ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Παράλληλα Συστήματα

Εργασία 2016-2017
Conway’s Game of Life

Αργυρός Ιωάννης Α.Μ. : 1115201200009
Γιαννούδης Αστέριος Α.Μ. : 1115201200025

Περιεχόμενα

1. Εισαγωγή

Στην εργασία αυτή έχουν υλοποιηθεί τρία προγράμματα :
1 πρόγραμμα MPI κώδικα,
1 υβριδικό πρόγραμμα OPENMP-MPI κώδικα και
1 πρόγραμμα CUDA κώδικα.
Και τα τρία προγράμματα υλοποιούν το παιχνίδι game of life το οποίο περιγράφεται στην εκφώνηση της άσκησης.

Πληροφορίες για το παραδοτέο :

Πληροφορίες για τη μεταγλώττιση/εκτέλεση :

- **MPI:**
  mpicc -o mpi_gameoflife mpi_gameoflife.c -lm
  mpiexec -f machines -n <n> ./mpi_gameoflife <Rows> <Cols> <Gens> <termCheckPerGens>
- **OPENMP-MPI:**
  mpicc -openmp -o hybrid_gameoflife hybrid_gameoflife.c -lm
  mpiexec -f machines -n <n> ./hybrid_gameoflife <Rows> <Cols> <Gens> <termCheckPerGens>
- **CUDA:**
  nvcc -o cuda_gameoflife cuda_gameoflife.cu
  [nvprof] ./cuda_gameoflife <Rows> <Cols> <Gens> <termCheckPerGens>

  - Gens : το πλήθος των γενεών που θα τρέξει το πρόγραμμα
  - termCheckPerGens : 0 για εκτέλεση χωρις έλεγχο τερματισμού ή θετικό για έλεγχο τερματισμού ανά τόσες γενιές
  - nvprof πρόγραμμα για μετρήσεις χρόνων του cuda εκτελέσιμου
  - τα ορίσματα πρέπει να δοθούν με την παραπάνω σειρά

Παρατηρήσεις

Και στις τρεις υλοποιήσεις τα προγράμματα αρχικοποιούν το grid του παιχνιδιού τυχαία με πιθανότητα 1/4 για τοποθέτηση ζωντανών οργανισμών και δεν έχει υλοποιηθεί η υποστήριξη εισαγωγής grid από αρχείο.

Κάθε πείραμα έγινε με δεδομένο ότι οι διαστάσεις του πλέγματος διαιρούνται ακριβώς από το πλήθος των διεργασιών. Ενδεικτικά, επιλέξαμε το 420x420 σαν ελάχιστο μέγεθος grid και διπλασιάζοντάς σε κάθε επόμενη μέτρηση διενεργήσαμε τα πειράματα.

2. Σχεδιασμός Διαμοιρασμού Δεδομένων MPI

Το αρχικό πρόγραμμα διαμοιράζει τα δεδομένα του πλέγματος σε ζώνες. Για να επιτευχθεί καλύτερη παραλληλία, έγινε διαχωρισμός του grid σε blocks με βάση των αριθμό των n διεργασιών. Κάθε διεργασία χρησιμοποιεί τη δομή Block για να αποθηκεύσει τις απαραίτητες πληροφορίες για το μπλοκ δεδομένων που διαχειρίζεται. Σε κάθε μία αντιστοιχίζεται ένας υποπίνακας του αρχικού πλέγματος με δύο επιπλέον γραμμές και στήλες. Η επιπλέον αυτή περίμετρος των υποπινάκων των διεργασιών θα χρησιμοποιηθεί για την αποθήκευση δεδομένων από την επικοινωνία της εκάστοτε διεργασίας με τις οκτώ γειτονικές της, ενώ ο υπόλοιπος πίνακας θα περιέχει τους οργανισμούς που έχει αναλάβει να ενημερώσει.

Η συνάρτηση divide_into_blocks αναλαμβάνει τον υπολογισμό του μεγέθους των μπλοκ. Κάθε μπλοκ περιέχει N*M/procs οργανισμούς, οπότε προσπαθούμε να κατανήμουμε τα τετράγωνα του πλέγματος σε μπλοκ ίσου μεγέθους. Εάν ο συνδιασμός πλήθους διεργασιών και μεγέθους πίνακα δεν μας το επιτρέπει, τότε εμφανίζεται σχετικό μήνυμα και το πρόγραμμα τερματίζει.

Για την επικοινωνία των διεργασιών χρησιμοποιήθηκε καρτεσιανή τοπολογία, έτσι ώστε κάθε μ να γνωρίζει και τους 8 της γείτονες (top, bot, left, right, top_left, top_right, bot_left, bot_right).

Για την εύρεση των γειτόνων στον οριζόντιο και στον κατακόρυφο άξονα του καρτεσιανού επιπέδου χρησιμοποιήθηκε η συνάρτηση MPI_Cart_shift.

Για την εύρεση των διαγώνιων γειτόνων χρησιμοποιήθηκε η συνάρτηση MyMPI_Cart_shift.

3. Σχεδιασμός και Υλοποιήση MPI Κώδικα

Για την επικοινωνία μεταξύ των διεργασιών, χρησιμοποιήσαμε τις συναρτήσεις
MPI_Isend() και MPI_Irecv(). Αυτές οι συναρτήσεις επιτρέπουν non-blocking επικοινωνία μεταξύ των διεργασιών.

Τα δεδομένα μεταφέρονται με χρήση MPI_Datatypes. Τα MPI_Datatypes που χρησιμοποιήθηκαν είναι τα:

- ✔ **MPI_Type_vector** : Στήλες των blocks
- ✔ **MPI_Type_contiguous** : Γραμμές των blocks
- ✗ **MPI_Type_create_subarray** : Το sub-grid μέσα στην εξωτερική περίμετρο. Θα ήταν ιδιάιτερα χρήσιμο εάν η κάθε διεργασία έστελνε πίσω στην master process το block της για ανασύνταξη του τελικού ανανεωμένου πλέγματος μέσω της MPI_Gather. Κάτι τέτοιο δεν ζητείται, οπότε μπήκε σε σχόλιο.

Λόγω της non-blocking επικοινωνίας, δίνεται η δυνατότητα οι εσωτερικοί υπολογισμοί των blocks να ξεκινήσουν χωρίς να έχει ολοκληρωθεί ακόμα η ανταλλαγή δεδομένων με τις υπόλοιπες. Στη συνέχεια, το πρόγραμμα περιμένει για την πρώτη άφιξη δεδομένων από κάποια γειτονική διεργασία και ανανεώνει το αντίστοιχο τμήμα του πίνακα και ξαναμπαίνει σε κατάσταση αναμονής για τα επόμενα δεδομένα. Η διαδικασία αυτή προσομοιώνεται από τη συνάρτηση MyMPI_WaitanyAndUpdate που χρησιμοποιεί την MPI_Waitany.

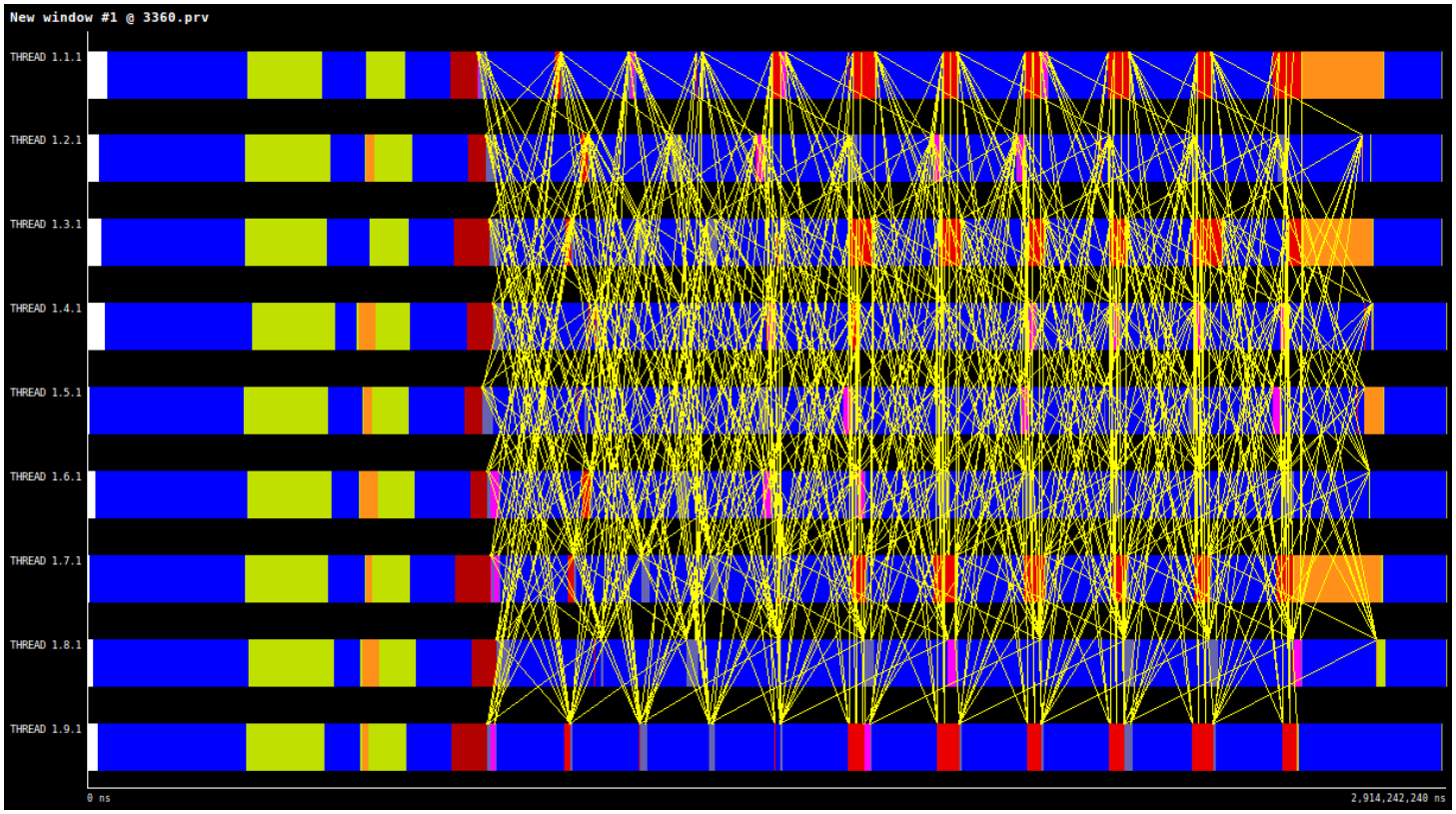
Έπειτα, εάν έχει δοθεί θετικό όρισμα <termCheckPerGens>, τότε πραγματοποιούνται οι έλεγχοι τερματισμού για το ανά τόσες γενιές. Και οι δύο έλεγχοι υλοποιούνται σε κάθε συμμετέχουσα διεργασία για το κομμάτι του πίνακα που τους αναλογεί. Τα αποτελέσματα συλλέγονται στη διεργασία 0 με δύο κλήσεις στην MPI_Allreduce.

Οπτικοποίηση με Paraver

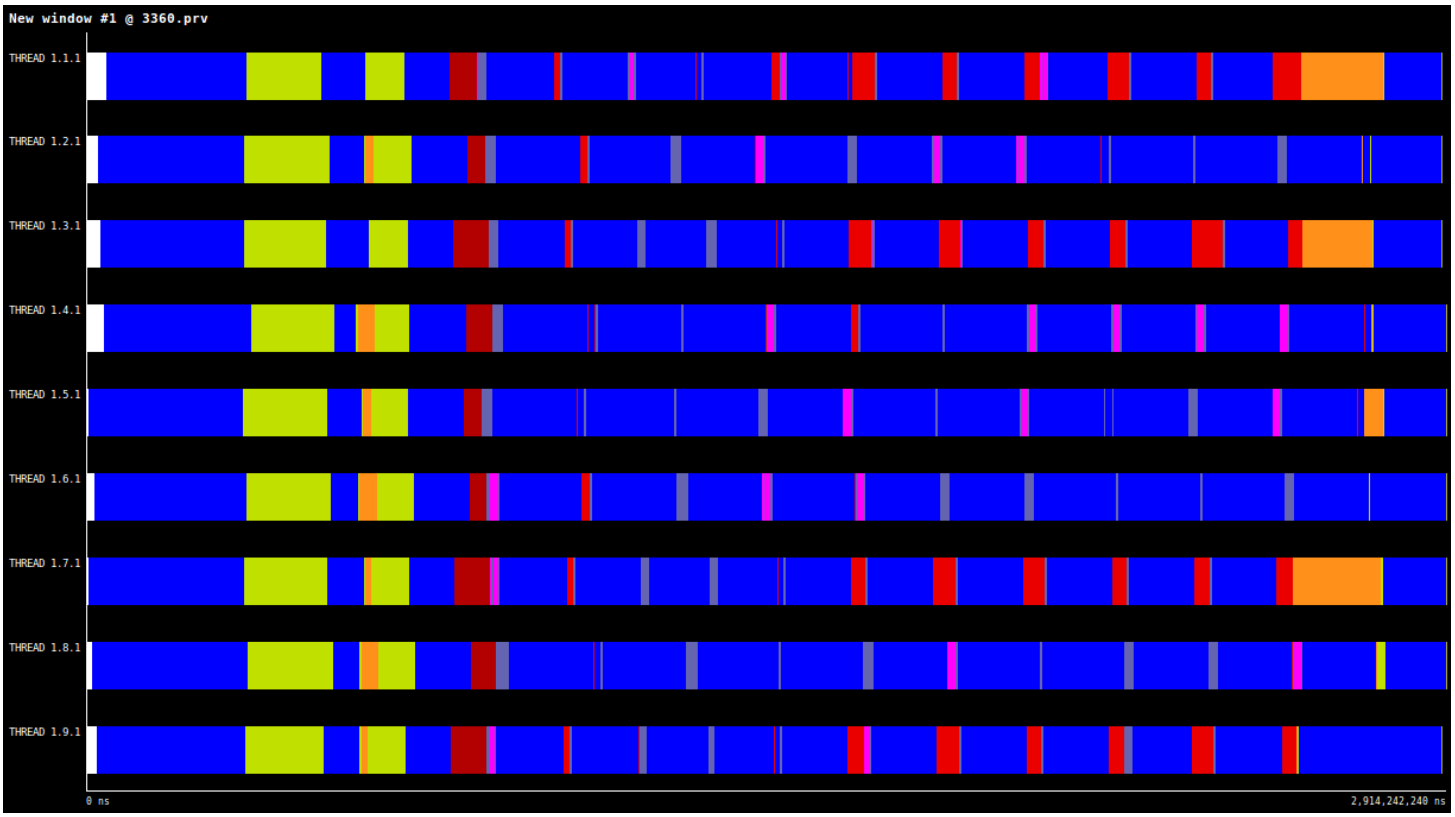
Καθώς ήταν αδύνατο να παρατηρήσουμε τη συμπεριφορά των παράλληλων μηχανών, λόγω των προβλημάτων των μηχανημάτων της σχολής, παραθέτουμε τον τοπικό διαχωρισμό των εργασιών σε τοπικό επίπεδο.

Πιο συγκεκριμένα, έχουμε στιγμιότυπα από την εκτέλεση του MPI προγράμματος με διαστάσεις πλέγματος 3360χ3360 και εκτέλεση μόνο 10 επαναλήψεων, καθώς μας ενδιαφέρει απλώς η επικοινωνία των διεργασιών.

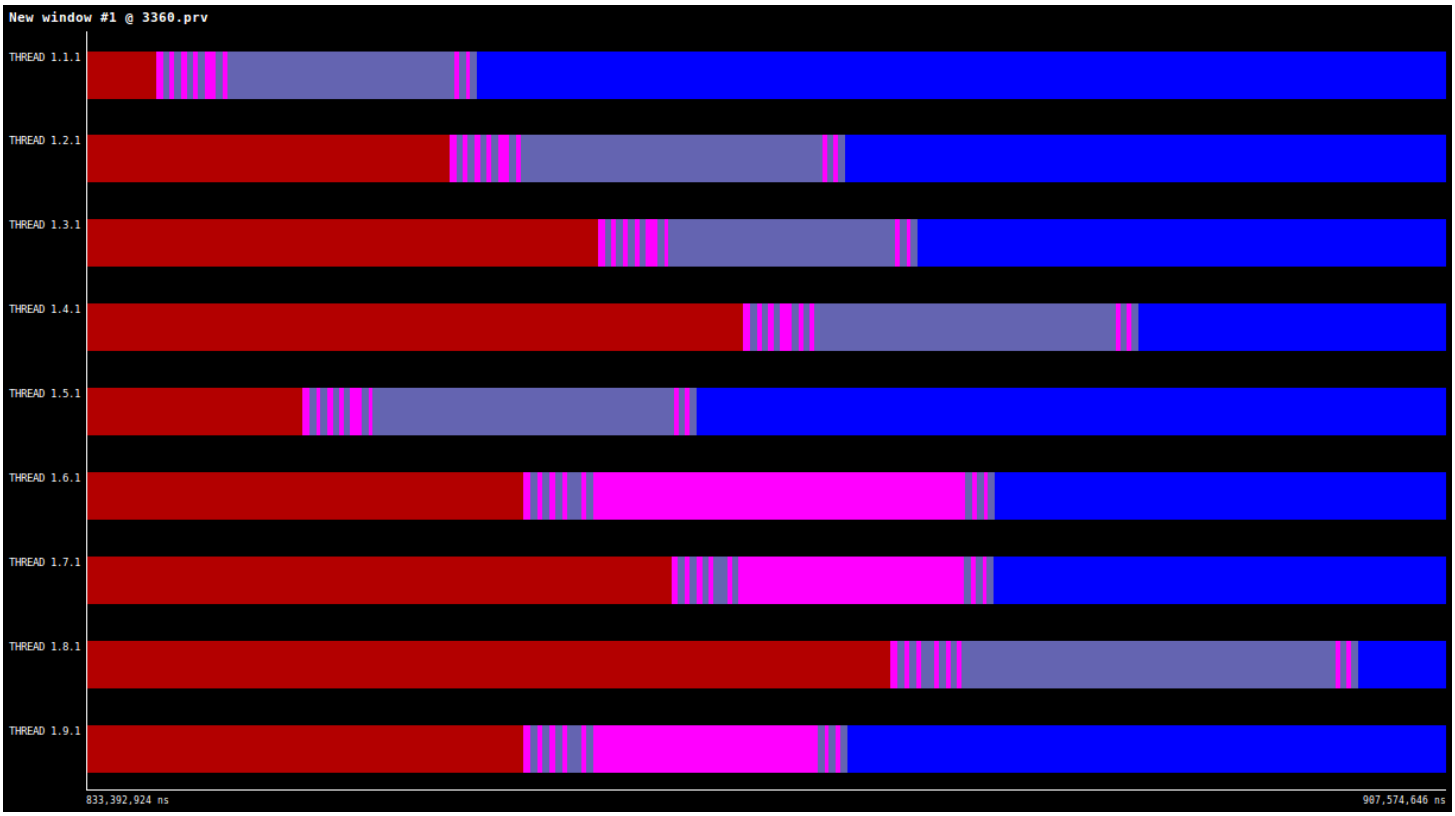
Συνολική εικόνα με communication lines :



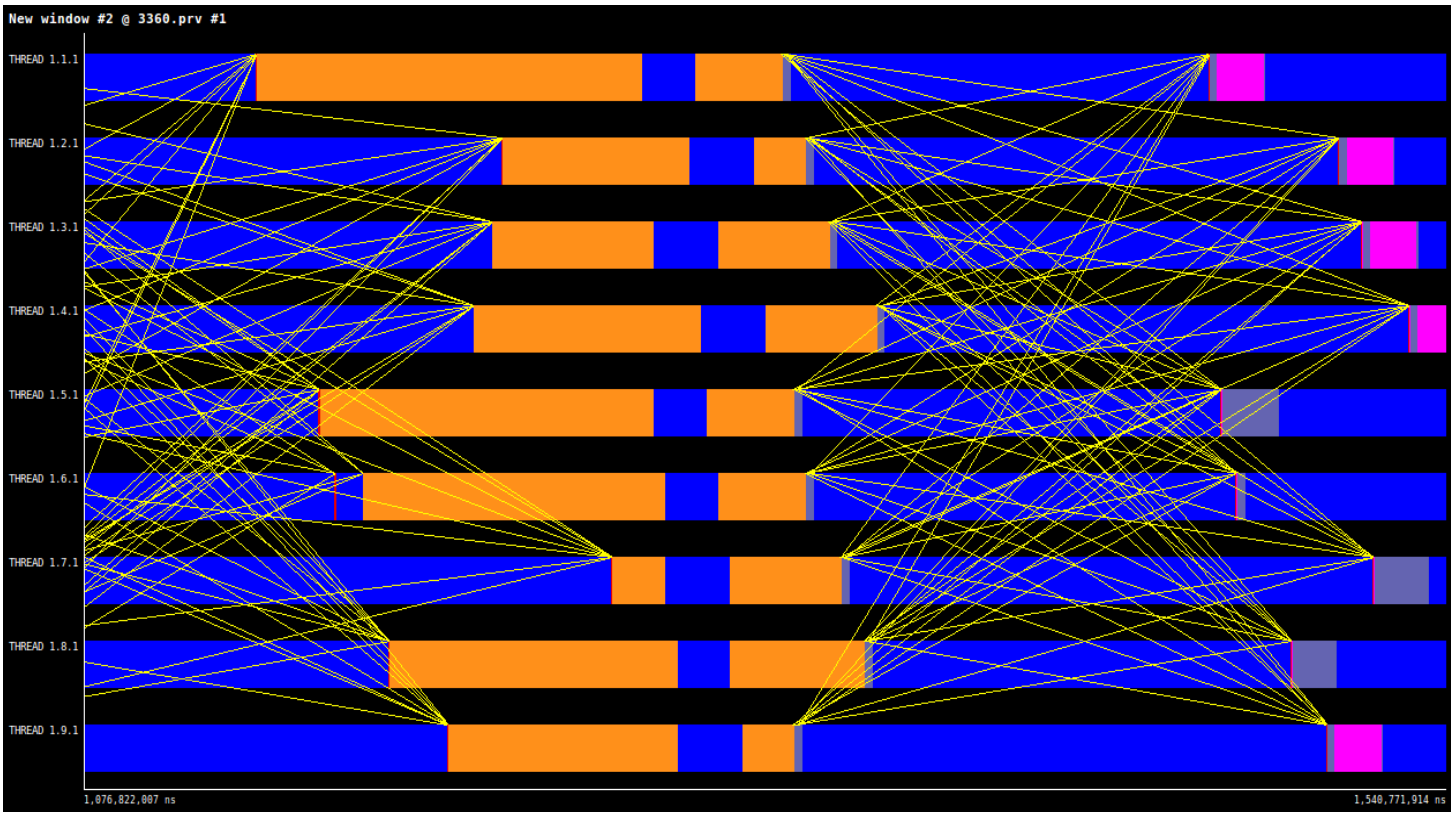
Συνολική εικόνα χωρίς communication lines :



Αυξάνοντας το zoom μπορούμε να παρατηρήσουμε καλύτερα την αλληλουχία των Isend και Irecv μεταξύ των διεργασιών και των γειτόνων τους.



Εκτέλεση της Allreduce για τον έλεγχο τερματισμού:



4. Σχεδιασμός και Υλοποιήση OPENMP-MPI

Στον πηγαίο κώδικα του MPI, προστέθηκε η γραμμή:

" #pragma omp parallel for shared(t1, t2) schedule(static) collapse(3)"

πριν το εξωτερικό loop της update(), στο σημείο όπου διατρέχονται οι εσωτερικοί πίνακες των διεργασιών και γίνονται οι υπολογισμοί.

5. Σχεδιασμός και Υλοποιήση CUDA Κώδικα

Για το παράλληλο πρόγραμμα CUDA εργαστήκαμε με την λογική πως κάθε gpu thread θα αναλάβει ένα στοιχείο του πίνακα του πλέγματος, θα το αρχικοποιήσει και έπειτα θα αποφασίσει για την κατάστασή του στην επόμενη γενιά.

Συνεπώς, δημιουργήσαμε δύο __global__ συναρτήσεις για να επιτύχουμε το παρπάνω.

(1) Στην initdat κάθε thread αρχικοποιεί τον οργανισμό για τον οποίο είναι υπεύθυνο με τυχαιότητα 1/4 να ξεκίνησει ως ζωντανός. (χρήση βιβλιοθήκης curand για την προσομοίωση της τυχαιότητας)

(2) Στην kernel_update κάθε thread αναλαμβάνει να βρεί τους γείτονες του οργανισμού του να αποφασίσει για την κατάστασή του στην επόμενη γενιά.

Τον έλεγχο τερματισμού τον αναλαμβάνει η cpu και δεν παραλληλήσαμε το κομμάτι αυτό.

6. Μετρήσεις

Αρχικά παραθέτουμε τους πίνακες με τους χρόνους εκτέλεσεις των προγραμμάτων MPI με και χωρίς έλεγχο τερματισμού των 250 επαναλήψεων και του OPENMP-MPI χωρίς έλεγχο τερματισμού των 250 επαναλήψεων.

MPI_250_NOTERM

Size/Procs	1	2	4	9	16	25	36	49
420x420	4.08721	0.94047	0.56464	0.82369	0.70258	2.60404	3.27992	3.65598
840x840	7.48987	3.76366	1.98430	0.93040	0.92038	2.10281	3.36340	3.50799
1680x1680	30.08594	21.41694	7.71857	4.68508	3.93231	4.12543	3.82751	4.02552
3360x3360	120.16845	88.82500	42.82410	16.97611	9.23330	10.80059	16.94423	17.76004
6720x6720	505.28015	402.1264	123.1387	58.71132	42.68929	51.66379	57.70617	61.33037

MPI_250_TERM

Size/Procs	1	2	4	9	16	25	36	49
420x420	4.52114	2.68885	1.92924	1.22116	1.51602	3.74921	4.33727	4.73663
840x840	15.31283	11.54648	6.05147	2.64582	2.16118	3.85608	4.19708	5.10078
1680x1680	69.14159	40.56487	24.71911	11.68484	6.76807	7.95612	5.87082	4.88400
3360x3360	122.18842	101.0186	53.67673	30.78542	17.12583	14.60646	17.36807	11.42273
6720x6720	580.37726	482.14517	261.0404	106.1606	60.65335	63.78206	66.44414	69.57568

OPENMPI_250_NOTERM

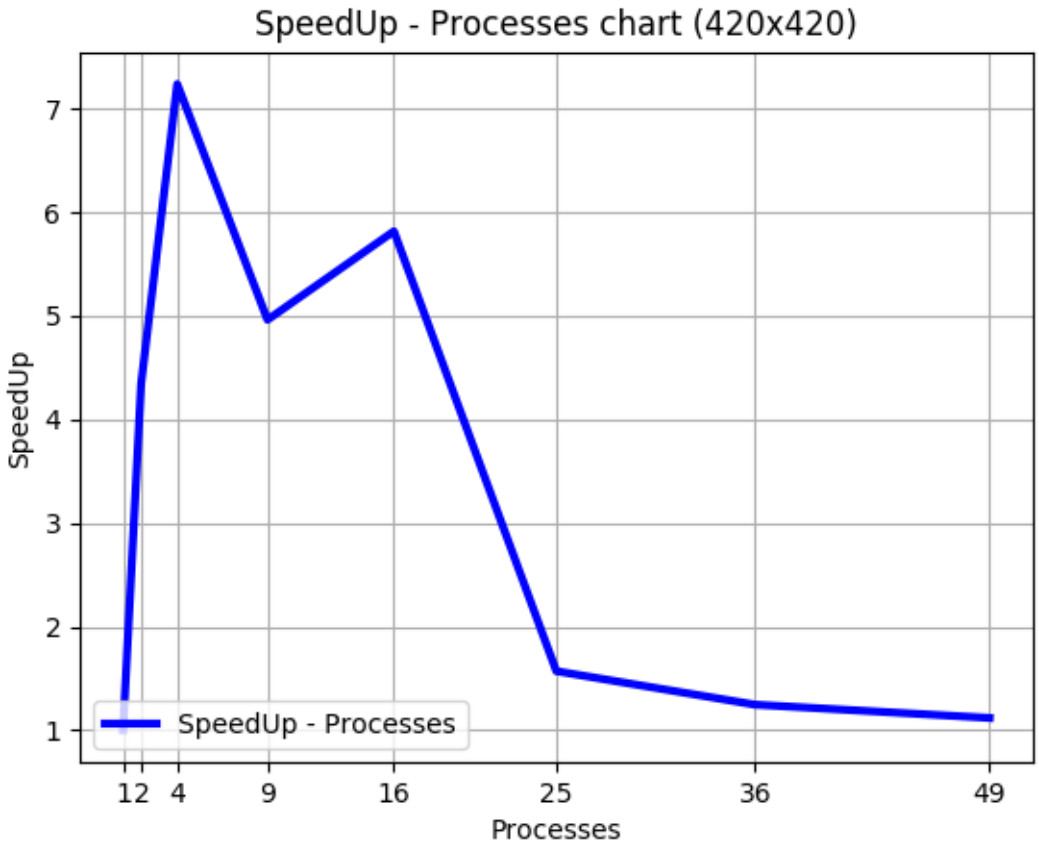
Size/Procs	1	2	4	9	16	25	36	49
420x420	1.48837	0.75762	0.43883	0.25370	1.64000	2.49634	3.08053	3.53603
840x840	5.93003	3.01064	2.57644	0.77096	1.74080	2.84387	3.23159	3.56870
1680x1680	27.03341	12.96863	6.06070	2.82255	2.60111	3.08731	3.91066	3.81804
3360x3360	114.05917	75.91677	25.93314	24.56814	25.34174	25.73041	12.02086	24.47199
6720x6720	615.87164	241.8863	22.75080	101.01855	197.7954	47.34381	73.92719	39.88797

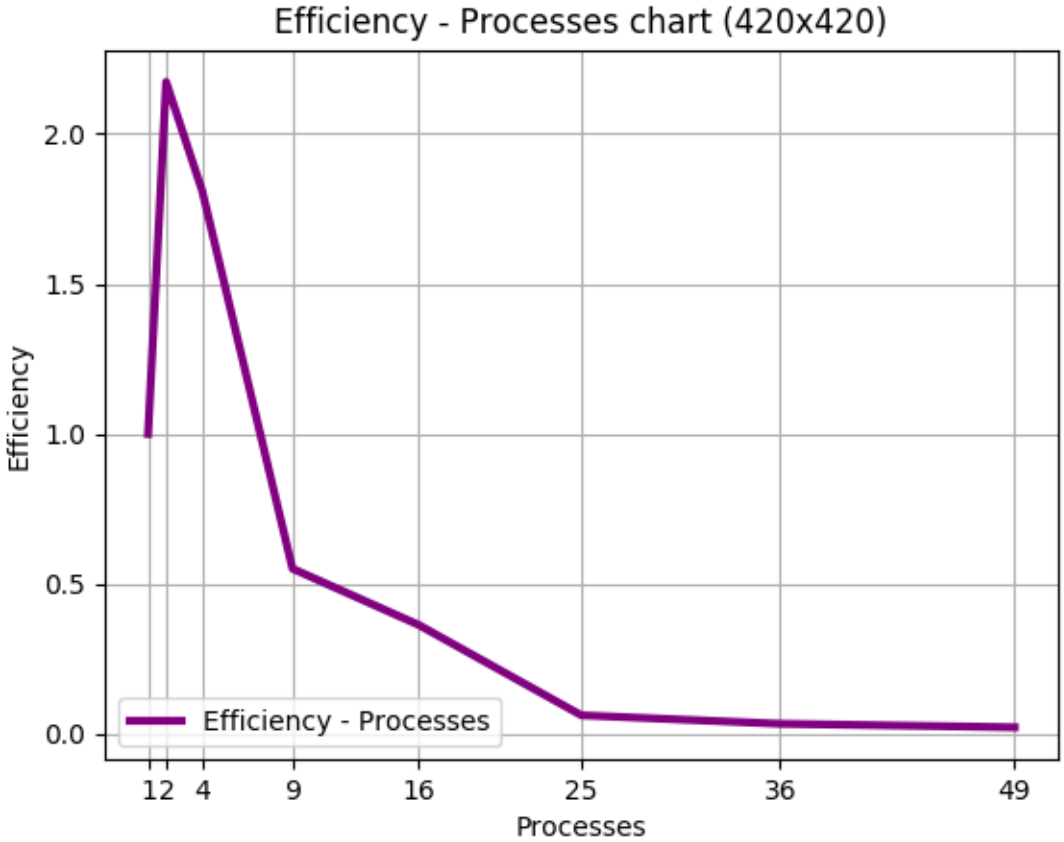
Για τις μετρήσεις των πειραμάτων μας, επιλέξαμε ως διαστάσεις του πλέγαματος τις 420x420, 840x840, 1680x1680 και 3360x3360.

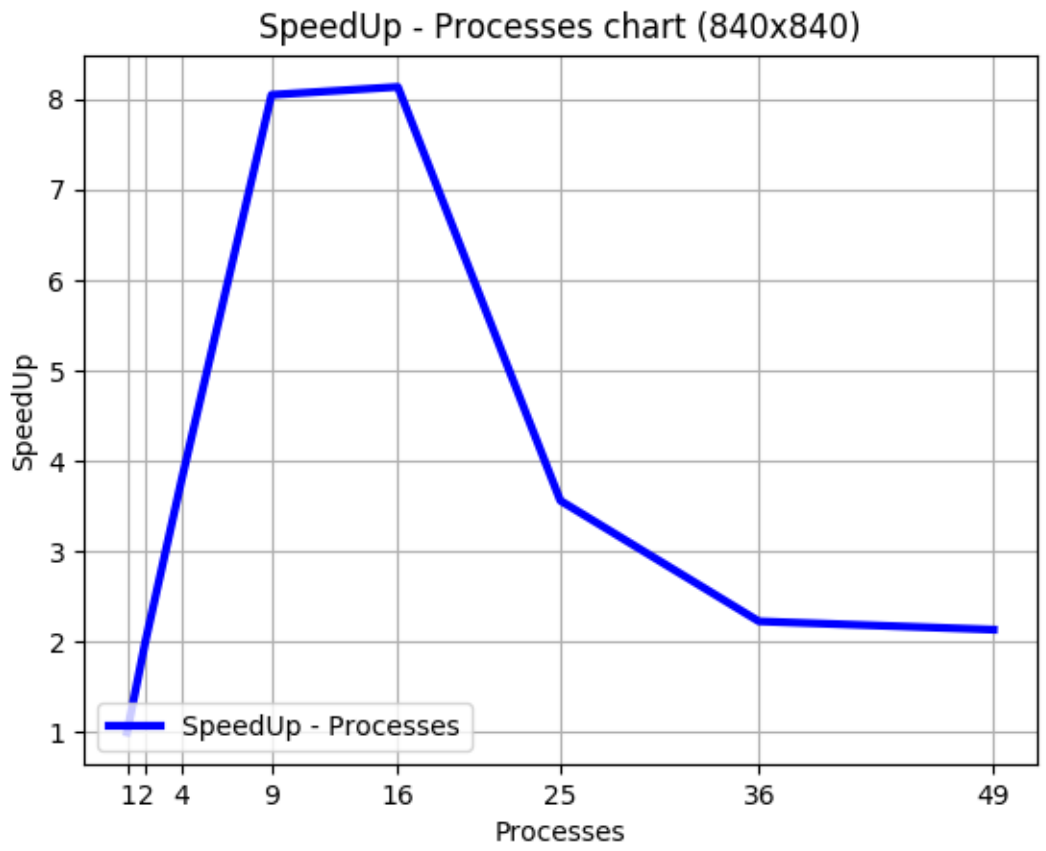
Για το MPI πρόγραμμα έχουμε 2 διαφορετικές εκτελέσεις των 250 επαναλήψεων, όπου η δεύτερη κάνει έλεγχο για πρόωρο τερματισμό ανά n γενιές, στην περίπτωση αυτή ανά 25 γενιές. Επιλέχθηκαν οι 25, ώστε να

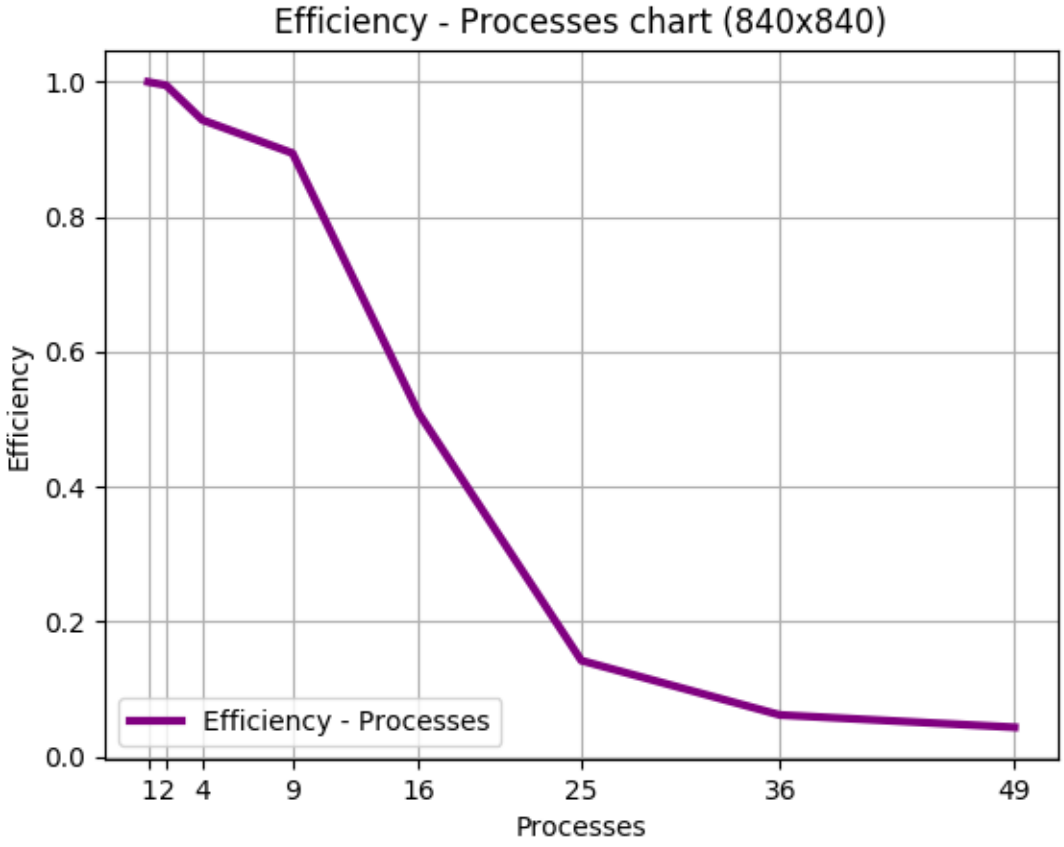
γίνει έλεγχος μόνο 10 φορές κατά την εκτέλεση του προγράμματος ώστε να μην προστεθεί μεγάλη καθυστέρηση κατά τις μετρήσεις.

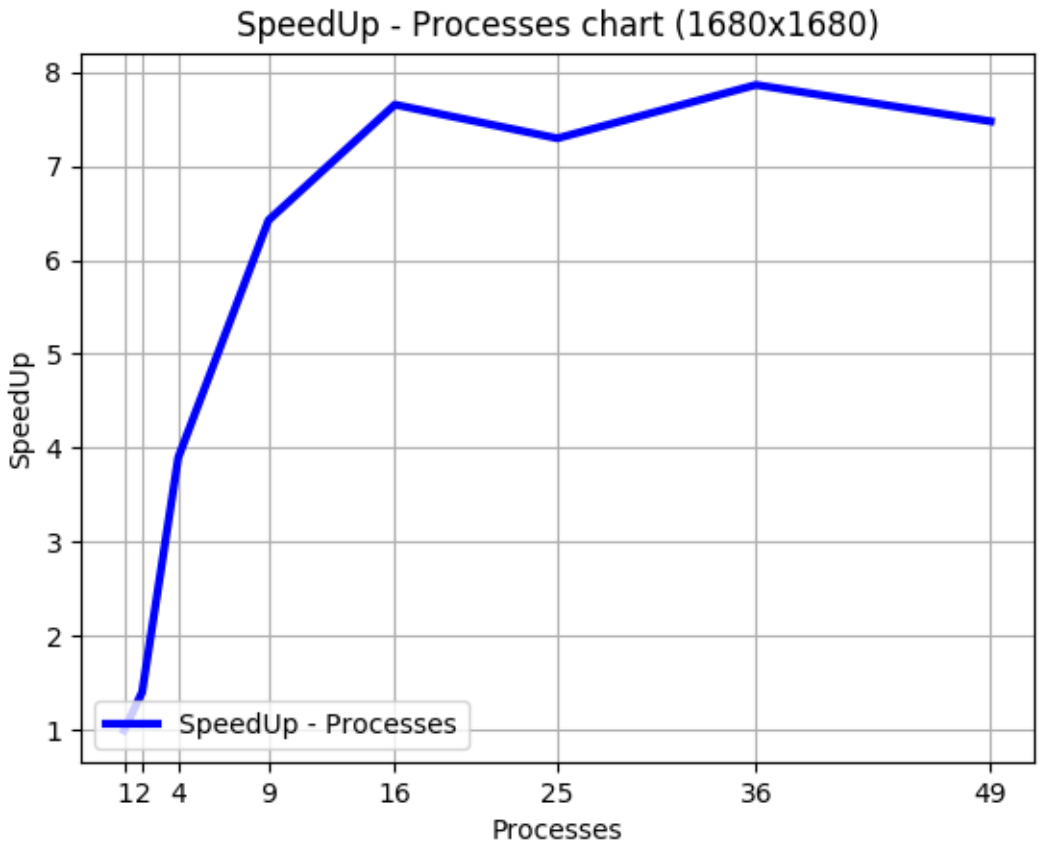
MPI (no termination check) :

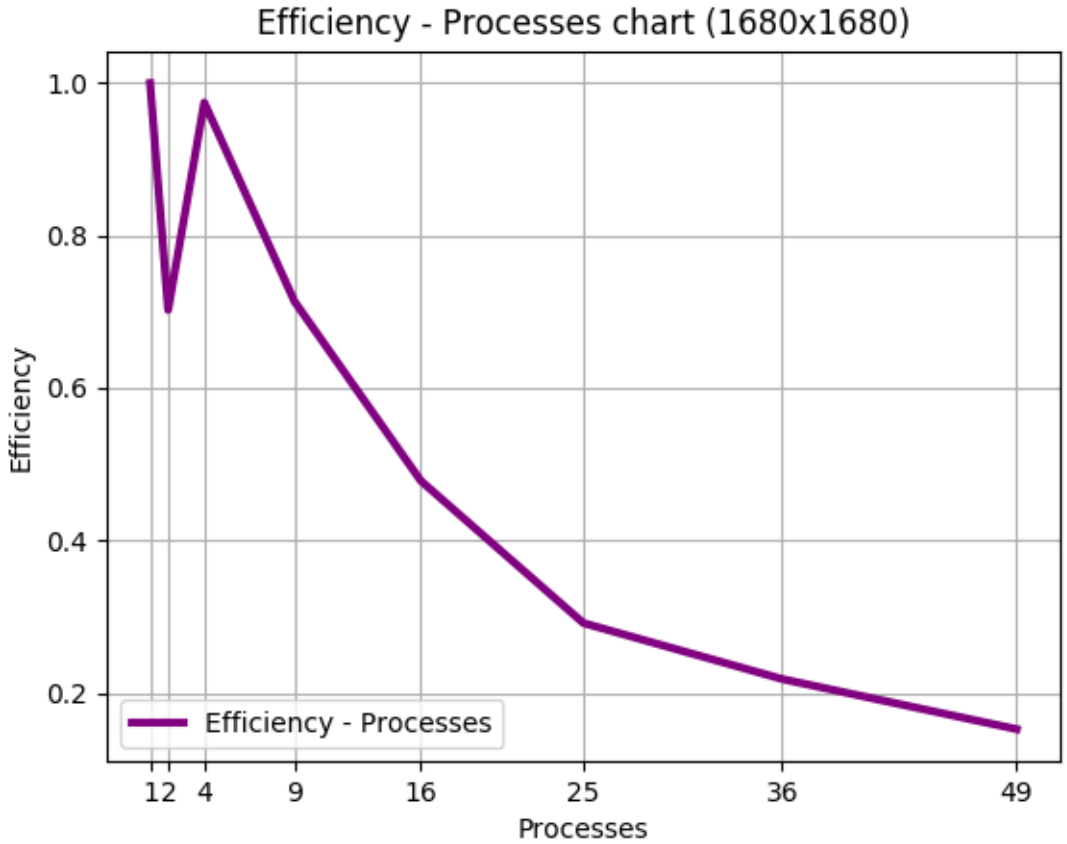


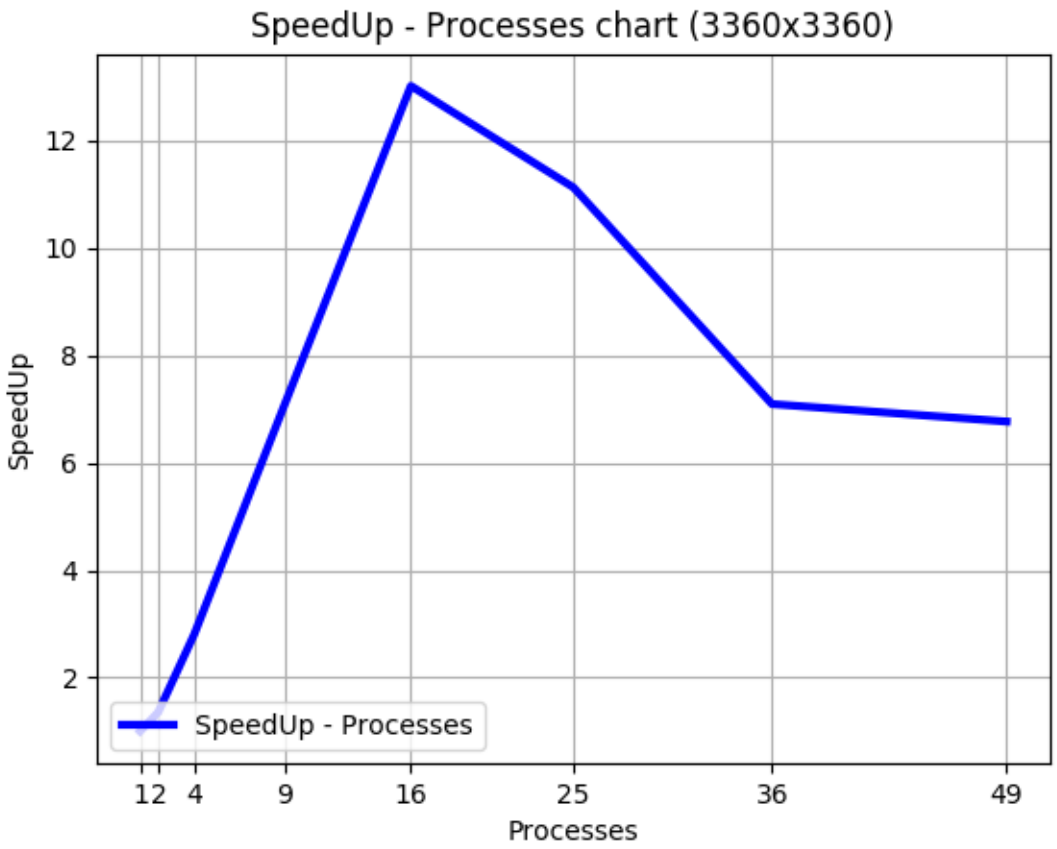


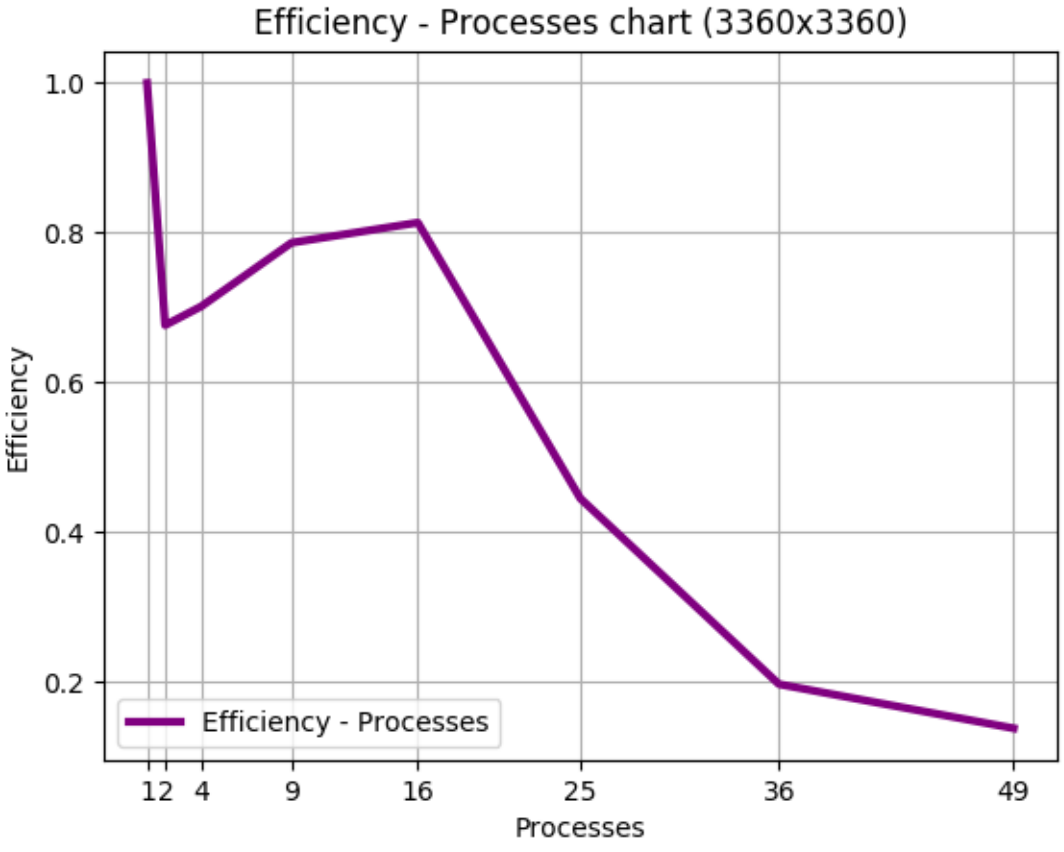

SpeedUp - Processes chart (840x840)
SpeedUp
Processes
SpeedUp - Processes
1
2
3
4
5
6
7
8
12
4
9
16
25
36
49


Efficiency - Processes chart (840x840)
Efficiency
Processes
Efficiency - Processes
0.0
0.2
0.4
0.6
0.8
1.0
12
4
9
16
25
36
49

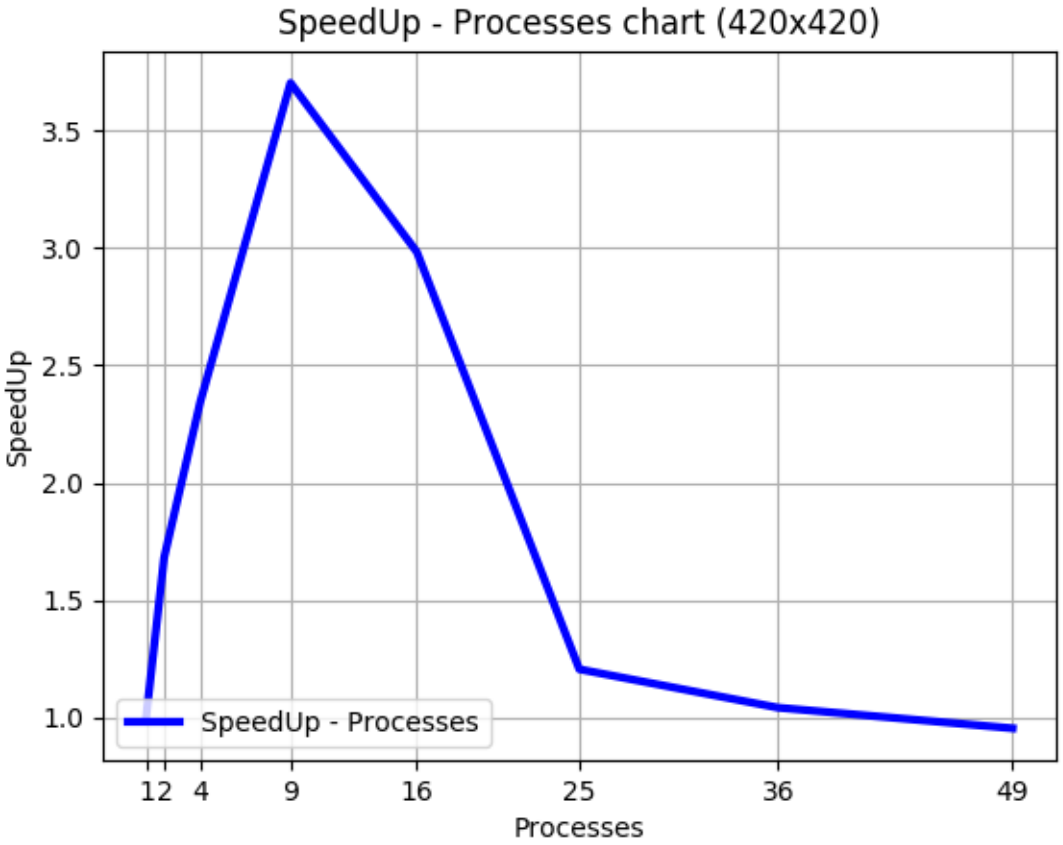

SpeedUp - Processes chart (1680x1680)
SpeedUp
Processes
SpeedUp - Processes
1
2
3
4
5
6
7
8
12
4
9
16
25
36
49

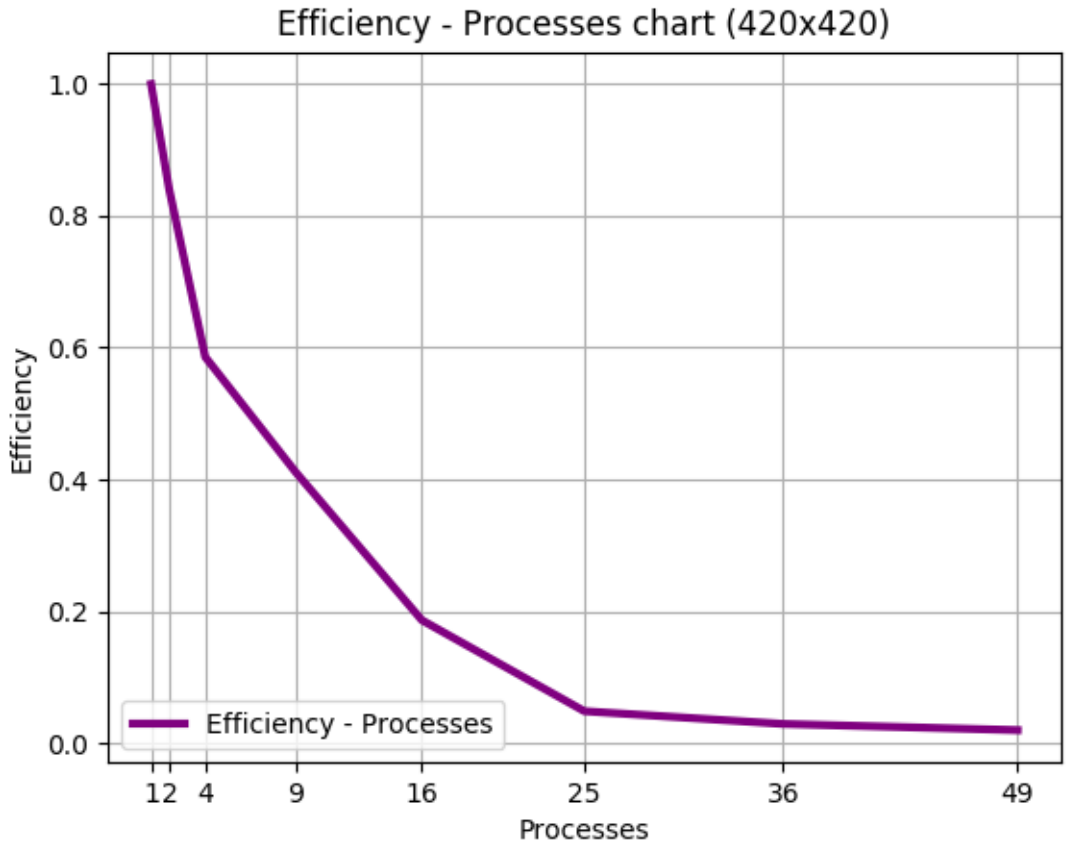

Efficiency - Processes chart (1680x1680)
Efficiency
Processes
Efficiency - Processes
0.2
0.4
0.6
0.8
1.0
12
4
9
16
25
36
49

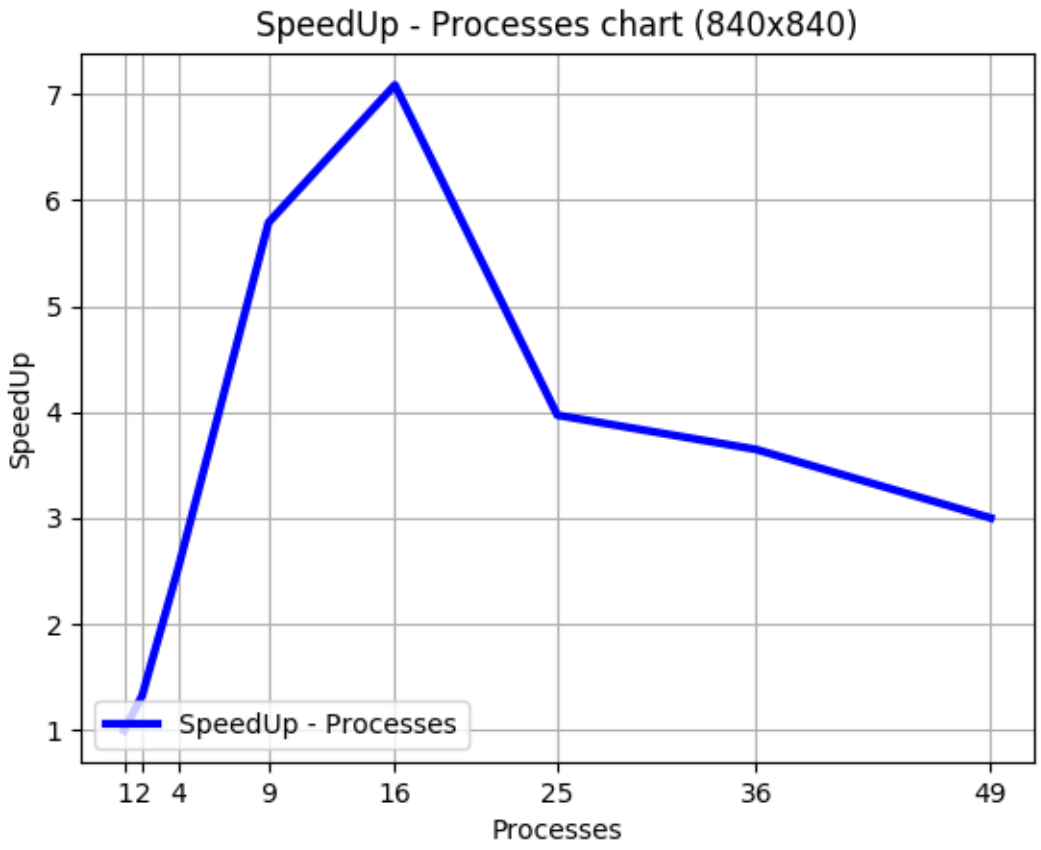

SpeedUp - Processes chart (3360x3360)
SpeedUp
12
10
8
6
4
2
SpeedUp - Processes
12
4
9
16
25
36
49
Processes

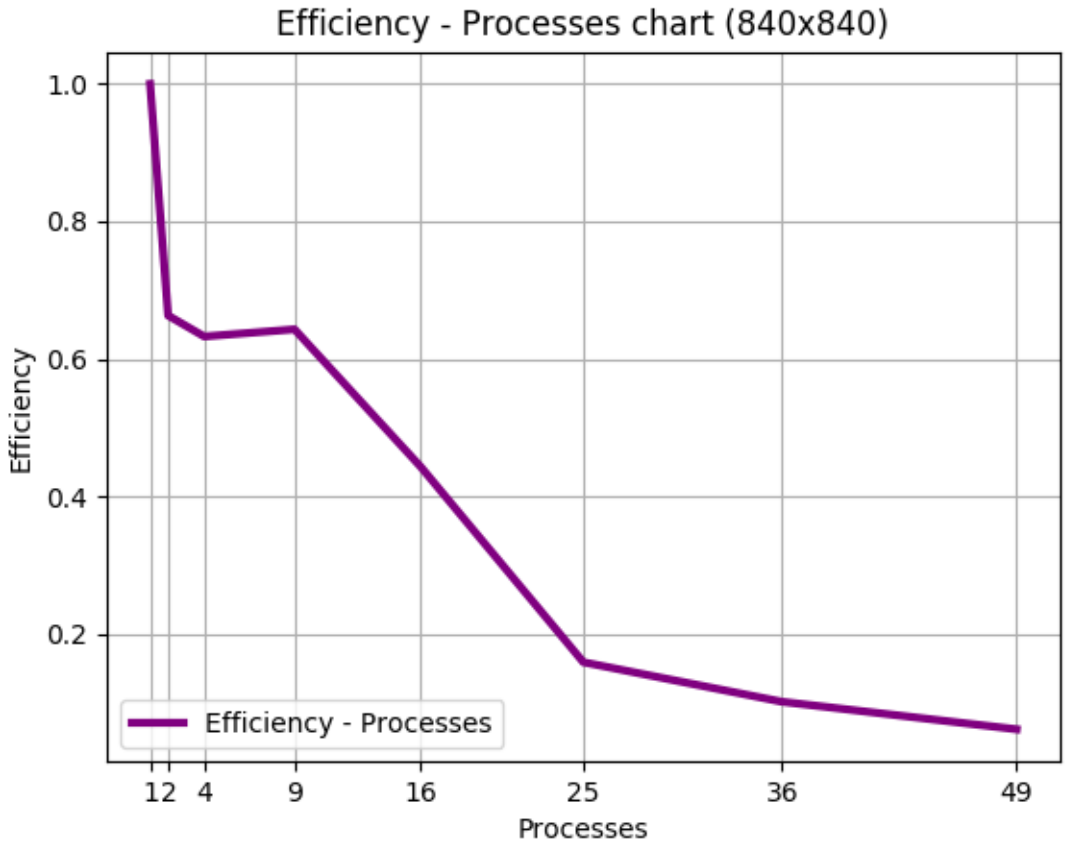

Efficiency - Processes chart (3360x3360)
Efficiency
1.0
0.8
0.6
0.4
0.2
Efficiency - Processes
12
4
9
16
25
36
49
Processes

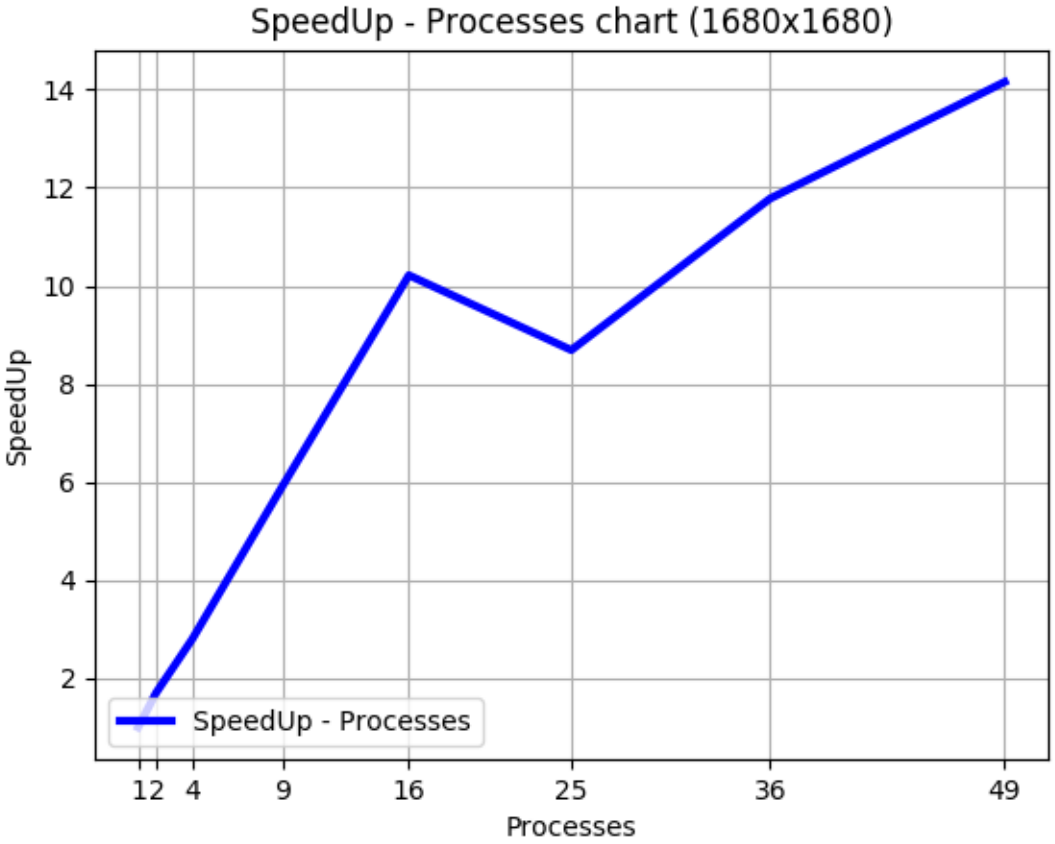
MPI(with termination check) :



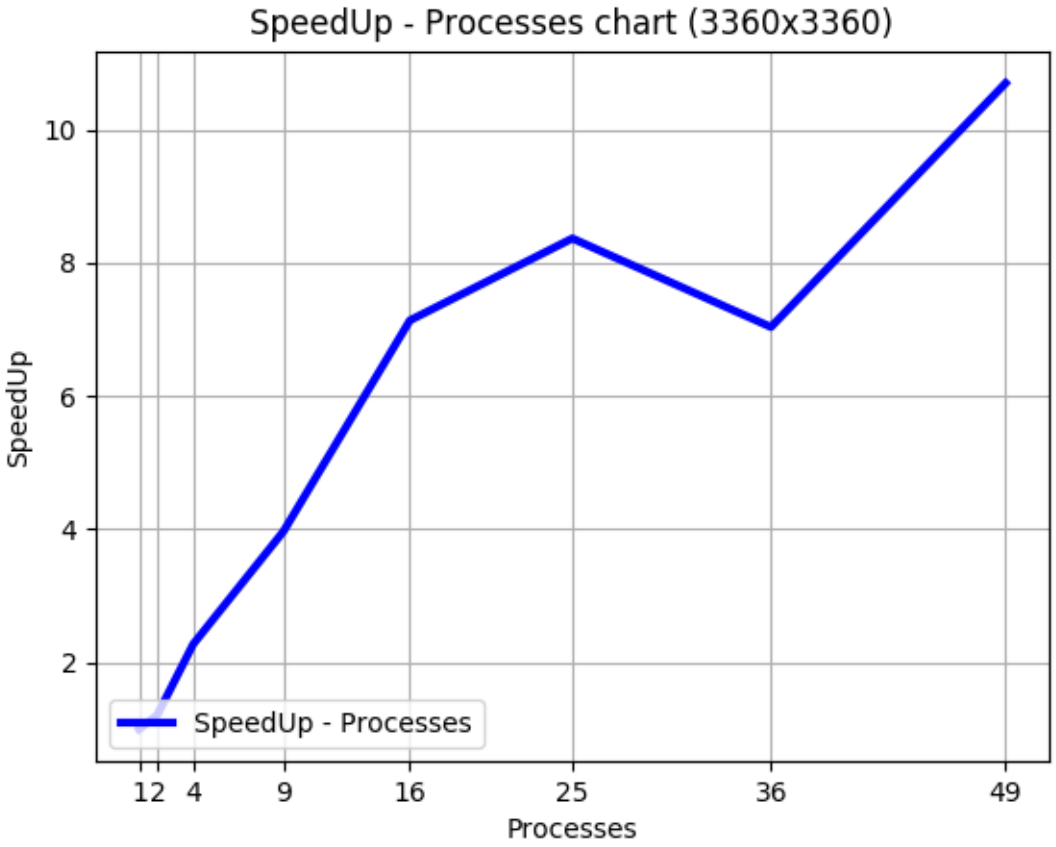


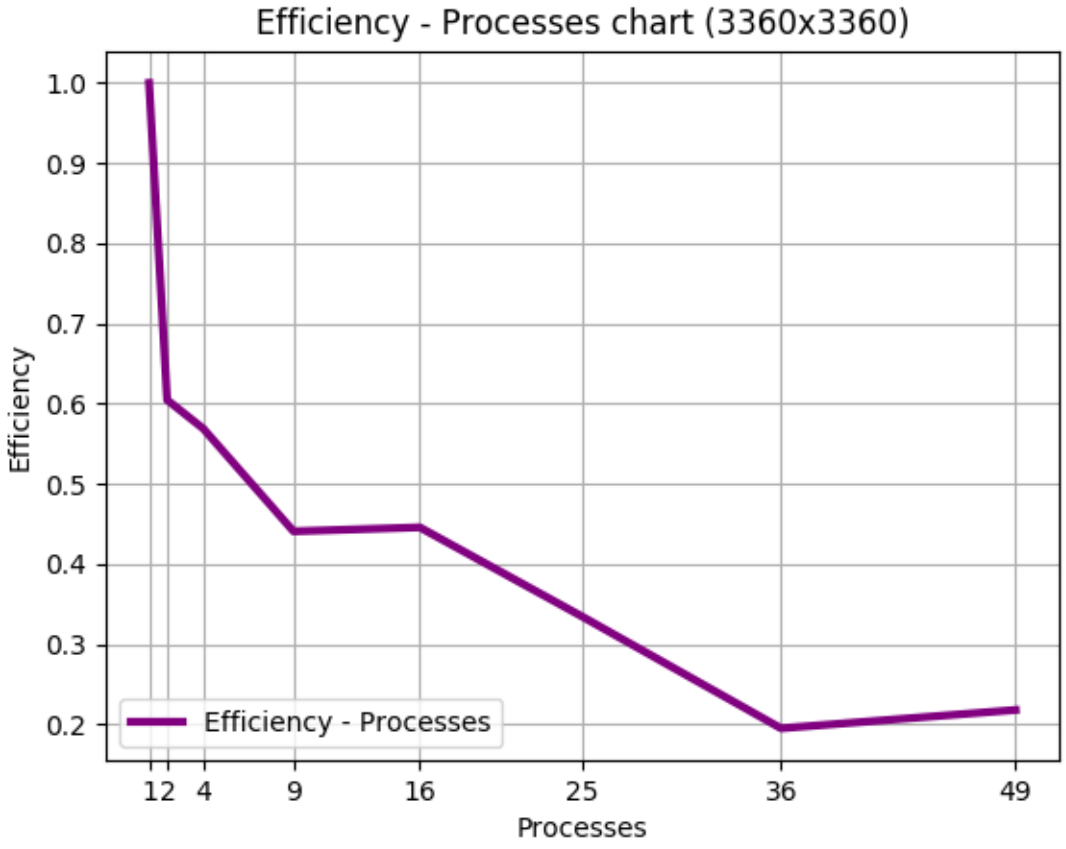

SpeedUp - Processes chart (840x840)
SpeedUp
Processes
SpeedUp - Processes
1
2
3
4
5
6
7
12
4
9
16
25
36
49


Efficiency - Processes chart (840x840)
Efficiency
Processes
Efficiency - Processes
0.2
0.4
0.6
0.8
1.0
12
4
9
16
25
36
49

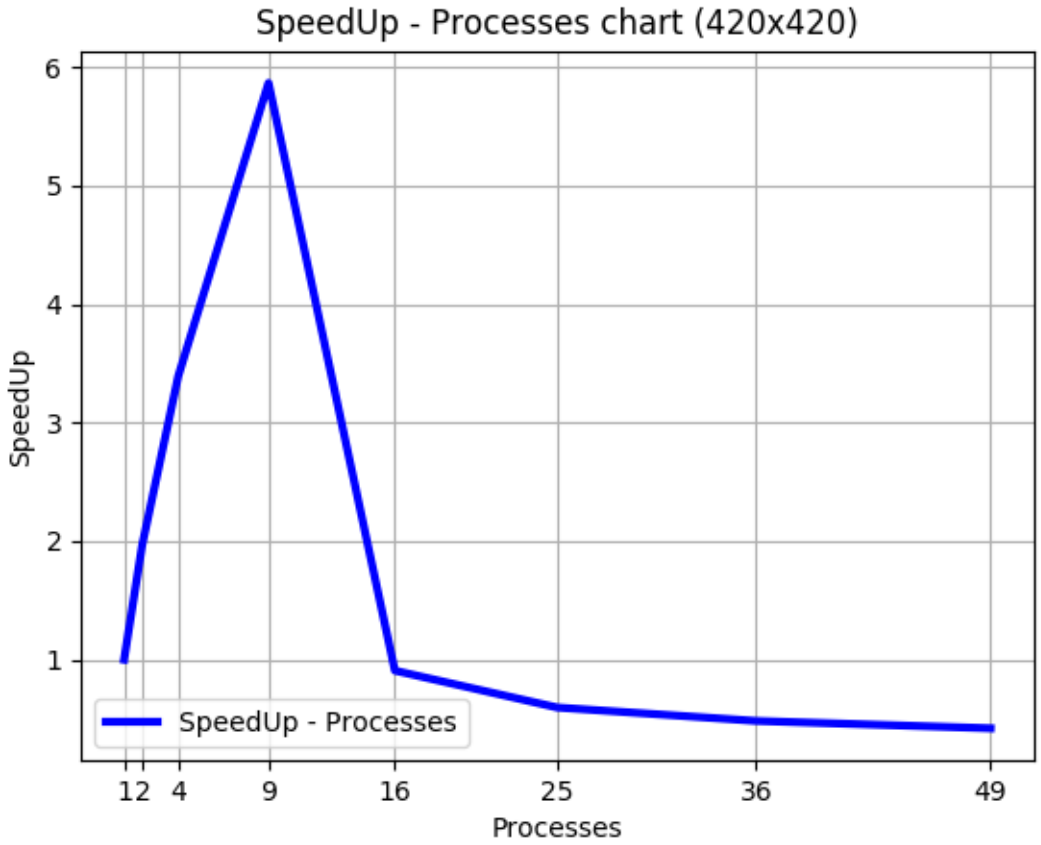

SpeedUp - Processes chart (1680x1680)
SpeedUp
14
12
10
8
6
4
2
SpeedUp - Processes
12
4
9
16
25
36
49
Processes

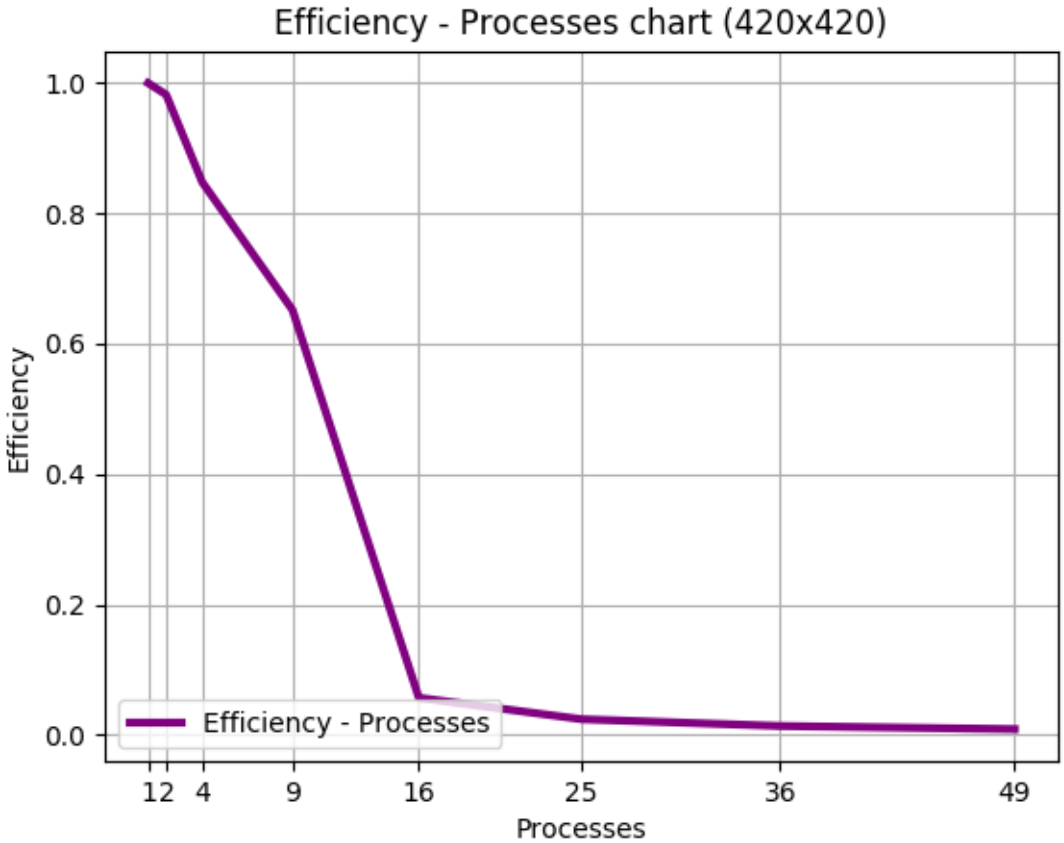

Efficiency - Processes chart (1680x1680)
Efficiency
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Efficiency - Processes
12
4
9
16
25
36
49
Processes

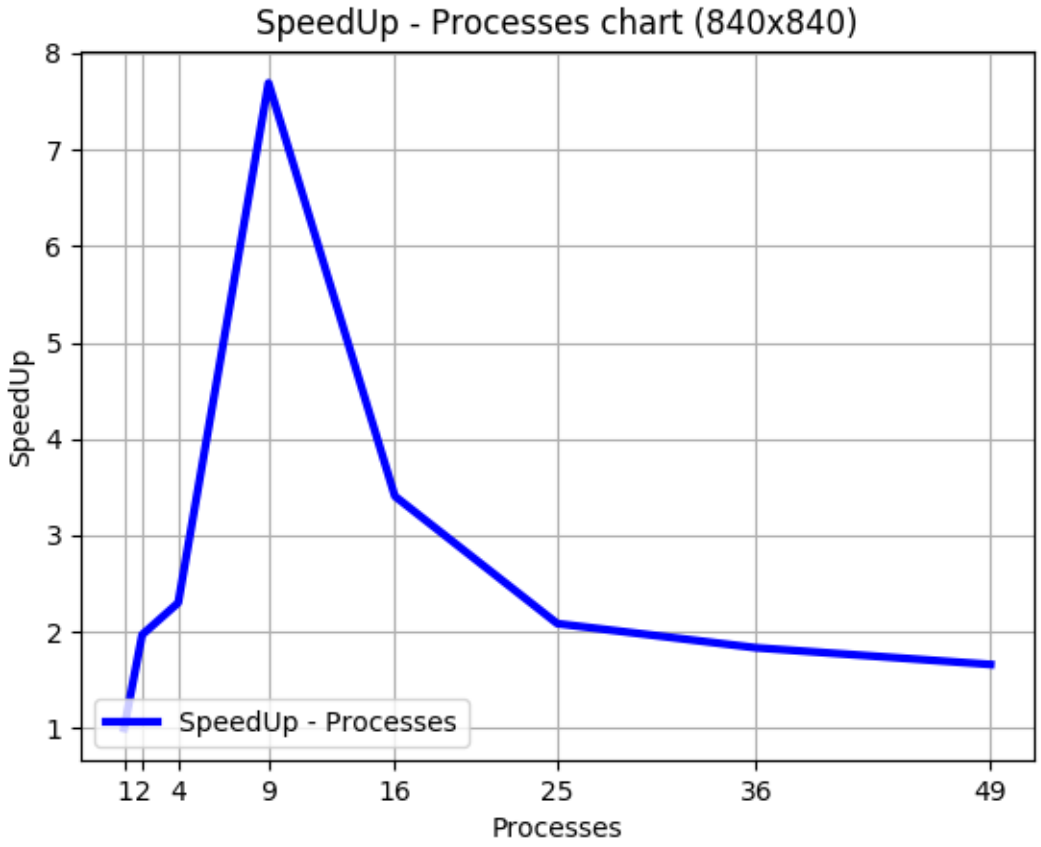

SpeedUp - Processes chart (3360x3360)
SpeedUp
10
8
6
4
2
SpeedUp - Processes
12
4
9
16
25
36
49
Processes

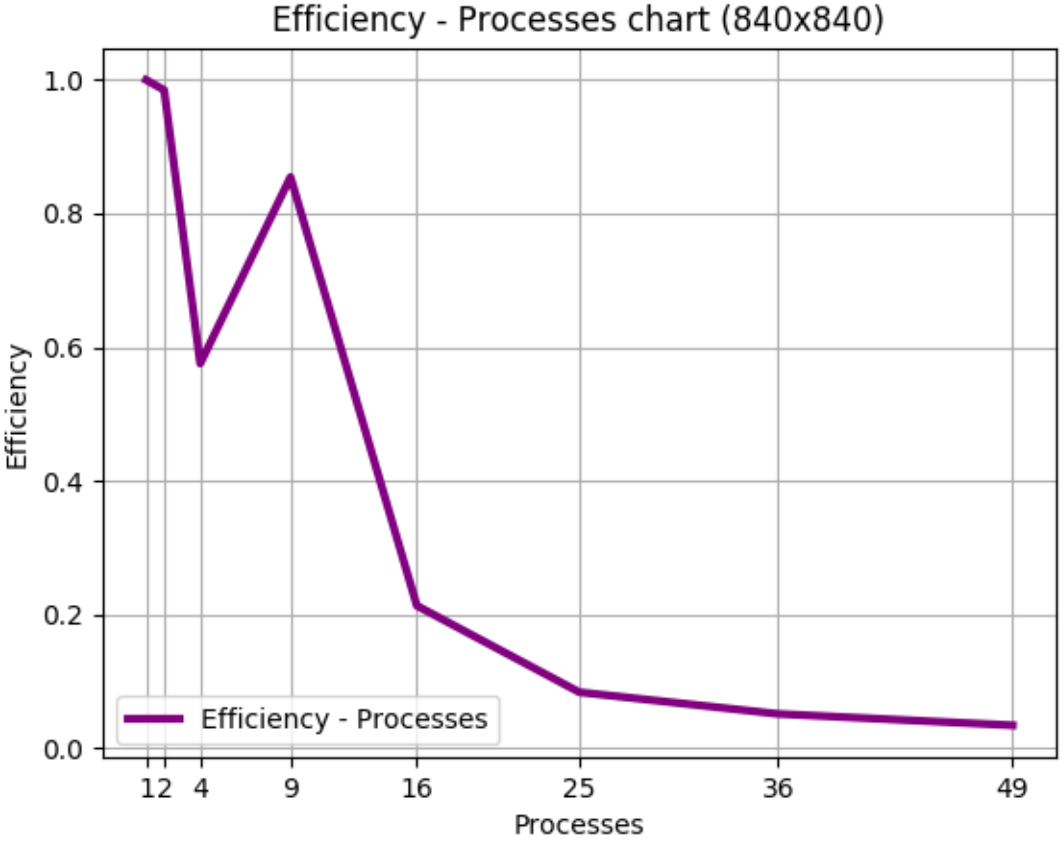

Efficiency - Processes chart (3360x3360)
Efficiency
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
Efficiency - Processes
12
4
9
16
25
36
49
Processes

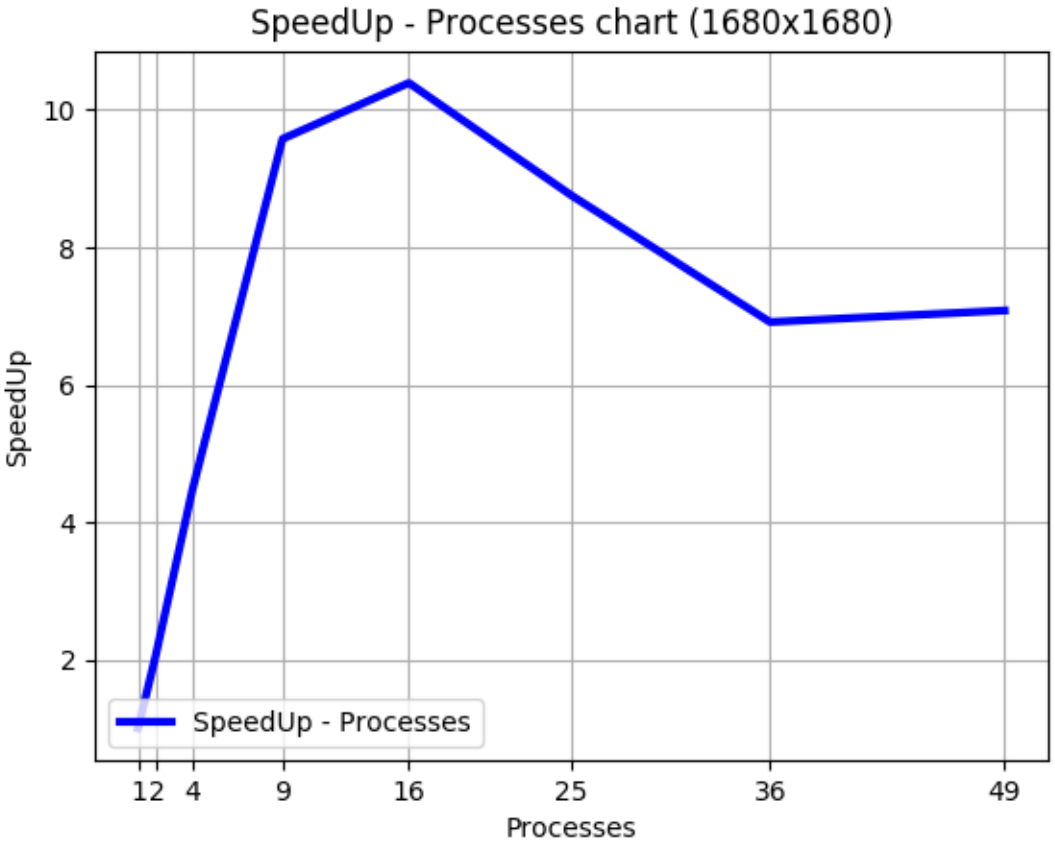
OPENMP-MPI(no termination check) :

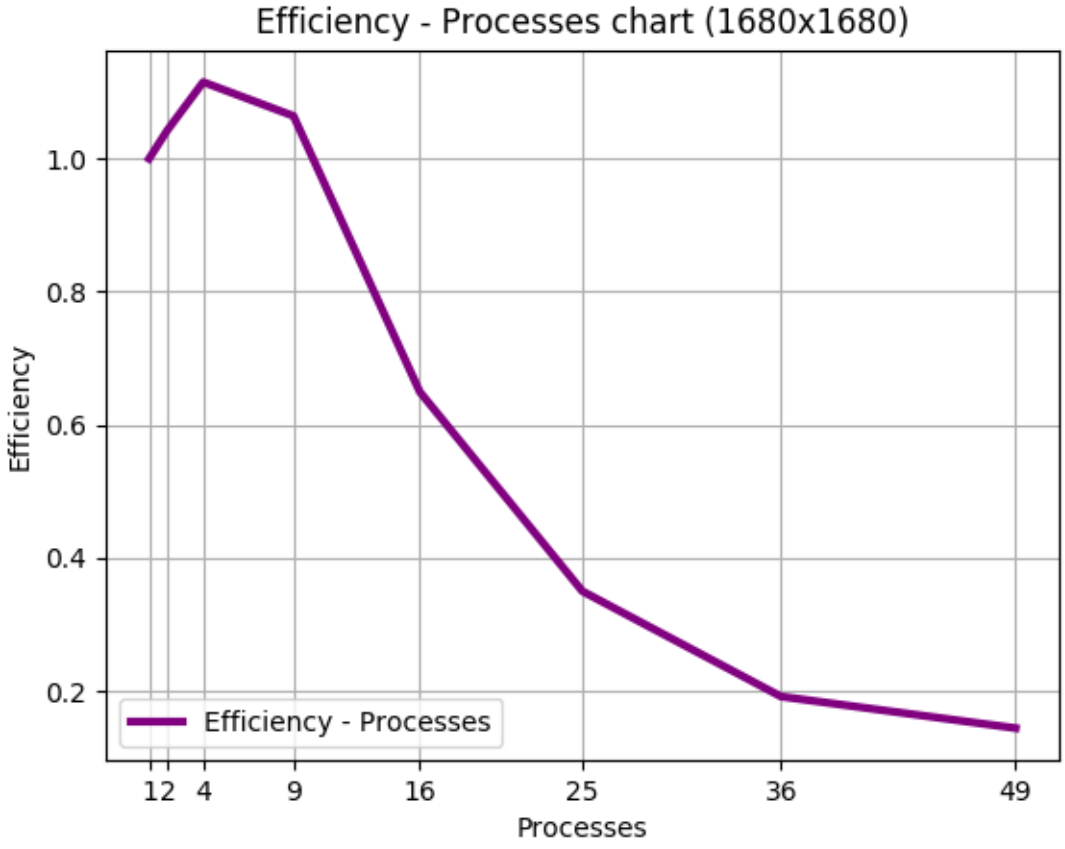


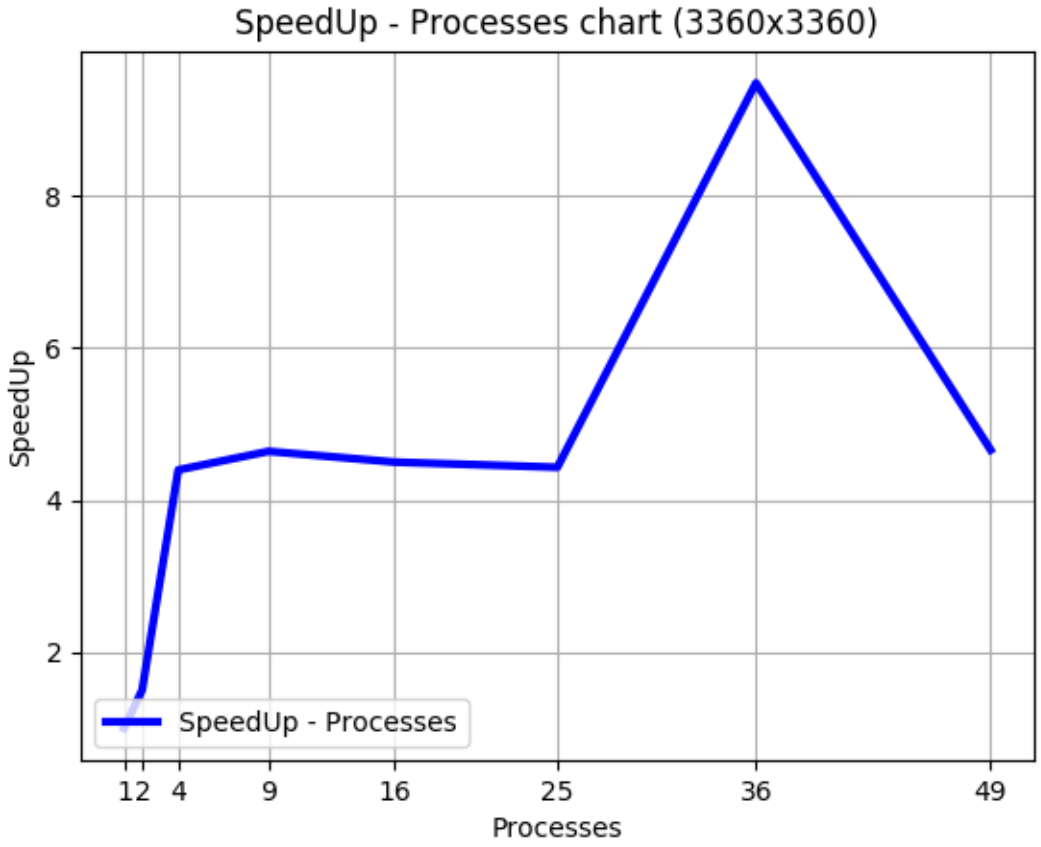


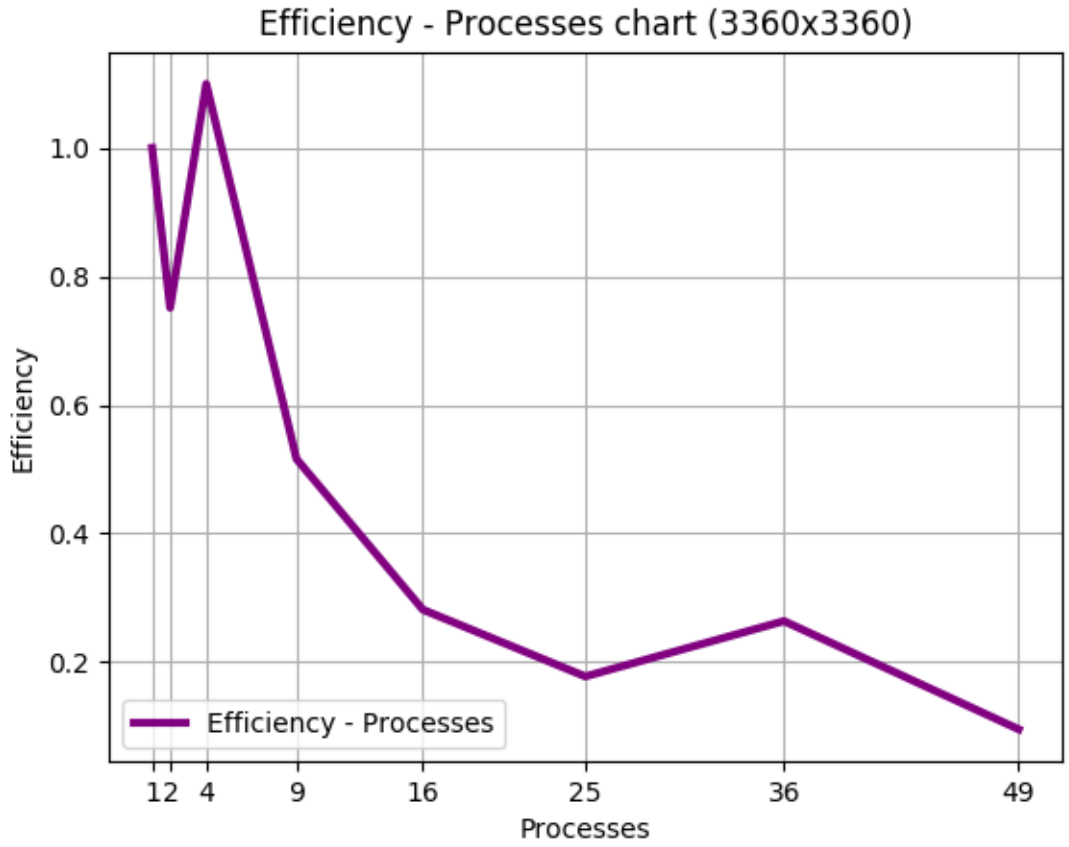

SpeedUp - Processes chart (840x840)
SpeedUp
Processes
SpeedUp - Processes
1
2
3
4
5
6
7
8
12
4
9
16
25
36
49


Efficiency - Processes chart (840x840)
Efficiency
Processes
Efficiency - Processes
0.0
0.2
0.4
0.6
0.8
1.0
12
4
9
16
25
36
49


SpeedUp - Processes chart (1680x1680)
SpeedUp
10
8
6
4
2
SpeedUp - Processes
12
4
9
16
25
36
49
Processes


Efficiency - Processes chart (1680x1680)
Efficiency
1.0
0.8
0.6
0.4
0.2
Efficiency - Processes
12
4
9
16
25
36
49
Processes


SpeedUp - Processes chart (3360x3360)
SpeedUp
8
6
4
2
SpeedUp - Processes
12
4
9
16
25
36
49
Processes


Efficiency - Processes chart (3360x3360)
Efficiency
1.0
0.8
0.6
0.4
0.2
Efficiency - Processes
12
4
9
16
25
36
49
Processes

CUDA:

Δεν καταφέραμε να χρησιμοποιήσουμε το compute node καθώς φαινόταν ότι είναι συνεχώς απασχολημένο (ή απόν;). Συνεπώς δεν έχουμε μετρήσεις για την ταχύτητα του προγράμματός μας.

7. Συμπεράσματα

Από τα παραπάνω αποτελέσματα (μετρήσεις), βγαίνουν τα ακόλουθα συμπεράσματα:

1. Για μικρό μέγεθος προβλημάτων, μετά από έναν αριθμό διεργασιών ο χρόνος εκτέλεσης μεγαλώνει. Αυτό συμβαίνει γιατί η επικοινωνία μεταξύ των διεργασιών κοστίζει.

2. Η αποτελεσματικότητα (efficiency) μειώνεται σημαντικά όσο μεγαλώνει ο αριθμός διεργασιών.

3. Η αποτελεσματικότητα του OpenMP γίνεται αισθητή για μεγαλύτερο αριθμό διεργασιών.

4. Στα MPI/OPENMP προέκυψε μεγάλη βελτίωση στα αποτελέσματα των μετρήσεων όταν χρησιμοποιήσαμε την MPI_WaitAny() έναντι της MPI_Wait(). Το παραπάνω είναι λογικό καθώς το πρόγραμμα μπορεί να καθυστερούσε περιμένοντας συγκεκριμένες διεργασίες να αποστείλουν τις πληροφορίες τους, ενώ πληροφορίες άλλων είχαν ήδη ολοκληρωθεί.

Ωστόσο, καθώς υπήρχε μεγάλη κίνηση στα μηχανήματα της σχολής και πολλές φορές οι μετρήσεις είχαν τεράστια απόκλιση από προηγούμενες προσπάθειες κατά την εκτέλεσή τους, είναι δύσκολο να προκύψουν αξιόπιστα συμπεράσματα.