθεί με την παραλληλοποίηση της συνάρτησης εξαρτάται από το ποσοστό της συνάρτησης που μπορεί να παραλληλοποιηθεί και τον αριθμό των επεξεργαστών που χρησιμοποιούνται. Για το συγκεκριμένο ποσοστό παραλληλοποίησης, αν χρησιμοποιηθούν 40 επεξεργαστές, η μέγιστη θεωρητική επιτάχυνση που μπορεί να επιτευχθεί βάσει του νόμου του Amdahl είναι 1 / ((1 - 0.88) + (0.88 / 40)) = 7.04.**




**Μπορούμε να πούμε ότι θεωρητικά μετά τα 14-15 threads η γραμμή του γραφήματος να αρχίζει να ευθυγραμμίζεται.**

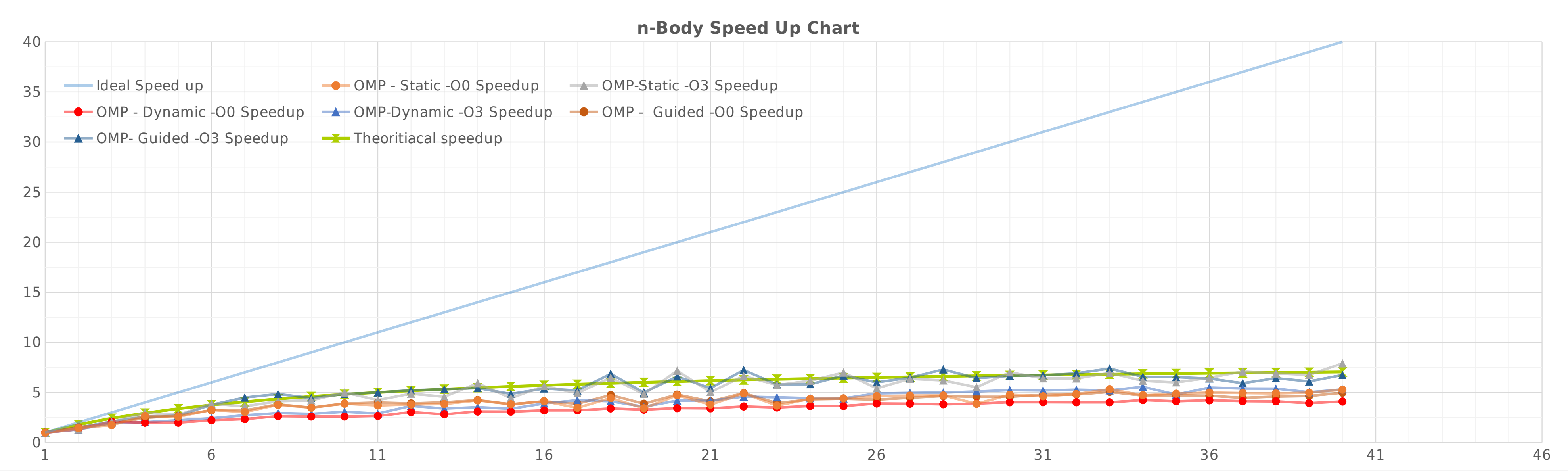
**Μετά από πιάσιμο μετρήσεων ο ιδανικός αριθμός threads που επέλεξα για μέγιστο speedup και λιγότερους πόρους για κάθε περίπτωση έχει ως εξής.**

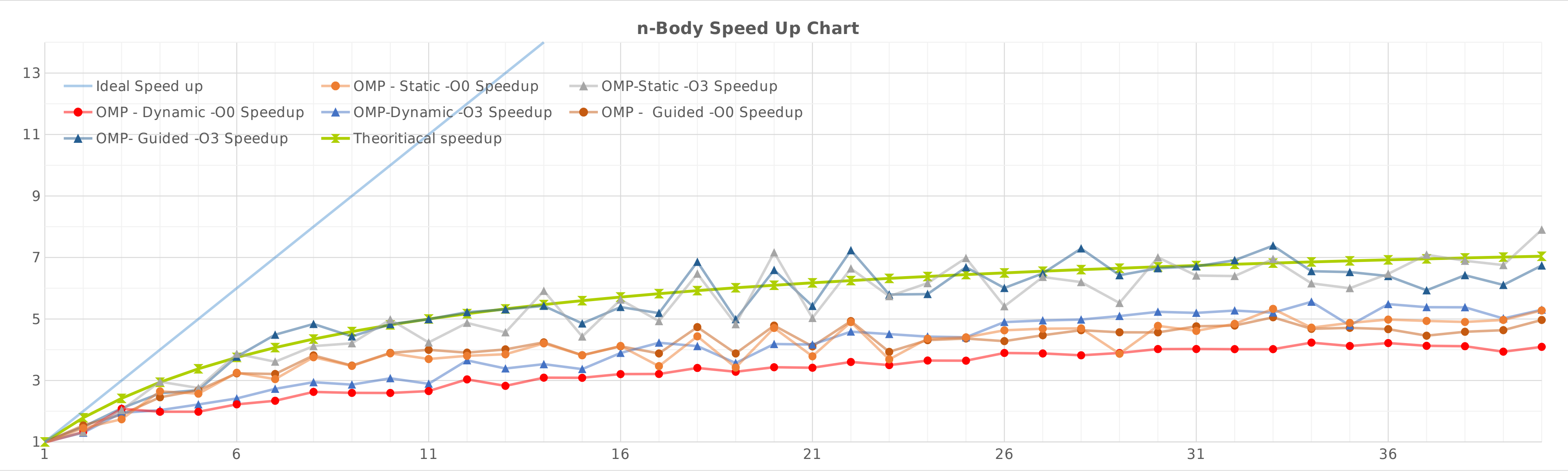
| Schedule | Threads | Speedup |
|---|---|---|
| Static O0 | 14 | 4.21 |
| Static O3 | 14 | 5.91 |
| Dynamic O0 | 14 | 3.09 |
| Dynamic O3 | 12 | 3.65 |
| Guided O0 | 14 | 4.24 |
| Guided O3 | 13 | 5.31 |

## 5) Αναμενόμενη επιτάχυνση (Speedup) των πιο πάνω 3 μεθόδων, η καταμετρημένη, efficiency (10%)

| Number of Threads | Ideal Speed up | Theoritiacal speedup | QMP - Static -O0 Speedup | QMP - Static -O0 Efficienc | QMP-Static -O3 Speedu | QMP - Static -O3 Efficienc | QMP - Dynamic -O0 Speed | QMP - Dynamic-O0 Efficier | QMP-Dynamic -O3 Speedu | QMP - Dynamic-O3 Efficier | QMP - Guided -O0 Speed | QMP - guided-O3 Efficienc | QMP- Guided -O3 Speedu | QMP - guided-O3 Efficiency |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0.981062491750765 | 0.981062491750765 | 0.998253747800382 | 0.998253747800382 | 0.980612066972696 | 0.980612066972696 | 0.997989199978261 | 0.997989199978261 | 0.994591163247737 | 0.994591163247737 | 0.998542720821297 | 0.998542720821297 |
| 2 | 2 | 1.78571428571429 | 1.45044035527704 | 0.725220177638521 | 1.31943045543343 | 0.659715227716714 | 1.32319338189284 | 0.661596690946422 | 1.29789489158851 | 0.648947445794255 | 1.53582139756726 | 0.767910698783629 | 1.49178425668378 | 0.745892128341892 |
| 3 | 3 | 2.41935483870968 | 1.73774420291383 | 0.579248067637942 | 2.03653092885571 | 0.678843642951903 | 2.07896192773165 | 0.692987309243883 | 1.94792220345874 | 0.649307401152913 | 1.8901041788241 | 0.6300347262747 | 2.09706809117803 | 0.699022697059343 |
| 4 | 4 | 2.94117647058824 | 2.64686637056476 | 0.661716592641191 | 2.96062312616007 | 0.740155781540016 | 1.98132358911896 | 0.49533089727974 | 2.03553520468213 | 0.508883801170534 | 2.45302326404369 | 0.613255816010923 | 2.58304753574708 | 0.645761883936769 |
| 5 | 5 | 3.37837837837838 | 2.56932978400782 | 0.513865956801564 | 2.75335350278361 | 0.550670700556723 | 1.98287317673186 | 0.396574635346372 | 2.22032578554467 | 0.444065157108934 | 2.68252060776829 | 0.536504121553657 | 2.68529177955084 | 0.537058355910168 |
| 6 | 6 | 3.75 | 3.25059070160537 | 0.541765116934228 | 3.86693753096421 | 0.644489588494036 | 2.22295105863489 | 0.370491843105815 | 2.41505463762225 | 0.402509106270375 | 3.23244912601994 | 0.538741521003324 | 3.77326065759341 | 0.628876776265568 |
| 7 | 7 | 4.06976744186047 | 3.04385850815006 | 0.434836929735722 | 3.61022702251352 | 0.515746717501931 | 2.3393774494752 | 0.334196778496457 | 2.72748140046718 | 0.38964020006674 | 3.21119897747002 | 0.458742711067145 | 4.48685086151979 | 0.640978694502827 |
| 8 | 8 | 4.34782608695652 | 3.75504468708371 | 0.469380585885464 | 4.11534706636873 | 0.514418383296091 | 2.62853009437141 | 0.328566261796426 | 2.94314942631898 | 0.367893678289873 | 3.81568450669455 | 0.476960563336818 | 4.83647448783651 | 0.604559310979563 |
| 9 | 9 | 4.59183673469388 | 3.47335707363403 | 0.385928563737114 | 4.20816150007208 | 0.467573500008009 | 2.59570786229983 | 0.288411984699981 | 2.86519295514299 | 0.318354772793666 | 3.484210434125 | 0.387134492680556 | 4.43181499059227 | 0.492423887843585 |
| 10 | 10 | 4.80769230769231 | 3.88186405839342 | 0.388186405839342 | 4.98159276810834 | 0.498159276810834 | 2.59321473320072 | 0.259321473320072 | 3.0700431873006 | 0.30700431873006 | 3.89671283526104 | 0.389671283526103 | 4.83045578459037 | 0.483045578459037 |
| 11 | 11 | 5 | 3.69717434864071 | 0.336106758967337 | 4.23684731576846 | 0.385167937797132 | 2.6553279808886 | 0.241393452808055 | 2.89225574184355 | 0.262932340167596 | 3.9950051520285 | 0.363182286548046 | 4.99300978629918 | 0.453909980572653 |
| 12 | 12 | 5.17241379310345 | 3.80051570713683 | 0.316709642261402 | 4.87361885984341 | 0.406134904986951 | 3.03471515135435 | 0.252892929279529 | 3.654526402325 | 0.304543866860417 | 3.90513467593144 | 0.325427889660953 | 5.22113847699908 | 0.435094873083256 |
| 13 | 13 | 5.32786885245902 | 3.85107817562432 | 0.296236782740332 | 4.56424559798848 | 0.351095815229883 | 2.82469829752501 | 0.217284484425001 | 3.38883870872495 | 0.26067990067115 | 4.01212670778475 | 0.308625131368058 | 5.31268015771822 | 0.408667704439863 |
| 14 | 14 | 5.46875 | 4.20630863548267 | 0.300450616820191 | 5.91448474882764 | 0.422463196344832 | 3.09086014077432 | 0.220775724341023 | 3.5305229438549 | 0.25218021027535 | 4.23938878670083 | 0.302813484764345 | 5.42626246562919 | 0.387590176116371 |
| 15 | 15 | 5.59701492537313 | 3.81996325320595 | 0.254664216880397 | 4.42797956991604 | 0.295198637994403 | 3.08537381074014 | 0.205691587382676 | 3.37082626627926 | 0.224721751085284 | 3.8222978234207 | 0.254819854894713 | 4.85453692524205 | 0.323635795016137 |
| 16 | 16 | 5.71428571428571 | 4.12239782085095 | 0.257649863803185 | 5.62848436618808 | 0.351780272886755 | 3.20658727647195 | 0.200411704779497 | 3.8937957569192 | 0.24336223480745 | 4.11145780161475 | 0.256966112600922 | 5.38839095177123 | 0.336774434485702 |
| 17 | 17 | 5.82191780821918 | 3.46646582632968 | 0.203909754489981 | 4.92878635711936 | 0.289928609242315 | 3.21134436351155 | 0.188902609618326 | 4.22731751172557 | 0.248665735983857 | 3.8781604888771 | 0.228127087581006 | 5.19229484009274 | 0.305429108240749 |
| 18 | 18 | 5.92105263157895 | 4.43446729689413 | 0.246359294271896 | 6.47170391634201 | 0.359539106463445 | 3.40712711414099 | 0.1892848396745 | 4.11555500500386 | 0.228641944722437 | 4.73505430563922 | 0.263058572535512 | 6.85749048438325 | 0.380971693576847 |
| 19 | 19 | 6.0126582278481 | 3.41884812841418 | 0.179939375179694 | 4.8300353596846 | 0.254212387351821 | 3.28367389637156 | 0.172824941914293 | 3.56906602136068 | 0.187845580071615 | 3.8782887040497 | 0.204120458107879 | 4.98945298799073 | 0.262602788841617 |
| 20 | 20 | 6.09756097560976 | 4.70812699902441 | 0.235406349951221 | 7.16143432893815 | 0.358071716446908 | 3.4295635121429 | 0.171478175607145 | 4.18403801923498 | 0.209201900961749 | 4.78462894810325 | 0.239231447405162 | 6.59262044260634 | 0.329631022130317 |
| 21 | 21 | 6.1764705882353 | 3.7857160588834 | 0.180272193280162 | 5.03237160227722 | 0.239636742965582 | 3.4135460700012 | 0.1625498128572 | 4.16736810486253 | 0.198446100231549 | 4.10828265703069 | 0.195632507477652 | 5.42212592431142 | 0.258196472586258 |
| 22 | 22 | 6.25 | 4.89919176721611 | 0.222690534873459 | 6.64015862747264 | 0.301825392157847 | 3.60115284119152 | 0.163688765508706 | 4.58688514167117 | 0.208494779166871 | 4.93062369964904 | 0.224119259074956 | 7.23404499146239 | 0.328820226884654 |
| 23 | 23 | 6.31868131868132 | 3.68583623769172 | 0.160253749464858 | 5.74191186335842 | 0.249648341885148 | 3.49898777916378 | 0.152129903441903 | 4.50788209879849 | 0.195994873860804 | 3.93020640670893 | 0.170878539422127 | 5.79389564333467 | 0.251908506231942 |
| 24 | 24 | 6.38297872340426 | 4.36017629407888 | 0.181674012253287 | 6.16558189052613 | 0.256899245438589 | 3.64818280625398 | 0.152007616927249 | 4.42633784965878 | 0.184430743735782 | 4.31123640134737 | 0.17963485005614 | 5.80911812985507 | 0.242046588743961 |
| 25 | 25 | 6.44329896907217 | 4.40198228719302 | 0.176079291487721 | 6.98230366246023 | 0.279292146498409 | 3.6429074781127 | 0.145716299124508 | 4.40409153184769 | 0.176163661273907 | 4.36517269625029 | 0.174606907850012 | 6.68293509215834 | 0.267317403686333 |
| 26 | 26 | 6.5 | 4.6307261585925 | 0.178104852253558 | 5.41218910724601 | 0.208161119509462 | 3.8950872095127 | 0.149811046519719 | 4.9006478559423 | 0.188486455997781 | 4.27953251602837 | 0.16459740446263 | 6.00787805316199 | 0.231072232813923 |
| 27 | 27 | 6.55339805825243 | 4.68282197599734 | 0.173437850962864 | 6.36483485459188 | 0.235734624244144 | 3.87787450096205 | 0.143624981517113 | 4.95255355622766 | 0.183427909489913 | 4.46996695155582 | 0.165554331539104 | 6.4808481312699 | 0.240031412269255 |
| 28 | 28 | 6.60377358490566 | 4.69423593999381 | 0.167651283571208 | 6.19867679730941 | 0.221381314189622 | 3.81875002086749 | 0.136383929316696 | 4.98119967015581 | 0.17789998821985 | 4.63482633837575 | 0.165529512084848 | 7.29196598622757 | 0.260427356650985 |
| 29 | 29 | 6.65137614678899 | 3.86884816145147 | 0.13340855729143 | 5.51995216770161 | 0.190343178196607 | 3.89239961118544 | 0.134220676247774 | 5.09580620194932 | 0.175717455239632 | 4.56708752087009 | 0.157485776581727 | 6.4225177286925 | 0.221466128575604 |
| 30 | 30 | 6.69642857142857 | 4.77617736686993 | 0.159205912228998 | 7.00171368675581 | 0.233390456225194 | 4.02187371843983 | 0.134062457281328 | 5.23234188225469 | 0.174411396075156 | 4.56416694974478 | 0.152138898324826 | 6.64859477670596 | 0.221619825890199 |
| 31 | 31 | 6.73913043478261 | 4.61279260985252 | 0.148799761608146 | 6.41185164118009 | 0.206833923909035 | 4.02274720660874 | 0.129766038922862 | 5.1988161111826 | 0.167703745522019 | 4.76087899874812 | 0.153576741895101 | 6.71047298868521 | 0.216466870602749 |
| 32 | 32 | 6.77966101694915 | 4.84151575953064 | 0.151297367485333 | 6.3960402811218 | 0.199876258785056 | 4.01939147730867 | 0.125605983665896 | 5.27354559789476 | 0.164798299934211 | 4.78622821677691 | 0.149569631774278 | 6.91099189840158 | 0.215968496825049 |
| 33 | 33 | 6.81818181818182 | 5.32861542188657 | 0.161473194602623 | 6.94943289875295 | 0.210588875719786 | 4.01870476486624 | 0.121778932268674 | 5.2064593086853 | 0.157771494202585 | 5.05479061565692 | 0.153175473201725 | 7.3849519539413 | 0.223786422846706 |
| 34 | 34 | 6.85483870967742 | 4.71802212262526 | 0.138765356547802 | 6.15876190592339 | 0.18114005605657 | 4.23204756173933 | 0.12447198710998 | 5.55800743782461 | 0.163470806994841 | 4.67906473933518 | 0.137619551156917 | 6.55427065893133 | 0.192772666439157 |
| 35 | 35 | 6.88976377952756 | 4.87249246361207 | 0.139214070388916 | 6.00260073077207 | 0.171502878022059 | 4.12168660316717 | 0.117762474376205 | 4.78944463307164 | 0.136841275230618 | 4.71066964099549 | 0.1345905611713 | 6.5252222451933 | 0.186434921291237 |
| 36 | 36 | 6.92307692307692 | 4.98014202385352 | 0.138337278440376 | 6.46960630203709 | 0.179711286167697 | 4.2166015205572 | 0.117127820015478 | 5.48243233267182 | 0.152289787018662 | 4.66846500360467 | 0.129679583433463 | 6.39476060168999 | 0.177632238935833 |
| 37 | 37 | 6.95488721804511 | 4.93616433596991 | 0.133409846918106 | 7.08922171576011 | 0.191600586912435 | 4.12605806333206 | 0.111515082792758 | 5.38412248931971 | 0.145516824035668 | 4.45576376173595 | 0.120426047614485 | 5.93211080948057 | 0.160327319175151 |
| 38 | 38 | 6.98529411764706 | 4.90306401311771 | 0.129028000345203 | 6.89213736780403 | 0.181372035994843 | 4.11375730360518 | 0.108256771147505 | 5.38001553279732 | 0.141579356126245 | 4.58351617304953 | 0.120618846659198 | 6.42359609445357 | 0.16904200248562 |
| 39 | 39 | 7.01438848920863 | 4.9691856183622 | 0.127415015855441 | 6.75173405922274 | 0.173121386133916 | 3.93670636645014 | 0.100941188883337 | 5.01593677337203 | 0.128613763419796 | 4.63631292021336 | 0.118879818467009 | 6.10632748536597 | 0.156572499624769 |
| 40 | 40 | 7.04225352112676 | 5.27584906574227 | 0.131896226643557 | 7.90225613324612 | 0.197556403331153 | 4.09364652637315 | 0.102341163159329 | 5.30110819404365 | 0.132527704851091 | 4.96822238089162 | 0.12420555952229 | 6.73754654577329 | 0.168438663644332 |

**5) Γραφική Παράσταση και Σχολιασμός (~200 λέξεις) για τις λύσεις πιο πάνω. (15%-25%)**



**Από τις πιο πάνω μετρησεις μπορούμε να συμπεράνουμε ότι οι εκτελέσεις μας δεν είναι καθόλου κοντά στο ιδεατό speedup. Αυτό γιατί δεν είναι 100% του προγράμματος μας παραλληλισμένο. Από την άλλη παρατηρούμε πως εφαρμόζεται με σχετική ακρίβια ο νόμος του Αμνταλ, αφού τα πραγματικά μας speedup είναι κοντά στο θεωριτικό speedup που προτείνει ο νόμος. Φαίνεται να ξεπερνάμε το θεωριτικό speedup σε κάποιες περιπτώσεις, πιθανώς από βελτιστοποιήσης που κάνει το υλικό η το λειτουργικό σύστημα. Στο συγκεκριμένο πείραμα παρατηρούμε πως το dynamic scheduling μας δίνει τα χειρότερα speedups σε σύγκριση με τα άλλα 2 schedulings. Το στατικό scheduling φαίνεται να μας δίνει το πιο μεγάλο speedup. Αυτό γίνεται επειδή στο συγκεκριμένο πείραμα εκμεταλευόμαστε την χωρική τοπικότητα του πινάκων positions και accelerations, σε αντίθεση με το dynamic schedule που φαίνεται να χάνει αυτό το πλεονέκτημα εκμετάλλευσης της τοπικής χωρικότητας. Το guided scheduling, είναι ενδιάμεσα του static και του dynamic, αφού ξεκινά να επεξεργάζεται μεγάλα chunks και σταδιακά τα μικραίνει. Όσο αφορά τις βελτιστοποιήσης μεταξύ O0 και O3, μπορούμε να πούμε πως οι εκτελέσεις με O3 μας δίνουν παρόμοιο speedup, με κάποιες φορές το O3 να είναι δίνει καλύτερο speedup. Πιθανότατα επειδή γίνονται καλύτερες βελτιστοποιήσεις από τον compiler στο παράλληλο κομμάτι του προγράμματος,.**

---

**6) Bonus: `resolveCollisions`: Κώδικας και επεξήγηση/ορθότητα λύσης (~200 λέξεις). (10% +10%)**

**ΚΩΔΙΚΑΣ**

```
void resolveCollisions(char* exec_type)
{
int i, j;
double dx, dy, dz, md;
// int velocity_swaps[bodies-1][bodies];

# pragma omp parallel private(i,j,dx,dy,dz,md)
shared(bodies,masses,positions,velocity_swaps,velocities,threads, exec_type) default(none)
{
if (strcmp(exec_type, "static") == 0){
# pragma omp for schedule(static, bodies/threads)
for(i=0;i<bodies-1;i++){
int step = 0;
for(j=i+1;j<bodies;j++){
md = masses[i]+masses[j];
dx = fabs(positions[i].x-positions[j].x);
dy = fabs(positions[i].y-positions[j].y);
dz = fabs(positions[i].z-positions[j].z);
if(dx<md && dy<md && dz<md){
//Swap Velocities
// Store the swap
velocity_swaps[i][step++] = j;
}
}
velocity_swaps[i][step] = -1;
}
}
if(strcmp(exec_type, "dynamic")==0){
```

```
# pragma omp for schedule(dynamic)
for(i=0;i<bodies-1;i++){
int step = 0;
for(j=i+1;j<bodies;j++){
md = masses[i]+masses[j];
dx = fabs(positions[i].x-positions[j].x);
dy = fabs(positions[i].y-positions[j].y);
dz = fabs(positions[i].z-positions[j].z);
if(dx<md && dy<md && dz<md){
//Swap Velocities
// Store the swap
velocity_swaps[i][step++] = j;
}
}
velocity_swaps[i][step] = -1;
}
}
if(strcmp(exec_type, "guided")==0){
# pragma omp for schedule(guided)
for(i=0;i<bodies-1;i++){
int step = 0;
for(j=i+1;j<bodies;j++){
md = masses[i]+masses[j];
dx = fabs(positions[i].x-positions[j].x);
dy = fabs(positions[i].y-positions[j].y);
dz = fabs(positions[i].z-positions[j].z);
if(dx<md && dy<md && dz<md){
//Swap Velocities
// Store the swap
velocity_swaps[i][step++] = j;
}
}
velocity_swaps[i][step] = -1;
}
}

# pragma omp barrier
# pragma omp single
{
for (i=0; i<bodies-1; i++){
j = 0;
while(velocity_swaps[i][j] != -1){
vector temp = velocities[i];
velocities[i] = velocities[velocity_swaps[i][j]];
velocities[velocity_swaps[i][j]] = temp;
j++;
}}}}}
```

**<u>ΕΠΕΞΗΓΗΣΗ ΚΑΙ ΟΡΘΟΤΗΤΑ ΛΥΣΗΣ</u>**

**Αυτή η υλοποίηση χρησιμοποιεί το OpenMP για να παραλληλίσει τον κώδικα ανάλυσης σύγκρουσης.**

**Η υλοποίηση ξεκινά με μια δήλωση pragma που ορίζει μια παράλληλη περιοχή, η οποία είναι ένα μπλοκ κώδικα που μπορεί να εκτελεστεί παράλληλα από πολλαπλά νήματα. Οι όροι private και shared χρησιμοποιούνται για τον καθορισμό των μεταβλητών που είναι ιδιωτικές σε κάθε νήμα και εκείνων που μοιράζονται μεταξύ των νημάτων, αντίστοιχα.**

**Η παράλληλη περιοχή περιέχει τρεις προτάσεις if, καθεμία από τις οποίες αντιστοιχεί σε διαφορετική στρατηγική προγραμματισμού: στατική, δυναμική ή καθοδηγούμενη. Η ρήτρα χρονοδιαγράμματος καθορίζει τη στρατηγική προγραμματισμού που θα χρησιμοποιηθεί και το σχετικό μέγεθος κομματιού.**

**Για κάθε i, η υλοποίηση χρησιμοποιεί έναν ένθετο βρόχο για επανάληψη σε όλα τα j > i. Για κάθε j, υπολογίζει την απόσταση μεταξύ του i-ου και του j-ου σώματος και καθορίζει εάν συμβαίνει σύγκρουση. Εάν συμβεί σύγκρουση, αποθηκεύει το δείκτη j στον πίνακα velocity_swaps, ο οποίος καταγράφει τις εναλλαγές που θα εκτελεστούν αργότερα.**

**Αφού όλα τα νήματα ολοκληρώσουν την εργασία τους, η υλοποίηση χρησιμοποιεί ένα barrier για να διασφαλίσει ότι όλα τα νήματα έχουν τελειώσει πριν συνεχιστεί. Το single pragma διασφαλίζει ότι το μπλοκ κώδικα εκτελείται από ένα μόνο νήμα, το οποίο επαναλαμβάνεται πάνω από τον πίνακα velocity_swaps και εκτελεί τις ανταλλαγές.**

**Η ορθότητα της παράλληλης υλοποίησης μπορεί να αποδειχθεί παρατηρώντας ότι έχει την ίδια συμπεριφορά με τη σειριακή υλοποίηση. Εφόσον κάθε νήμα εκτελεί τον ίδιο υπολογισμό με τη σειριακή υλοποίηση, τα αποτελέσματα που λαμβάνονται είναι ίδια με τη σειριακή υλοποίηση, σύμφωνα με τον πινακα velocity swaps. Επίσης η χρήση του barrier pragma διασφαλίζει ότι όλα τα νήματα έχουν ολοκληρώσει την εργασία τους πριν εκτελεστούν οι ανταλλαγές, διασφαλίζοντας ότι οι ταχύτητες ενημερώνονται σωστά.**