ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΟΝ ΗΛΕΚΤΡΟΝΙΚΟ ΑΥΤΟΜΑΤΙΣΜΟ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μετρήσεις απόδοσης και οπτικοποίηση συμπεριφοράς παράλληλων προγραμμάτων με χρήση των Paraver και Extrae

Ιωάννης Δ. Μονιός

Επιβλέποντες: **Ιωάννης Κοτρώνης,** Επίκουρος Καθηγητής

ΑΘΗΝΑ

ΜΑΡΤΙΟΣ 2015

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μετρήσεις απόδοσης και οπτικοποίηση συμπεριφοράς παράλληλων προγραμμάτων με χρήση των Paraver και Extrae

Ιωάννης Δ. Μονιός

Α.Μ.: 2010509

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Ιωάννης Κοτρώνης** Επίκουρος Καθηγητής

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Ιωάννης Κοτρώνης** Επίκουρος Καθηγητής, **Ιζαμπώ Καράλη** Επίκουρη Καθηγήτρια, **Παναγιώτης Σταματόπουλος** Επίκουρος Καθηγητής

Μάρτιος 2015

ΠΕΡΙΛΗΨΗ

Σκοπός της εργασίας αυτής είναι η παρουσίαση των προγραμμάτων Extrae και Paraver, τα οποία έχουν αναπτυχθεί από το Barcelona Supercomputing Center και οπτικοποιούν και αναλύουν τη συμπεριφορά παράλληλων προγραμμάτων. Η παρουσίαση γίνεται με χρήση απλών παράλληλων προγραμμάτων και διαφορετικών υλοποιήσεών τους, μέσω των οποίων επεξηγείται ο τρόπος λειτουργίας των Extrae και Paraver. Επίσης παραθέτονται κάποια παραδείγματα άντλησης πληροφοριών με τη χρήση τους.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Παράλληλος προγραμματισμός.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Οπτικοποίηση συμπεριφοράς, εντοπισμός λαθών, εντοπισμός καθυστερήσεων, βελτιστοποίηση απόδοσης, κλιμάκωση.

ABSTRACT

The purpose of this dissertation project is the presentation of programs Extrae and Paraver, which have been developed by the Barcelona Supercomputing Center and visualize and analyze the behavior of parallel programs. The presentation is made with the use of simple parallel programs and different implementations of them, through which the function of Extrae and Paraver is explained. Furthermore, some examples are given of how to derive information with their use.

SUBJECT AREA: Parallel Programing

KEYWORDS: Behavior visualization, error tracking, delays tracking, performance optimization, scalability.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Για την εργασία αυτή έγινε χρήση του εργαστηρίου Linux του τμήματος Πληροφορικής και Τηλεπικοινωνιών του ΕΚΠΑ, στο οποίο υπάρχουν εγκατεστημένα τα Extrae και Paraver. Στα παραρτήματα Ι και ΙΙ υπάρχουν οδηγίες εγκατάστασης των προγραμμάτων για περιβάλλον Linux. Υπάρχουν ακόμα οδηγίες εύρεσης των απαραίτητων αρχείων, τα οποία παρέχονται και σε συμπληρωματικό CD. Τα αρχεία κώδικα και τα παραγόμενα αρχεία από το Extrae των οποίων έγινε χρήση επίσης παρέχονται στο συμπληρωματικό CD.

1. ΕΙΣΑΓΩΓΗ

Η τεχνολογία των υπολογιστών στις μέρες μας έχει φτάσει σε τέτοιο σημείο, όπου η περεταίρω βελτίωση των επεξεργαστών είναι αδύνατη. Αυτό συμβαίνει διότι η κατανάλωση ρεύματος και κατά συνέπεια η θερμοκρασία που θα αναπτυσσόταν κατά τη λειτουργία ακόμα γρηγορότερων επεξεργαστών είναι απαγορευτικές, αφού οι διαρροές ρεύματος που προκαλούνται στα τρανζίστορ σε τέτοιες συνθήκες καταστούν τους επεξεργαστές αναξιόπιστους.

Για να ξεπεραστεί αυτό το πρόβλημα γίνεται χρήση παραλληλισμού. Δηλαδή, αντί να κατασκευάζονται όλο και γρηγορότεροι και πολύπλοκοι επεξεργαστές, γίνεται χρήση πολλών απλούστερων και πιο αργών επεξεργαστών σε ένα τσιπ. Αυτά τα συστήματα ονομάζονται πολυπύρηνοι επεξεργαστές, όπου ο πυρήνας αναφέρεται σε μία κεντρική μονάδα επεξεργασίας (CPU).

Όμως η χρήση πολυπύρηνων επεξεργαστών απαιτεί και αντίστοιχα προγράμματα που να εκμεταλλεύονται την αρχιτεκτονική αυτή, αφού η εκτέλεση παλιότερων προγραμμάτων παράλληλα σε πολλούς πυρήνες δε προσφέρει επιτάχυνση, αλλά απλά πολλαπλές εκτελέσεις του ίδιου προγράμματος. Αυτό που αναμένεται είναι να εκτελούνται πιο γρήγορα τα προγράμματα. Για να γίνει αυτό πρέπει να γραφτούν είτε νέα προγράμματα, είτε προγράμματα που θα μεταφράζουν τα παλιά, έτσι ώστε να μπορούν να εκμεταλλευτούν τους πυρήνες. Η έρευνα, όμως έχει δείξει ότι η μετάφραση προγραμμάτων δεν αποδίδει, οπότε πρέπει να δημιουργηθούν νέα προγράμματα.

Σε πολλές περιπτώσεις, όμως και τα νέα προγράμματα δεν αποδίδουν όπως θεωρητικά αναμένεται. Για το λόγο αυτό οι κατασκευαστές χρησιμοποιούν εργαλεία μέτρησης απόδοσης και οπτικοποίησης συμπεριφοράς παράλληλων προγραμμάτων, ώστε να εντοπίσουν και να χαρακτηρίσουν τα μη αποδοτικά τμήματα κώδικα που φέρουν φτωχά αποτελέσματα. Τέτοια εργαλεία είναι και τα **Extrae** και **Paraver**, τα οποία έχουν αναπτυχθεί από το **Barcelona Supercomputing Center** και παρέχονται με δωρεάν διάθεση. Η λειτουργία των προγραμμάτων αυτών συνοψίζεται στο ότι το Extrae μετά από συλλογή πληροφοριών από τους μετρητές των επεξεργαστών και του συστήματος γενικότερα, δημιουργεί αρχεία ονομαζόμενα **traces**, τα οποία οπτικοποιεί και αναλύει το Paraver.

2. EXTRAE

2.1 Λειτουργία

Το Extrae χρησιμοποιεί διάφορους μηχανισμούς για να εισάγει μετρητές στο προς μελέτη πρόγραμμα, με σκοπό τη συλλογή πληροφοριών σχετικά με την απόδοσή του. Μπορεί να χρησιμοποιηθεί με MPI, OpenMP, CUDA, OpenCL, pthread και OmpSs. Υπάρχει, επίσης δυνατότητα συνδυαστικής χρήσης του MPI με όλα τα υπόλοιπα. Η εισαγωγή μετρητών στο πρόγραμμα γίνεται με δύο τρόπους. Ο πρώτος τρόπος εισάγει μια διαμοιραζόμενη βιβλιοθήκη στο πρόγραμμα, πριν τη φόρτωσή του. Αν το πρόγραμμα και η διαμοιραζόμενη βιβλιοθήκη έχουν κοινά σύμβολα, τότε μέσω τον συμβόλων αυτών μπορεί να εισαχθεί κώδικας στο πρόγραμμα, δηλαδή μετρητές στη περίπτωση του Extrae. Στα συστήματα Linux αυτό επιτυγχάνεται με τη χρήση της μεταβλητής περιβάλλοντος **LD_PRELOAD**. Ο δεύτερος τρόπος κάνει χρήση του **Dyninst**, η οποία είναι μία βιβλιοθήκη ανεπτυγμένη από τα πανεπιστήμια του Μέρυλαντ και του Γουισκόνσιν-Μάντισον και η λειτουργία της είναι να εισάγει γραμμές κώδικα στο πηγαίο κώδικα του προγράμματος, σε συγκεκριμένα σημεία, κατά τη μεταγλώττιση του προγράμματος. Έτσι η διαδικασία δεν μεταβάλει το πηγαίο κώδικα και γίνεται μόνο μια φορά, εκτός και αν γίνει αλλαγή στο πρόγραμμα και πρέπει να μεταγλωττιστεί ξανά.

Για τα κομμάτια του προγράμματος για τα οποία δεν υπάρχουν πληροφορίες, το Extrae παρέχει δύο μηχανισμούς δειγματοληψίας για την συλλογή δεδομένων. Ο πρώτος μηχανισμός είναι οι signal timers, που ανά τακτά χρονικά διαστήματα συλλέγουν δείγματα. Ο δεύτερος μηχανισμός κάνει χρήση των μετρητών απόδοσης του επεξεργαστή, που συλλέγουν δείγματα σε συγκεκριμένα σημεία μεταξύ γεγονότων.

Τρία είναι τα πιο κοινά είδη δεδομένων που συλλέγονται από το Extrae, χρονικές σφραγίδες με ακρίβεια nanosecond, δεδομένα σχετικά με τον μικροεπεξεργαστή και δεδομένα που συνδέουν την απόδοση με το πηγαίο κώδικα, ώστε να μπορεί ο προγραμματιστής να ξέρει που ακριβώς δημιουργείται το πρόβλημα για να μπορέσει να επέμβει και να το διορθώσει*. Τα δεδομένα που σχετίζονται με τον μικροεπεξεργαστή συλλέγονται μέσω του **PAPI** (λεπτομέρειες στο Παράρτημα III), το οποίο προσφέρει πληροφορίες και για τη κατανάλωση ρεύματος και τη θερμοκρασία του μικροεπεξεργαστή, αλλά και πληροφορίες για το σκληρό δίσκο, το δίκτυο, το λειτουργικό σύστημα κ.α., αρκεί να υπάρχουν οι αντίστοιχοι μετρητές στο hardware.

2.2 Χρήση

Για να δημιουργηθεί το αρχείο trace ενός προγράμματος μέσω του Extrae, πρέπει αρχικά να έχει γίνει μετάφρασή του και να έχει δημιουργηθεί το binary αρχείο. Στη συνέχεια πρέπει να δημιουργηθεί ένα αρχείο xml με τις παραμέτρους λειτουργίας του Extrae, όπως και ένα αρχείο sh με τις παραμέτρους εκτέλεσης του Extrae (λεπτομέρειες στο Παράρτημα I).

Η εντολή για την εκτέλεση του Extrae είναι:

`mpiexec -n` <# διεργασιών> `-f` <αρχείο με τις μηχανές στις οποίες θα εκτελεστεί το πρόγραμμα> `./`<όνομα αρχείου>`.sh ./`<binary αρχείο> <όνομα παραγόμενου αρχείου>

(πχ: `mpiexec -n 16 -f machines ./mpitrace.sh ./mpiprog mpiprog_trace`)

**Στο Παράρτημα IV υπάρχει επικοινωνία μέσω e-mail, με την Tools Group Manager Judit Gimenez του Barcelona Supercomputing Center σχετικά με τη συγκεκριμένη λειτουργία, η οποία παρουσιάζει πρόβλημα. Για καλύτερη κατανόηση προτείνεται πρώτα ανάγνωση της παρουσίασης του Paraver*, κεφάλαιο 3.2.

Μεταξύ των αρχείων που δημιουργούνται, υπάρχουν και 3 αρχεία με το όνομα που δόθηκε στην εντολή mpiexec και τις επεκτάσεις **.prv .pcf** και **.row**. Το αρχείο .prv περιέχει τη δραστηριότητα του προγράμματος, το .pcf περιέχει τις ετικέτες που σχετίζονται με την αριθμητικές τιμές και το .row περιέχει πληροφορίες για τη χρήση των πόρων. Τα υπόλοιπα αρχεία μπορούν να διαγραφούν, είναι προσωρινά.

3. PARAVER

3.1 Λειτουργία

Το Paraver αναπτύχθηκε λόγω της ανάγκης ύπαρξης, αρχικά μιας γενικής και ποιοτικής αντίληψης της συμπεριφοράς των προγραμμάτων, μέσω οπτικοποίησης, και στη συνέχεια την επικέντρωση σε λεπτομερείς και ποσοτικές αναλύσεις των προβλημάτων. Το Paraver προσφέρει πολλές πληροφορίες για τη συμπεριφορά ενός προγράμματος. Οι πληροφορίες αυτές άμεσα επηρεάζουν τον τρόπο επέμβασης στο κώδικα του προγράμματος για τη βελτίωσή του. Έτσι μειώνεται ο χρόνος που απαιτείται για την ανάπτυξη ενός προγράμματος και περιορίζονται οι απαιτήσεις σε υλικό.

Υπάρχουν τριών ειδών απεικονίσεις στο Paraver. Γραφική απεικόνιση, ώστε να γίνεται εύκολα αντιληπτή από το χρήστη η συμπεριφορά ενός προγράμματος, κείμενο που παρέχει όλες τις λεπτομέρειες και επεξηγήσεις της γραφικής απεικόνισης και αναλυτική απεικόνιση όπου παρέχονται ποσοτικά δεδομένα .

Στη γραφική απεικόνιση τα αντικείμενα που αναπαριστούνται έχουν άμεση σχέση με τη δομή και τον τρόπο λειτουργίας του προγράμματος και χωρίζονται σε προγράμματα (applications), διεργασίες (tasks) και νήματα (threads). Οι πληροφορίες για τα αντικείμενα αυτά χωρίζονται με τη σειρά τους σε χρονικές τιμές για κάθε αντικείμενο, σημαίες που αντιστοιχούν σε γεγονότα και γραμμές επικοινωνίας ανάμεσα στα αντικείμενα.

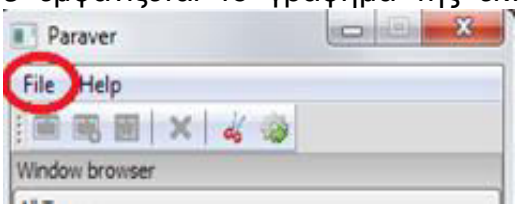
Κατά την γραφική απεικόνιση παραθέτονται οι τιμές και τα γεγονότα, που λαμβάνουν χώρα κατά τη διάρκεια του προγράμματος, χωρίς να τους αναθέτονται ερμηνείες και ιδιαίτερες σημασίες. Είναι στην ευχέρεια του χρήστη να το κάνει αυτό. Παραθέτονται, όμως ορισμένα εργαλεία για το πώς θα δημιουργηθεί η απεικόνιση, που οδηγούν σε μια ευρεία γκάμα επιλογών και αποτελεσμάτων. Έτσι, για παράδειγμα όταν απεικονίζονται νήματα, μια συνάρτηση για τα νήματα υπολογίζει, από τα δεδομένα που περιγράφουν τη δραστηριότητα των νημάτων, τη τιμή που θα χρησιμοποιηθεί για την γραφική απεικόνιση. Όταν απεικονίζεται διεργασία, χρησιμοποιείται μια συνάρτηση για τα νήματα, για κάθε νήμα της διεργασίας, και στη συνέχεια με τη χρήση μιας συνάρτησης για διεργασίες υπολογίζεται από τις τιμές των νημάτων η τελική τιμή που θα χρησιμοποιηθεί για την γραφική απεικόνιση της διεργασίας. Αντίστοιχα υπολογίζεται η τιμή για τη γραφική απεικόνιση ενός προγράμματος από τις τιμές των διεργασιών από τις οποίες αποτελείται.

Με παρόμοιο τρόπο, υπάρχει η δυνατότητα υπολογισμού πολλών παραμέτρων, σχετικών με την απόδοση ενός προγράμματος, από βασικές τιμές των μετρητών του συστήματος. Για παράδειγμα από το μετρητή των κύκλων και το μετρητή των εντολών μπορεί να προκύψει πληροφορία για τις εντολές ανά κύκλο παρόλο που αρχικά δεν υπήρχε μέτρηση για κάτι τέτοιο. Οι νέες πληροφορίες που προκύπτουν, από το συνδυασμό άλλων μπορούν να αποθηκευτούν σε αρχείο και να φορτωθούν ξανά. Τα αρχεία αυτά ονομάζονται **configuration files (cfgs)**. Τα αρχεία αυτά δε, μπορούν να χρησιμοποιηθούν με trace αρχεία από διαφορετικά προγράμματα, μιας και στην ουσία σε αυτά αποθηκεύεται ο τρόπος διεξαγωγής των επιθυμητών πληροφοριών από το αρχείο tracee και όχι αυτές καθαυτές οι πληροφορίες. Μαζί με το Paraver παρέχονται έτοιμα configuration files που μπορούν να χρησιμοποιηθούν με οποιοδήποτε trace αρχείο για να παραχθούν επιπλέον δεδομένα και στατιστικά. Αυτά βρίσκονται στο φάκελο `cfgs`, στο φάκελο εγκατάστασης του Paraver.

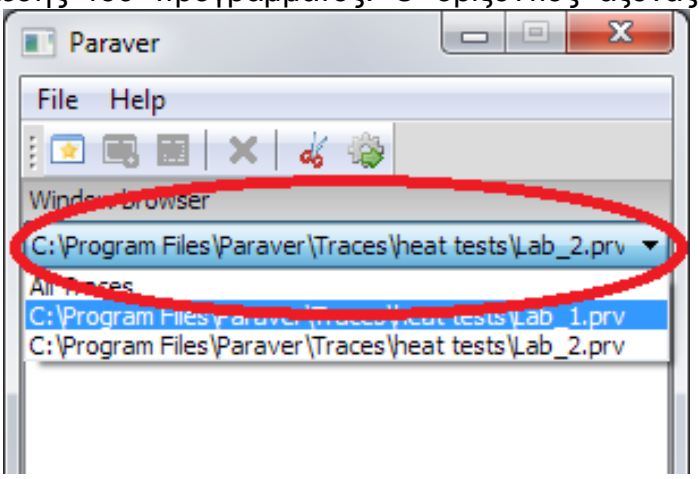
3.2 Παρουσίαση

Για τη παρουσίαση θα χρησιμοποιηθεί ένα πρόγραμμα προσομοίωσης μετάδοσης θερμότητας (λεπτομέρειες στη παράγραφο 3.3.1 όπου αναφέρονται αναλυτικά όλες οι υλοποιήσεις του, που χρησιμοποιούνται για την επεξήγηση της χρήσης του Paraver)

Αφού δημιουργηθούν τα απαιτούμενα αρχεία με τη χρήση του Extrae, φορτώνονται στο Paraver πατώντας **File** στο αρχικό παράθυρο του Paraver (*Εικόνα 1*) και επιλέγοντας **Load Trace...** στο νέο menu που προκύπτει. Στο παράθυρο που εμφανίζεται γίνεται επιλογή του επιθυμητού αρχείου με επέκταση .prv που δημιουργήθηκε από το Extrae. Αν έχουν φορτωθεί πολλά traces ταυτόχρονα, τότε από το σημείο που φαίνεται στην *Εικόνα 2* γίνεται εναλλαγή μεταξύ τους. Πατώντας το εικονίδιο που φαίνεται στην *Εικόνα 3* εμφανίζεται το γράφημα της εκτέλεσης του προγράμματος. Ο οριζόντιος άξονας



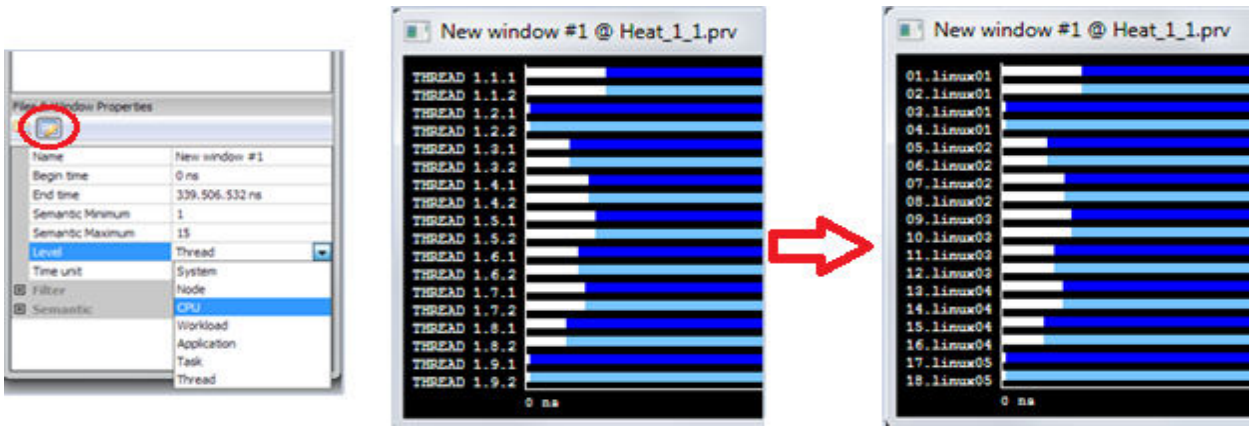
Εικόνα 1: Κουμπί File



Εικόνα 2: Εναλλαγή αρχείων trace

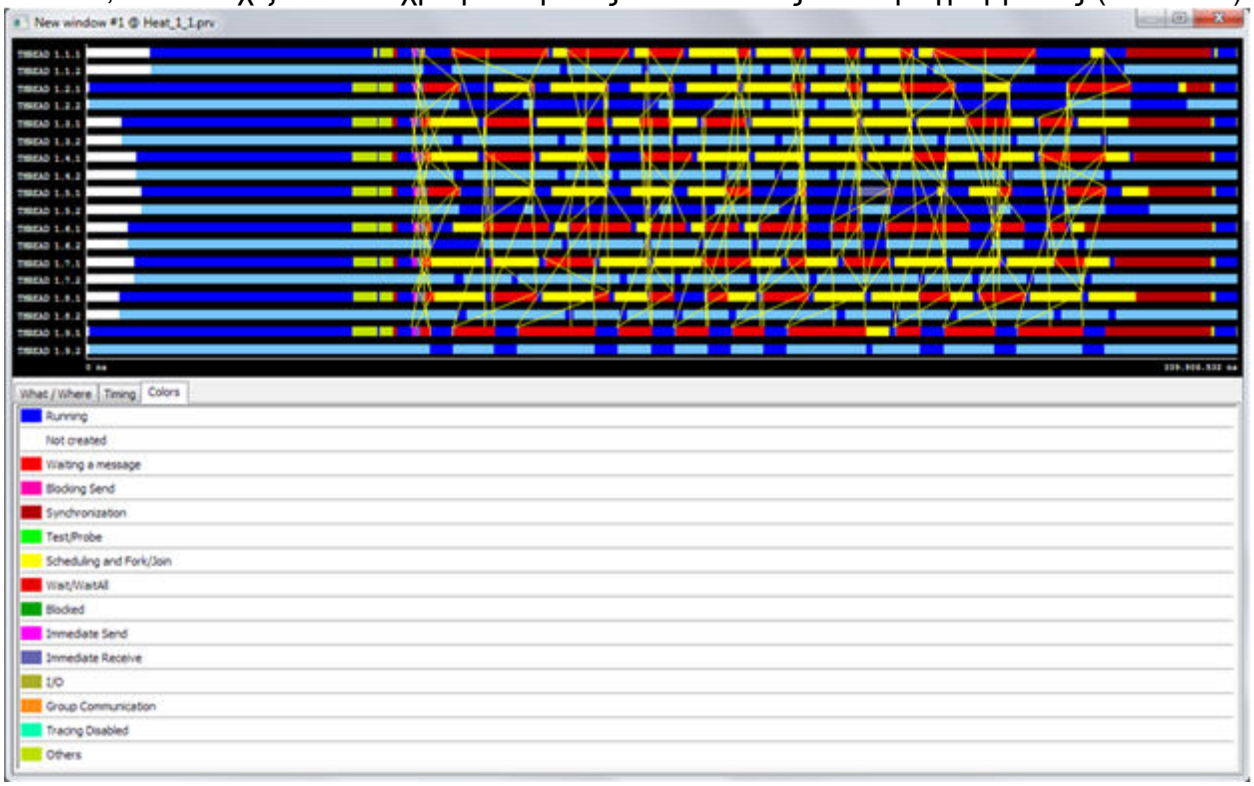
Εικόνα 3: Κουμπί γραφήματος

αντιστοιχεί στο χρόνο και ο κατακόρυφος στα νήματα. Κάθε νήμα χαρακτηρίζεται από τρεις αριθμούς. ο πρώτος αριθμός απαριθμεί τις εφαρμογές, ο δεύτερος τις διεργασίες και ο τρίτος τα νήματα. Νήματα με τον ίδιο αριθμό εφαρμογής ανήκουν στην ίδια εφαρμογή και αντίστοιχα ισχύει για τις διεργασίες. Πατώντας το κουμπί που φαίνεται στην *Εικόνα 4* εμφανίζεται μια σειρά πληροφοριών για το γράφημα. Η γραμμή με τίτλο **Level** έχει σχέση με το επίπεδο στο οποίο αναφέρονται οι πληροφορίες, δηλαδή επίπεδο εφαρμογής, διεργασίας, νήματος κ.τ.λ. Κάνοντας κλικ στο δεξί πεδίο της γραμμής εμφανίζεται μια λίστα με επιλογές. Επιλέγοντας CPU, στο κατακόρυφο άξονα των γραφημάτων παρουσιάζεται, αντί του αριθμού του νήματος, το όνομα του επεξεργαστή στον οποίο εκτελείται το νήμα. Έτσι, μπορεί να γίνει επεξεργασία των πληροφοριών λαμβάνοντας υπόψη και την τοπολογία των επεξεργαστών, η οποία μπορεί να εξηγεί τυχόν καθυστερήσεις στην επικοινωνία μεταξύ των διεργασιών.



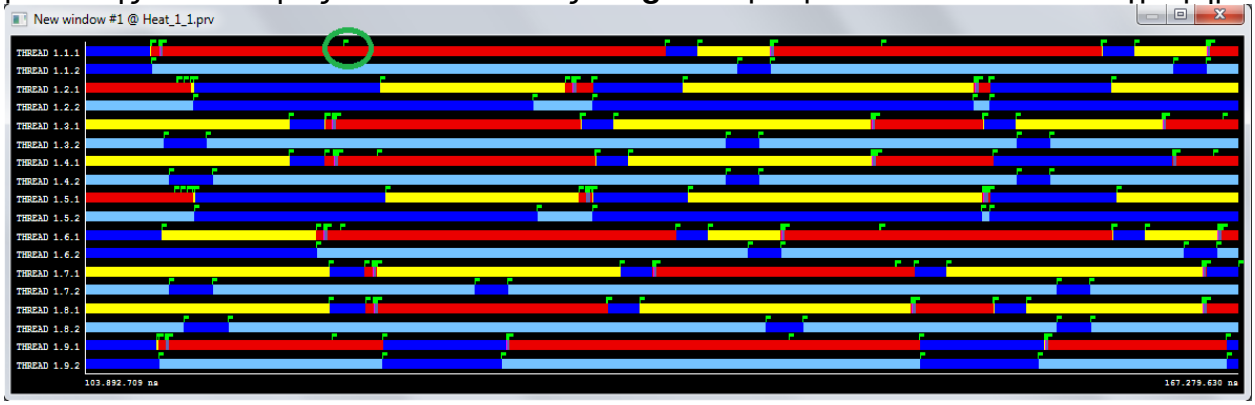
Εικόνα 4: Αλλαγή επιπέδου πληροφοριών

Πατώντας δεξί κλικ στο γράφημα και επιλέγοντας **Info Panel**, κάτω από το γράφημα εμφανίζεται ένα παράθυρο στο οποίο παρουσιάζονται πληροφορίες. Στο πρώτο tab, με τίτλο **What/Where**, εμφανίζονται πληροφορίες για το σημείο του γραφήματος στο οποίο γίνεται διπλό κλικ. Στο δεύτερο tab με τίτλο **Timing**, σε περίπτωση που γίνεται μελέτη συγκεκριμένου τμήματος του tracee αρχείου, παρουσιάζονται οι χρονικές στιγμές αρχής και τέλους και ο χρόνος συνολικής διάρκειας του τμήματος. Στο τρίτο tab, με τίτλο **Colors**, αντιστοιχίζονται τα χρώματα με τις καταστάσεις του προγράμματος (*Εικόνα 5*).



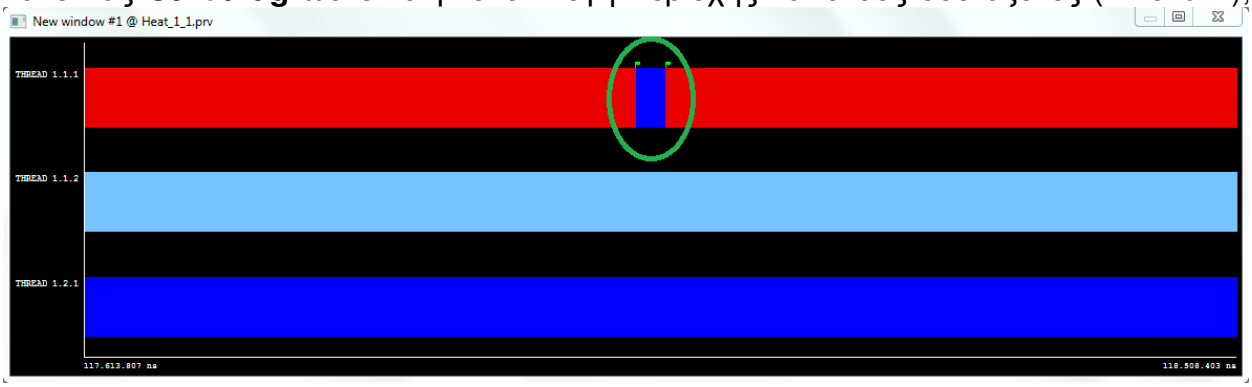
Εικόνα 5: Αρχικό γράφημα με Info Panel

Πατώντας δεξί κλικ στο γράφημα και κάνοντας uncheck την επιλογή **View → Communication Lines** εξαφανίζονται οι κίτρινες γραμμές, σχετικές με την επικοινωνία μεταξύ των διεργασιών, και υπάρχει καλύτερη εικόνα των καταστάσεων του προγράμματος μέσω των χρωμάτων. Στο menu **View** υπάρχει και η επιλογή **Event Flags**. Κάνοντας check την επιλογή αυτή σηματοδοτούνται τα σημεία, για κάθε διεργασία, όπου εκτελείται μια νέα εντολή. Με το τρόπο αυτό μπορούν να βρεθούν εντολές που διαρκούν τόσο λίγο ώστε να μην είναι δυνατόν να αποτυπωθούν στο γράφημα με την αρχική κλίμακα χρόνου. Οι εντολές αυτές μπορούν να εμφανιστούν μέσω της δυνατότητας zoom. Κάνοντας **drag** το κέρσορα του ποντικιού στο γράφημα



Εικόνα 6: Zoom και Event flags

(αριστερό κρατημένο κλικ και επιλογή περιοχής), γίνεται zoom στην επιλεγμένη περιοχή στον άξονα x Στην *Εικόνα 6*, όπου έχει γίνει zoom στο αρχικό γράφημα, έχουν απενεργοποιηθεί οι γραμμές επικοινωνίας, έχουν ενεργοποιηθεί οι σηματοδότες των εντολών (μικρές πράσινες σημαίες) και έχει σημειωθεί σηματοδότης σε περιοχή όπου δεν φαίνεται να υπάρχει εντολή. Με περεταίρω zoom στη συγκεκριμένη περιοχή, κάνοντας **Ctrl+drag** ώστε να γίνει επιλογή περιοχής και στους δύο άξονες (*Εικόνα 7*),



Εικόνα 7: Zoom και στους δύο άξονες

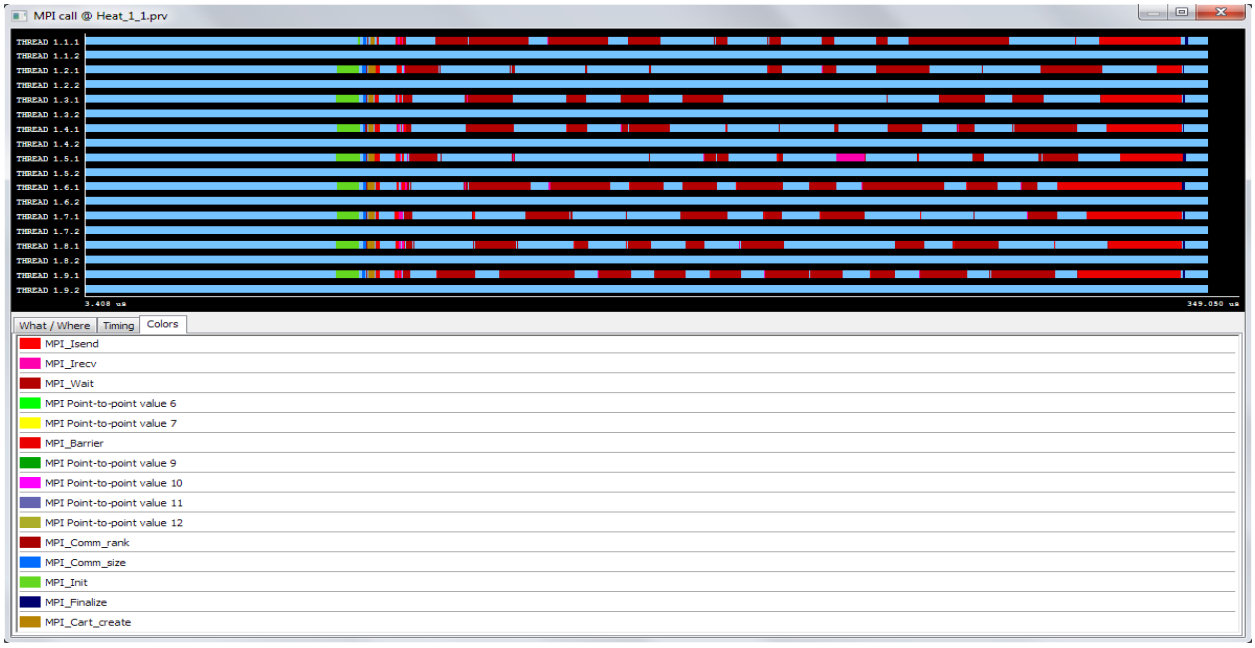
μπορεί να παρατηρηθεί ότι όντος υπάρχει εντολή. Με δεξί κλικ και επιλέγοντας **Undo Zoom** αναιρείται το τελευταίο zoom που έχει γίνει στο γράφημα. Με διαδοχικά Undo Zoom η εικόνα επιστρέφει στην αρχική, όπου φαίνεται όλη η εξέλιξη του προγράμματος.

Στο αρχικό γράφημα δεν παρουσιάζονται όλες οι πληροφορίες, που υπάρχουν στο αρχείο trace, για το πρόγραμμα. Με τη χρήση των configuration files τροποποιείται κατάλληλα το γράφημα ώστε να παρουσιάζονται διαφορετικές πληροφορίες. Υπάρχει πληθώρα έτοιμων τέτοιων αρχείων. Η φόρτωση ενός configuration file γίνεται πατώντας **File** (*Εικόνα 1*) και **Load configuration...** . Στο νέο παράθυρο που εμφανίζεται γίνεται η επιλογή του επιθυμητού cfg. Τα cfgs Βρίσκονται στο φάκελο `cfgs`, στο φάκελο εγκατάστασης του Paraver. Τα cfgs είναι χωρισμένα σε κατηγορίες, αρχικά ανάλογα με το είδος του προγράμματος και έπειτα ανάλογα με το είδος των πληροφοριών που παρέχουν. Μετά τη φόρτωση του εκάστοτε cfg πρέπει να γίνει έλεγχος, αν το γράφημα αναφέρεται στο επίπεδο εφαρμογής, διεργασίας ή νήματος και να γίνεται η ανάλογη αλλαγή όπου είναι απαραίτητο (όπως φαίνεται και στην *Εικόνα 4*). Μερικές φορές μετά τη φόρτωση ενός cfg παρουσιάζεται κάτω και αριστερά στο γράφημα ένα κόκκινο θαυμαστικό. Κάνοντας αριστερό κλικ πάνω του εμφανίζεται ένα σχετικό μήνυμα. Επί το πλείστον το μήνυμα αυτό είναι “Some semantic values are outside the maximum or minimum boundaries”. Στη περίπτωση αυτή το πρόβλημα διορθώνεται πατώντας δεξί κλικ στο γράφημα και επιλέγοντας μία από τις επιλογές στο menu **Fit Semantic Scale**, συνήθως χρειάζεται η **Fit Both**. Επίσης αν για κάποιο cfg το γράφημα είναι κενό, τότε κάνοντας δεξί κλικ στο γράφημα και επιλέγοντας **Fit Time Scale** μπορεί να διορθώσει την εικόνα. Παρακάτω επεξηγούνται τα χρησιμότερα cfgs γα το MPI και το OpenMP. Περαιτέρω ανάλυση έχει γίνει στα cfgs που παρουσιάζουν ιδιαίτερο ενδιαφέρον.

3.2.1 Configuration files για το MPI

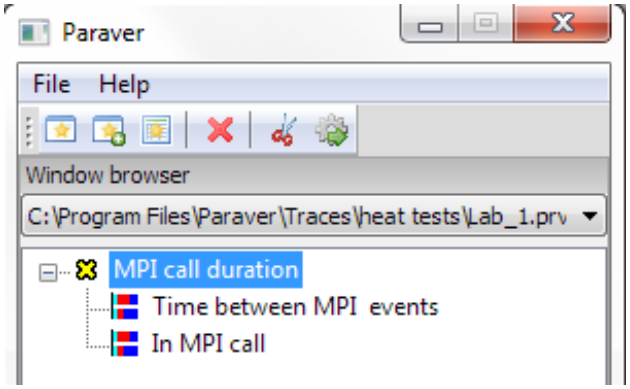
Φάκελος mpi/views

MPI_call.cfg: Τα χρώματα στο γράφημα (*Εικόνα 8*) αντιστοιχούν στις MPI εντολές, οπότε γίνεται ξεκάθαρο σε ποιό σημείο ακριβώς έγινε η κάθε κλήση και πόσο διήρκεσε. Τα χρώματα τα οποία δεν έχουν συγκεκριμένο όνομα εντολής αντιστοιχούν σε εντολές MPI οι οποίες δεν χρησιμοποιούνται και δεν εμφανίζονται στο γράφημα.



Εικόνα 8: MPI_call.cfg

MPI_call_duration.cfg: Παρουσιάζονται οι χρόνοι διάρκειας των εντολών του MPI. Το συγκεκριμένο γράφημα έχει δημιουργηθεί από το συνδυασμό δύο άλλων. Στο κεντρικό παράθυρο του Paraver, κάνοντας expand από το αρχικό cfg (*Εικόνα 9*) εμφανίζονται τα cfgs τα οποία χρησιμοποιήθηκαν ως βάση και με διπλό κλικ σε αυτά εμφανίζεται το αντίστοιχο γράφημα. Στη συγκεκριμένη περίπτωση έχει χρησιμοποιηθεί το cfg που παρουσιάζει το χρόνο που μεσολαβεί μεταξύ των MPI εντολών και του cfg που παρουσιάζει δύο καταστάσεις του προγράμματος, αν εκτελεί MPI εντολή ή όχι, με χρήση 2 χρωμάτων



Εικόνα 9: Εξαρτήσεις των cfgs

node_bandwidth.cfg: Παρουσιάζεται το συνολικό εύρος ζώνης. Προκύπτει από τα cfgs του εύρους ζώνης αποστολής και του εύρους ζώνης λήψης.

Φάκελος mpi/views/advanced

bytes_sr_within_call.cfg: Παρουσιάζονται τα bytes που στάλθηκαν και ελήφθησαν στις επικοινωνίες μεταξύ των διεργασιών.

bytesperMBS.cfg: Παρουσιάζεται ο λόγος των bytes προς το εύρος ζώνης. Στην ουσία είναι ο χρόνος που απαιτείται για την αποστολή των δεδομένων.

in_MPI_call.cfg: Παρουσιάζεται αν εκτελεί το πρόγραμμα MPI εντολή με χρήση 2 χρωμάτων.

long_MPI_calls.cfg: Παρουσιάζονται εντολές MPI που διαρκούν περισσότερο από 1 millisecond. Ανοίγοντας το cfg με ένα επεξεργαστή κειμένου και μεταβάλλοντας τις τιμές “Is in range” στο τμήμα του αρχείου “Long bursts” μεταβάλλεται ο χρόνος για τον οποίο

γίνεται έλεγχος. Συγκεκριμένα πρέπει να γίνει αλλαγή στη τιμή “1000.000000” στη γραμμή “window_semantic_module compose2 Is In Range”.

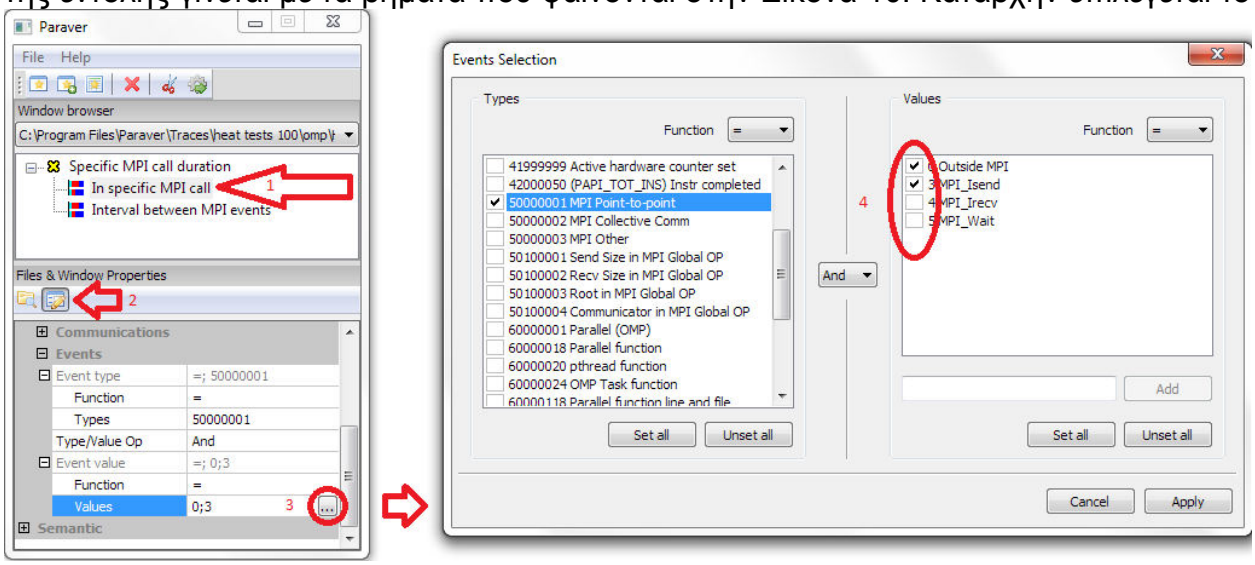
messages_arriving.cfg: Παρουσιάζεται ο αριθμός των μηνυμάτων που πρόκειται να λάβει κάθε διεργασία. Μπορεί να χρησιμοποιηθεί ως προσέγγιση του μεγέθους της ουράς από μηνύματα προς λήψη.

messages_outgoing.cfg: Παρουσιάζεται ο αριθμός των μηνυμάτων που έχουν αποσταλεί από κάθε διεργασία και βρίσκονται στο στάδιο της μεταφοράς.

MPI_collectives.cfg: Παρουσιάζονται οι collective MPI εντολές, δηλαδή εντολές που συμπεριλαμβάνουν όλες τις διεργασίες στη λειτουργία τους (π.χ. Barrier, Reduce, Broadcast, Scatter, Gather κ.τ.λ.)

MPIcall_cost.cfg: Παρουσιάζονται τα bytes των μηνυμάτων που είτε στάλθηκαν είτε ελήφθησαν κατά τη κλήση μίας MPI εντολής προς το συνολικό χρόνο διάρκειας της εντολής. Δηλαδή το κόστος που είχε η κλήση της εντολής σε εύρος ζώνης.

specific_MPI_duration.cfg: Παρουσιάζεται η διάρκεια συγκεκριμένης MPI εντολής, που επιλέγει ο χρήστης, κάθε φορά που αυτή καλείται από το πρόγραμμα. Η επιλογή της εντολής γίνεται με τα βήματα που φαίνονται στην Εικόνα 10. Καταρχήν επιλέγεται το



Εικόνα 10: Διαδικασία επιλογής εντολης για το specific_MPI_duration.cfg

cfg “In specific MPI call”, που είναι ένα από τα δύο από τα οποία πηγάζει το “specific_MPI_call_duration”. Μετά πατώντας το κουμπί που φαίνεται στο κόκκινο κύκλο εμφανίζεται το παράθυρο από το οποίο γίνεται η επιλογή της εντολής.

sr_msgs.cfg: Συνολικός αριθμός μηνυμάτων ανά πάσα στιγμή στο σύστημα.

total_sr_bw.cfg: Συνολικό εύρος ζώνης στις αποστολές και λήψεις μηνυμάτων μεταξύ των διεργασιών.

total_sr_msgs.cfg: Συνολικός αριθμός μηνυμάτων που αντάλλαξαν οι διεργασίες.

total_system_bw.cfg: Συνολικό εύρος ζώνης που χρησιμοποιήθηκε από το σύστημα.

typeof_MPI_Wait.cfg: Παρουσιάζεται, βάση χρώματος, με τι είδους εντολή σχετίζεται η εντολή MPI_Wait.

Φάκελος mpi/views/collectives

Εδώ είναι συγκεντρωμένα τα cfgs τα οποία παρουσιάζουν πληροφορίες σχετικά με τις collective MPI εντολές, η οποίες συμπεριλαμβάνουν όλες τις διεργασίες. Δηλαδή

εντολές όπως το Barrier, το Reduce, το Broadcast, το Scatter το Gather κ.τ.λ. Οι πληροφορίες που παρέχουν τα cfgs είναι παρόμοιες με αυτές που έχουν παρουσιαστεί προηγουμένως και η χρήση κάθε cfg μπορεί να γίνει αντιληπτή από το όνομά του.

Φάκελος mpi/views/point2point

Εδώ είναι συγκεντρωμένα τα cfgs τα οποία παρουσιάζουν πληροφορίες σχετικά με την επικοινωνία των διεργασιών με τις υπόλοιπες, ανά ζεύγη. Οι πληροφορίες που παρέχουν τα cfgs είναι παρόμοιες με αυτές που έχουν παρουσιαστεί προηγουμένως και η χρήση κάθε cfg μπορεί να γίνει αντιληπτή από το όνομά του.

Φάκελος mpi/views/point2point/models

linear_model.cfg: Παρουσιάζεται, βάση χρώματος, ο χρόνος που απαιτείται θεωρητικά για την αποστολή των μηνυμάτων μεταξύ των διεργασιών, βάση του τύπου $T = L + \frac{size}{BW}$ όπου L είναι η καθυστέρηση του συστήματος (latency), $size$ είναι το μέγεθος του μηνύματος και BW το εύρος ζώνης.

Φάκελος mpi/sanity_checks

cloged_system.cfg: Παρουσιάζεται ο λόγος των ανταλλασσόμενων μηνυμάτων προς το εύρος ζώνης. Μεγάλες τιμές του λόγου αντιστοιχούν σε καθυστέρηση στο σύστημα.

Φάκελος mpi/analysis

Στο φάκελο analysis τα cfgs εκτός από τα γραφήματα εμφανίζουν και ένα πίνακα με τη διανομή στις διεργασίες των εκάστοτε τιμών προς μέτρηση. Τα διαγράμματα αυτά μπορούν να δώσουν γρήγορα μια εικόνα για το που παρουσιάζεται πρόβλημα στο σύστημα (π.χ. μεγάλες τιμές χρόνου σε διάρκεια για την εντολή MPI_Wait) και σε ποιες διεργασίες συμβαίνει αυτό. Στη κορυφή κάθε πίνακα υπάρχουν οι παρακάτω επιλογές:


Εμφανίζουν τα γράφημα από τα οποία προκύπτει ο πίνακας.


Εναλλαγή μεταξύ της γενικής εικόνας και των επιμέρους στηλών με τις τιμές για κάθε διεργασία.


Από τη γενική εικόνα του πίνακα γίνεται επιλογή στηλών και παρουσιάζεται το γράφημα για τις στήλες αυτές.


Αφαίρεση των χρωμάτων σχετικών με το εύρος των τιμών.


Εναλλαγή γραμμών και στηλών.


Απόκρυψη των κενών στηλών.


Προσθήκη χρωμάτων στις ονομασίες των στηλών.

2d_collective_duration.cfg: Παρουσιάζεται ο συνολικός χρόνος των collective MPI εντολών (Reduce, Scatter, Gather κ.τ.λ.).

3dh_duration_MPIcall.cfg: Παρουσιάζεται ο συνολικός χρόνος κλήσης για όλες τις MPI εντολές για κάθε διεργασία.

avg_netbw.cfg: Παρουσιάζεται κατά μέσο όρο το εύρος ζώνης (MB/s) κάθε διεργασίας.

connectivity.cfg: Παρουσιάζεται η επικοινωνία μεταξύ των διεργασιών. Ποια διεργασία επικοινώνησε με ποιά.

mpi_stats.cfg: Παρουσιάζεται ο συνολικός χρόνος για κάθε MPI εντολή για κάθε διεργασία.

total_MPI_activity.cfg: Παρουσιάζεται ο συνολικός αριθμός κλήσεων για κάθε MPI εντολή για κάθε διεργασία.

uf_fraction_of_MPI.cfg: Παρουσιάζεται για κάθε διεργασία ο λόγος του χρόνου στον οποίο εκτελεί MPI εντολές προς το συνολικό χρόνο. Η τιμή του λόγου είναι πάντα μεταξύ 0 και 1.

Φάκελος mpi/analysis/advanced

2dc_e2ebw_bytes.cfg: Παρουσιάζεται ο συνολικός αριθμός bytes που έστειλε ή δέχτηκε κάθε εργασία στο αντίστοιχο εύρος ζώνης που αναφέρεται στον οριζόντιο άξονα.

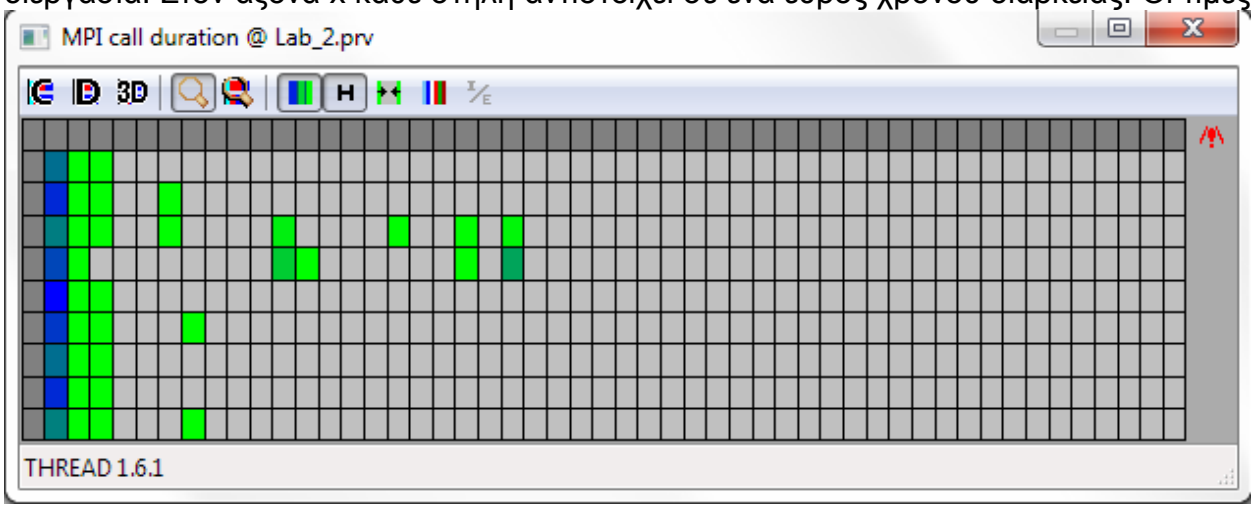
3dc_p2p_size_bw_per_call.cfg: Παρουσιάζεται κατά μέσο όρο ο λόγος του εύρους ζώνης προς το μέγεθος του μηνύματος για κάθε διεργασία.

3dc_size_bw_per_call.cfg: Παρουσιάζεται κατά μέσο όρο ο λόγος του συλλογικού εύρους ζώνης προς το μέγεθος των μηνυμάτων από τις collective MPI εντολές (Reduce, Scatter, Gather κ.τ.λ.)

Φάκελος mpi/analysis/other

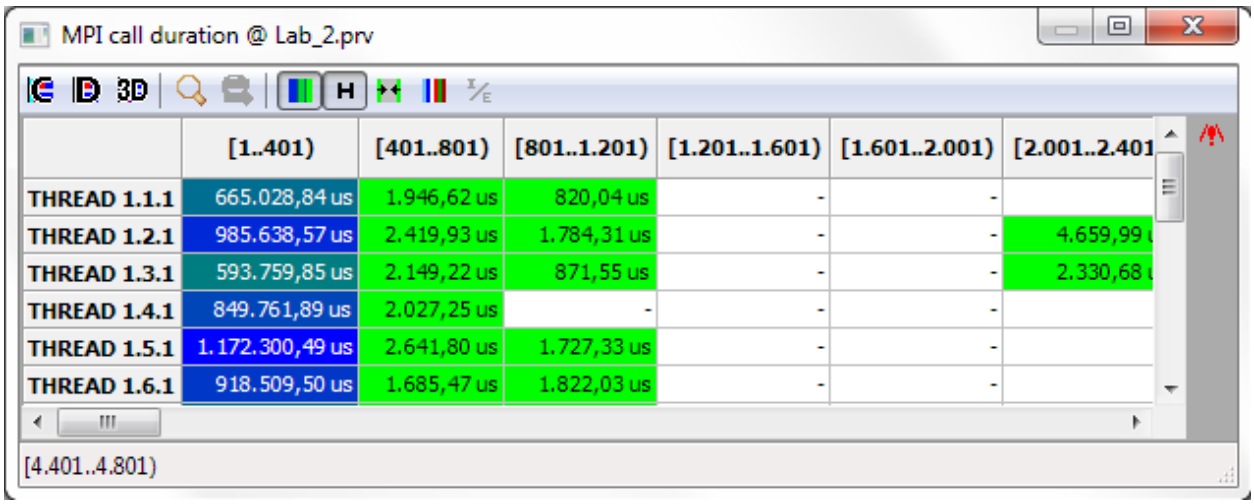
3dc_bw_per_call.cfg: Παρουσιάζεται το ενεργό εύρος ζώνης που φαίνεται να έχει στη διάθεσή της κάθε διεργασία.

3dh_duration_per_call.cfg: Παρουσιάζεται η διάρκεια των MPI εντολών για κάθε διεργασία. Στον άξονα x κάθε στήλη αντιστοιχεί σε ένα εύρος χρόνου διάρκειας. Οι τιμές



Εικόνα 11: 3dh_duration_per_call.cfg, γενική εικόνα

του πίνακα σε κάθε κελί αντιστοιχούν στο συνολικό χρόνο διάρκειας των επιμέρους εντολών που έχουν διάρκεια που εντάσσεται στο εύρος της στήλης που ανήκουν. Έτσι διαιρώντας τη τιμή του κελιού με τη μέση τιμή του εύρους της στήλης βρίσκουμε τον αριθμό των εντολών με το συγκεκριμένο εύρος. Χρησιμοποιώντας το κουμπί , η εικόνα αλλάζει από αυτή τις *Εικόνας 11* σε αυτή της *Εικόνας 12*, όπου φαίνονται αναλυτικά οι τιμές των κελιών.



Εικόνα 12: 3dh_duration_per_call.cfg, αναλυτικά οι τιμές

CommComp_overlap.cfg: Παρουσιάζεται, σε σχέση με τον αριθμό των μηνυμάτων προς λήψη ή που έχουν αποσταλεί για τη κάθε διεργασία, ο λόγος τις διάρκειας των υπολογισμών που εκτελεί η διεργασία προς το χρόνο που περιμένει για να λάβει ή να στείλει τα σχετικά μηνύματα. Μικρές τιμές στο πίνακα σημαίνουν ότι η διεργασία τελειώνει τους υπολογισμούς και περιμένει σε απραξία να ολοκληρωθεί η διαδικασία λήψης ή αποστολής μηνυμάτων.

System_BW.cfg: παρουσιάζεται ανά εύρος τιμών του εύρους ζώνης το ποσοστό των επικοινωνιών που αντιστοιχούν στο εύρος τιμών αυτό.

3.2.2 Configuration files για το OpenMP

Φάκελος OpenMP/views

parallel_functions.cfg: Παρουσιάζονται οι συναρτήσεις του προγράμματος που κάνουν χρήση OpenMP. Οι συναρτήσεις μπορεί να περιέχουν και εντολές που δεν είναι OpenMP.

parallel_functions_duration.cfg: Παρουσιάζεται η διάρκεια των συναρτήσεων που κάνουν χρήση OpenMP. Ο χρόνος αυτός δεν αντιστοιχεί μόνο στο χρόνο που απαιτείται για τους υπολογισμούς με χρήση OpenMP, αλλά στη διάρκεια που χρειάζεται για να εκτελεστεί η συνάρτηση που περιέχει τους υπολογισμούς αυτούς.

Φάκελος OpenMP/analysis

Amdahl.cfg: Παρουσιάζεται το ποσοστό του σειριακού τμήματος του προγράμματος, για κάθε συνάρτηση. Όσο μικρότερο είναι το νούμερο τόσο το καλύτερο, διότι υπάρχει μεγαλύτερη δυνατότητα για κλιμάκωση της συνάρτησης. Χρησιμοποιώντας το νούμερο αυτό μπορεί να βρεθεί με χρήση του νόμου του Amdahl η θεωρητική επιτάχυνση του προγράμματος. Ο νόμος του Amdahl είναι $S(n)=\frac{T(1)}{T(n)}$, όπου $T(n)$ είναι ο χρόνος εκτέλεσης με χρήση n νημάτων και $T(1)$ ο σειριακός χρόνος, οι οποίοι σχετίζονται με τη σχέση $T(n)=T(1)(B+\frac{1}{n}(1-B))$, όπου B είναι το ποσοστό του σειριακού τμήματος του προγράμματος ή του κώδικα προς εξέταση.

3.2.3 Γενικά Configuration files

Φάκελος General/views

instantaneous_parallelism.cfg: Παρουσιάζει τον αριθμό τον ενεργών διεργασιών κάθε χρονική στιγμή.

state_as_is.cfg: Είναι το γράφημα το οποίο παρουσιάζεται με το κουμπί της *Εικόνας 3*.

useful.cfg: Παρουσιάζεται με δύο χρώματα αν τα νήματα είναι ενεργά ή όχι.

3.3 Χρήση

Για να γίνει καλύτερα αντιληπτό πως μπορεί το Paraver να βοηθήσει στη βελτίωση ενός προγράμματος θα εξεταστούν διάφορες καταστάσεις παράλληλων προγραμμάτων και θα γίνει σύγκριση μεταξύ διαφορετικών υλοποιήσεων. Αρχικά θα γίνει σύγκριση μεταξύ υλοποιήσεων ενός μοντέλου διάχυσης θερμότητας και στη συνέχεια ενός προγράμματος υπολογισμού εμβαδού με τη μέθοδο των τραπεζίων.

3.3.1 Πρόγραμμα προσομοίωσης μετάδοσης θερμότητας

Στόχος της εφαρμογής είναι η προσομοίωση μεταφοράς θερμότητας. Στην ουσία έχουμε ένα πίνακα MxN διαστάσεων όπου κάθε στοιχείο του πίνακα αποτελεί ένα σημείο του χώρου που έχει μια συγκεκριμένη θερμοκρασία. Το σύστημα αλλάζει κατάσταση με την πάροδο του χρόνου. Δηλαδή κάθε σημείο επηρεάζεται από τα γειτονικά του. Γειτονικά στοιχεία θεωρούνται αυτά που βρίσκονται πάνω, κάτω, δεξιά και αριστερά σε ένα στοιχείο. Τα στοιχεία που βρίσκονται στα άκρα δεν αλλάζουν τιμές καθώς θεωρούμε ότι είναι στοιχεία που απορροφούν ή εκπέμπουν θερμότητα στο σύστημα.

3.3.1.1 Υλοποίηση 1η

Περιγραφή Παράλληλου Αλγορίθμου

1. Δέσμευση τοπικών δισδιάστατων πινάκων για το αντίστοιχο τμήμα του ολικού δισδιάστατου πίνακα. Για κάθε διεργασία είναι 2 πίνακες: ένας για το τρέχον βήμα και ένας για το επόμενο, όπου στη συνέχεια εναλλάσσονται, αφού γίνει το update.

2. Υπολογισμός των γειτονικών διεργασιών στην εικονική δισδιάστατη τοπολογία που βρίσκονται, όπως και διάφορων άλλων μεταβλητών που χρειάζονται για την αποδοτικότερη λειτουργία του προγράμματος.

3. Αρχικοποίηση του τοπικού πίνακα βάσει των συντεταγμένων του block που αναλαμβάνει για update, με τις ίδιες-αντίστοιχες τιμές του προγράμματος που μας δίνεται.

4. Για κάθε βήμα επαναλαμβάνεται η εξής διαδικασία: Αποστολή και λήψη μηνυμάτων σε non-blocking mode μέσω των MPI_Isend και MPI_Irecv για όσους γείτονες έχει (εφόσον δεν είναι περιοδική η ανταλλαγή). Αναμονή των κατάλληλων μηνυμάτων για τη σωστή διεκπεραίωση της update. Εκτέλεση της update. Αναμονή για την αποστολή των μηνυμάτων. Ανταλλαγή των 2 τοπικών πινάκων (swap).

5. Εκτύπωση μηνυμάτων και αποδέσμευση πόρων.

Εικονική Τοπολογία διεργασιών

Χρησιμοποιήσαμε την καρτεσιανή δισδιάστατη εικονική τοπολογία που προσφέρει το mpi, εφόσον φαινόταν η πλέον κατάλληλη για τα δεδομένα του προβλήματος. Μέσω των συντεταγμένων που έχει η κάθε διεργασία στον εικονικό χώρο, αναλαμβάνει και το αντίστοιχο block του ολικού πίνακα. Ο καθορισμός της επικοινωνίας με τους 4 (μέγιστο) γείτονες γίνεται βολικά μέσω της MPI_Cart_shift χωρίς περιοδικότητα.

Κατανομή Δεδομένων – Ισοκατανομή Φόρτου

Κάθε διεργασία αναλαμβάνει την ανανέωση ενός δισδιάστατου τμήματος (block) του συνολικού πίνακα. Ο συνολικός πίνακας είναι νοητός, καθώς σε κάθε διεργασία δεσμεύεται μόνο το αντίστοιχο τμήμα με ίδιο μέγεθος. Κάθε τμήμα έχει διαστάσεις x = nxprob/sqrt(total_processes) + 2 και y = nyprob/sqrt(total_processes) + 2, όπου το (+2) υπάρχει για την ανταλλαγή των δεδομένων. Επομένως, η κατανομή των δεδομένων και του φόρτου είναι ισάξια για όλες τις διεργασίες.

Στρατηγική Επικοινωνίας

Κάθε διεργασία επικοινωνεί με 2-4 γείτονες (πάνω, κάτω, δεξιά, αριστερά) εφόσον δεν είναι περιοδική η επικοινωνία. Για αποφυγή αδιεξόδου, αλλά και ταχύτερη εκτέλεση χρησιμοποιήθηκαν non-blocking standard mode συναρτήσεις. Τα βήματα της εποικοινωνία είναι τα εξής:

1. Για κάθε κατεύθυνση ελέγχεται αν υπάρχει γείτονας. Αν υπάρχει αποστολή και λήψη του αντίστοιχου μηνύματος με τις MPI_Isend,MPI_Irecv.

2. Αναμονή για την ολοκλήρωση λήψης μέσω της MPI_Wait, για να γίνει σωστά το update.

3. Αναμονή για την ολοκλήρωση αποστολής (αφού γίνει πρώτα το update).

Τα μηνύματα πάνω-κάτω είναι της μορφής MPI_FLOAT και μέγεθος όσο η οριζόντια διάσταση του block. Ενώ τα μηνύματα αριστερά-δεξιά είναι της μορφής column_t και μέγεθος 1. Μία πιθανή βελτίωση για ταχύτερη διαδικασία είναι η αποφυγή του ελέγχου αν υπάρχει γείτονας προς κάποια κατεύθυνση. Αυτό μπορεί να γίνει είτε μέσω της περιοδικότητας, όπου θα στέλνονται περιοδικά μηνύματα, αλλά θα είναι μηδενικά, είτε μέσω μίας κατάλληλης δομής, όπου θα υπάρχουν για κάθε διεργασία μόνο οι γείτονες που έχει.

3.3.1.2 Υλοποίηση 2η

Partitioning

Το partitioning μπορεί να γίνει σε αυτή την περίπτωση μόνο ως προς τα δεδομένα καθώς δεν επιτελείται καμιά διαφορετική διεργασία. Συγκεκριμένα με διάφορους τρόπους μπορούμε να αποφασίσουμε πως θα διαχωρίσουμε τα δεδομένα του πίνακα. Εφόσον ο φόρτος επεξεργασίας είναι ίδιος για όλο τον πίνακα δεν υπάρχει λόγος να γίνει κάποιος κυκλικός διαμοιρασμός. Οπότε τον πίνακα θα τον χωρίσουμε είτε σε λωρίδες (οριζόντιες ή κάθετες) είτε σε μπλοκ.

Communication

Όπως έχουμε αναφέρει, η τιμή ενός στοιχείου εξαρτάται από τα τέσσερα γειτονικά του. Άρα σε περίπτωση διαχωρισμού του πίνακα, οι διεργασίες πρέπει να ανταλλάσσουν πληροφορίες των στοιχείων που ανήκουν σε «γειτονικές» διεργασίες. Αυτές θα είναι είτε οι ακραίες γραμμές είτε οι ακραίες στήλες στις περιπτώσεις διαχωρισμού σε οριζόντιες λωρίδες και σε κάθετες λωρίδες αντίστοιχα. Ενώ στην περίπτωση διαχωρισμού σε μπλοκ θα είναι όλα τα ακραία στοιχεία. Θεωρούμε ότι η επικοινωνία μεταξύ των

υπολογιστών είναι ίδιων τεχνικών χαρακτηριστικών και ότι δεν υπάρχει επικοινωνιακός φόρτος απ' άλλες εφαρμογές.

Υλοποίηση MPI

Για την υλοποίηση χρησιμοποιήθηκε ο σχεδιασμός που περιγράφηκε πιο πάνω. Το πρόγραμμα σχεδιάστηκε έτσι ώστε να δέχεται οποιοδήποτε μέγεθος πίνακα χωρίς να υπάρχει δέσμευση ώστε αυτός να είναι τετράγωνο. Επίσης οι διεργασίες παίρνουν κομμάτι του πίνακα ανάλογα με το πλήθος τους και το ID τους. Συγκεκριμένα βρίσκεται το πάνω ακέραιο μέρος της τετραγωνικής ρίζας του πλήθους τους για να ορίσει τη μεγάλη διάσταση στο mapping και για τη μικρή έχουμε την ακέραια διαίρεση του πλήθους με το μέγεθος της μεγάλης. Γίνεται κατάλληλη προσαρμογή ώστε η μεγάλη διάσταση να συμπίπτει με τη μεγαλύτερη του πίνακα.

Κάθε διεργασία υπολογίζει το μέρος του πίνακα που θα διαχειριστεί, αν έχει γειτονικές διεργασίες και τον επιπλέον χώρο που θα χρειαστεί για την ανταλλαγή των δεδομένων. Δεσμεύει τον κατάλληλο χώρο για δυο πίνακες ώστε να κρατάει την τρέχουσα και την επόμενη κατάσταση. Αρχικοποιεί τον τρέχων πίνακα έτσι ώστε η θερμοκρασία να είναι στη μέση υψηλή και τα άκρα του να παραμένουν όπως και στη δοσμένη υλοποίηση στο μηδέν σε όλη τη διάρκεια της προσομοίωσης.

Στο υπολογιστικό κομμάτι, στην αρχή γίνεται με non-blocking διαδικασία η ανταλλαγή των δεδομένων όπου χρειάζεται. Μέχρι να ολοκληρωθεί η διαδικασία τις ανταλλαγής των δεδομένων, κάθε διεργασία υπολογίζει το ανάλογο κομμάτι στο πίνακα το οποίο δεν εξαρτάται από τα ανταλλασσόμενα δεδομένα. Στη συνέχεια η διεργασία ελέγχει αν τις έχει ολοκληρωθεί οποιαδήποτε αποστολή και λήψη δεδομένων και για την περίπτωση των λήψεων προβαίνει στον υπολογισμό του πίνακα που υπήρχε εξάρτηση με αυτό το κομμάτι των δεδομένων.

Αφού έχει ολοκληρωθεί η διαδικασία για όλες τις χρονικές στιγμές, η κύρια διαδικασία αναλαμβάνει να συγκεντρώσει και να γράψει τα αποτελέσματα σε ένα αρχείο. Κάθε άλλη διεργασία αποστέλλει τις σχετικές πληροφορίες του κομματιού που έχει αναλάβει και τα δεδομένα που έχει. Η κύρια διεργασία αφού δεσμεύσει τον κατάλληλο χώρο για όλο τον πίνακα και αντιγράψει το δικό της κομμάτι, παίρνει τα δεδομένα απ' όλες τις διεργασίες με non-blocking τρόπο. Στο τέλος δημιουργεί ένα αρχείο final.dat όπου απεικονίζει με αριθμούς τις κατά τόπους θερμοκρασίες του υλικού στο τέλος της προσομοίωσης.

Για την περίπτωση ανταλλαγής στηλών μεταξύ των διεργασιών χρησιμοποιήθηκε ένα custom datatype όπου αναπαριστά μια στήλη. Επίσης κάτι ανάλογο έγινε με την αποστολή των τελικών δεδομένων στην κύρια διεργασία όπου χρησιμοποιήθηκε ένα custom datatype που αναπαριστά ένα τετράγωνο. Επίσης για την αποστολή των πληροφοριών των υποπινάκων στην κύρια διεργασία χρησιμοποιείται ένας τύπος για την ταυτόχρονη αποστολή τους.

3.3.1.3 Υλοποίηση $3^{η}$

- Υποθέτουμε πως το πλέγμα είναι πάντα τετράγωνο, και πολλαπλάσιο των αριθμών 2, 3, 4, 5, 6, 7, δηλαδή το μικρότερο δυνατό πλέγμα είναι 420x420. Σε κάθε διάστασή του αντιστοιχεί μία σειρά διεργασιών ίση με κάποιο από τούς παραπάνω αριθμούς (π.χ. 2*2=4, 3*3=6,...). Η επιλογή τους οφείλεται στο πλήθος των διαθέσιμων υπολογιστών/πυρήνων που υπάρχουν στο Εργαστήριο Linux του Τμήματος (περίπου 50).

- Το πλέγμα χωρίζεται σε ίσου μεγέθους τετράγωνα block, πλήθους ίσο με τον αριθμό των διεργασιών, έστω N. Κάθε γραμμή και κάθε στήλη του πλέγματος

αποτελείται από $\sqrt{N}$ τέτοια block, κάθε ένα από τα οποία αντιστοιχίζεται σε μία διεργασία.

- Κάθε block έχει το πολύ τέσσερις γείτονες. Η επικοινωνία μεταξύ τους γίνεται με χρήση των non-blocking συναρτήσεων MPI_Isend και MPI_Irecv.

Σε κάθε βήμα της εκτέλεσης, κάθε διεργασία χρησιμοποιεί τις δύο αυτές συναρτήσεις για να στείλει τα εξωτερικά σημεία κάθε πλευράς της στον αντίστοιχο γείτονα και να λάβει από αυτόν τη δική του εξωτερική πλευρά.

- Στη συνέχεια εκτελείται η συνάρτηση update για όλα τα εσωτερικά της σημεία, τα οποία δεν εξαρτώνται από τις πλευρές των γειτόνων. Με αυτό τον τρόπο επιτυγχάνεται επικάλυψη τον χρόνου της επικοινωνίας με χρήσιμους υπολογισμούς.

- Αφού ολοκληρωθεί η παραπάνω διαδικασία, χρησιμοποιείται η συνάρτηση MPI_Waitall για να μπλοκαριστεί η εκτέλεση του προγράμματος έως ότου ολοκληρωθούν όλες οι ενεργές λήψεις (όχι όμως οι αποστολές οι οποίες δεν επηρεάζουν τα δεδομένα - το πλέγμα είναι τρισδιάστατο με βάθος δύο και έτσι μπορούμε να ανανεώσουμε στη μία πλευρά κάποιο σημείο χωρίς να πειράξουμε το πρωτότυπο).

- Μόλις ληφθούν όλες οι γειτονικές πλευρές εκτελείται πάλι η συνάρτηση update, για κάθε μία από τις εξωτερικές πλευρές του block.

- Τέλος, περιμένουμε και την ολοκλήρωση των ενεργών αποστολών με χρήση της συνάρτησης MPI_Waitall.

- Κάθε διεργασία γεννά μόνη της τα δεδομένα της και δεν στέλνει τα αποτελέσματα πίσω στη master-διεργασία. Στέλνει όμως το συνολικό χρόνο εκτέλεσης της.

- Ο ρόλος της master-διεργασίας είναι να συγκεντρώσει τους συνολικούς χρόνους των άλλων διεργασιών και να εκτυπώσει τον καλύτερο και τον χειρότερο.

- Για τη μέτρηση των χρόνων εκτέλεσης χρησιμοποιείται η συνάρτηση MPI_Wtime.

- Για την αποδοτικότερη επικοινωνία μεταξύ των γειτόνων, έχουν δημιουργηθεί δύο MPI Datatypes, ο τύπος γραμμής και ο τύπος στήλης.

3.3.2 Σύγκριση υλοποιήσεων προγράμματος προσομοίωσης μετάδοσης θερμότητας

Για την δημιουργία των trace αρχείων με το Extrae, έχει γίνει χρήση της μεθόδου με τη μεταβλητή περιβάλλοντος LD_PRELOAD και οι εκτελέσεις των προγραμμάτων πραγματοποιήθηκαν στο εργαστήριο Linux του τμήματος Πληροφορικής και Τηλεπικοινωνιών. Οι παράμετροι εκτέλεσης για κάθε υλοποίηση είναι:

Μέγεθος πίνακα δεδομένων: 840 x 840.

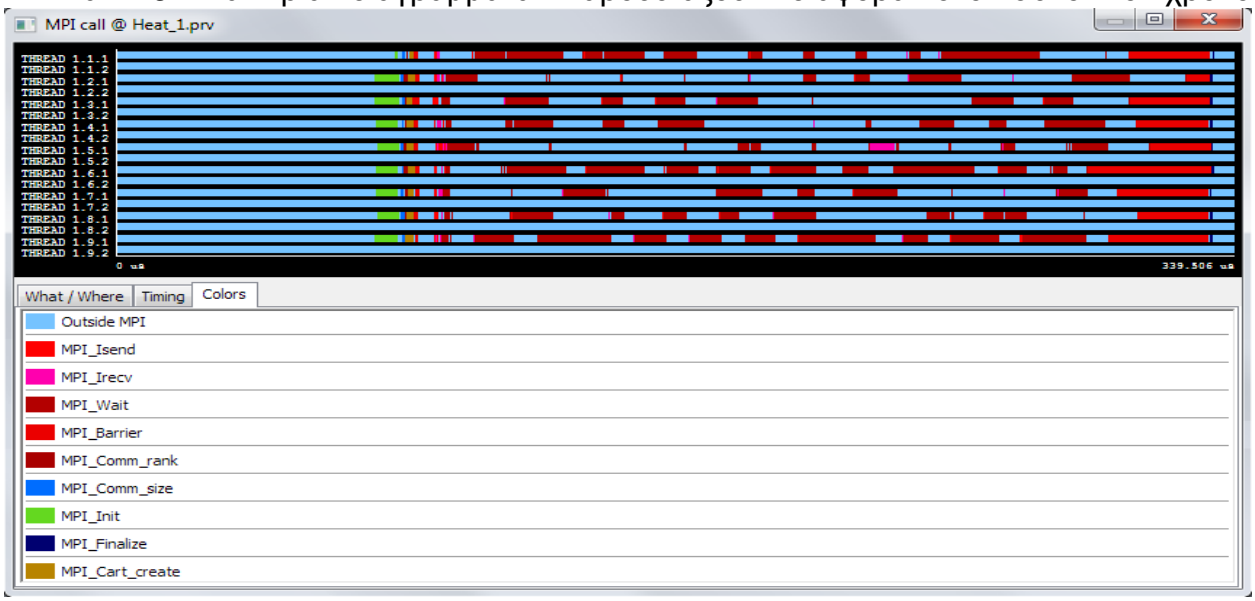
Επαναλήψεις: 10.

Διεργασίες: 9, 2 διεργασίες σε κάθε επεξεργαστή.

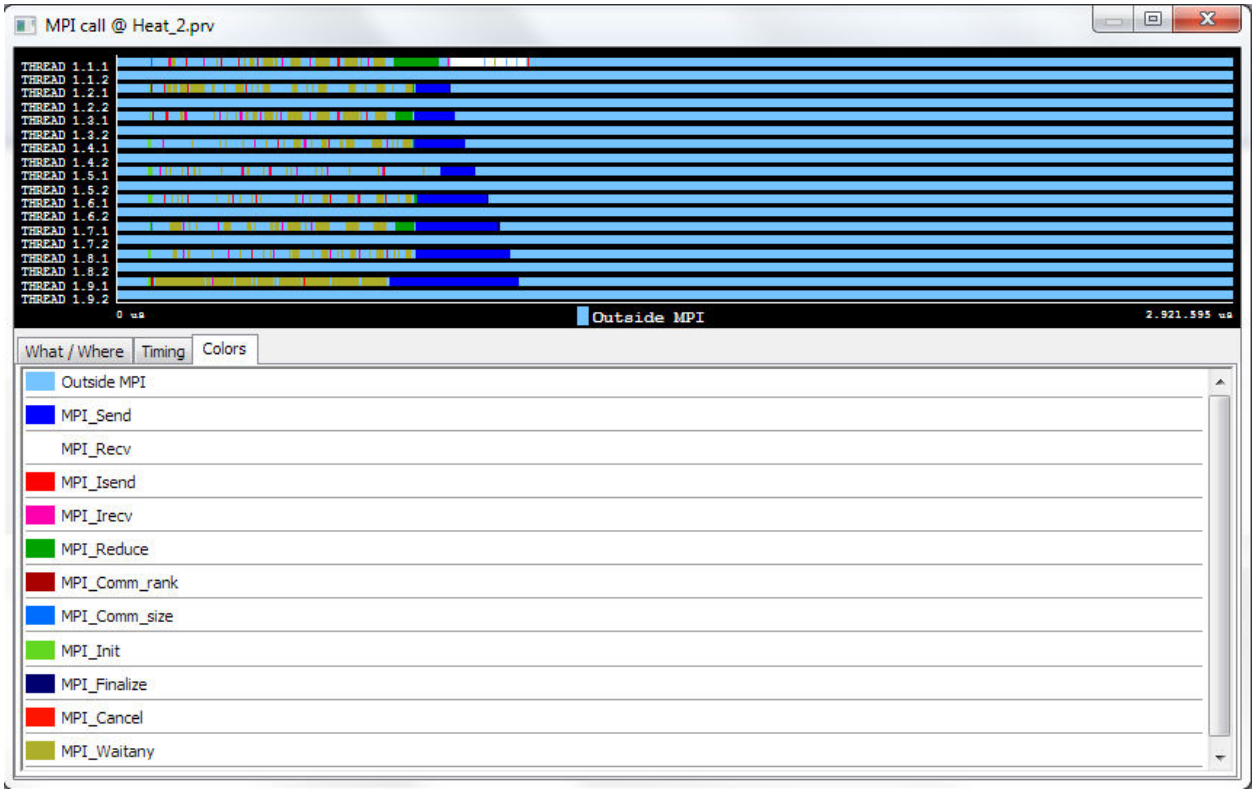
3.3.2.1 Φάκελος *mpi/views*

MPI_call.cfg: Χρησιμοποιώντας το συγκεκριμένο cfg με τις διαφορετικές υλοποιήσεις του μοντέλου διάχυσης θερμότητας δημιουργούνται τα διαγράμματα των *Εικόνων 13*,

14 και *15*. Τα τρία διαγράμματα παρουσιάζουν διαφορά στο συνολικό χρόνο



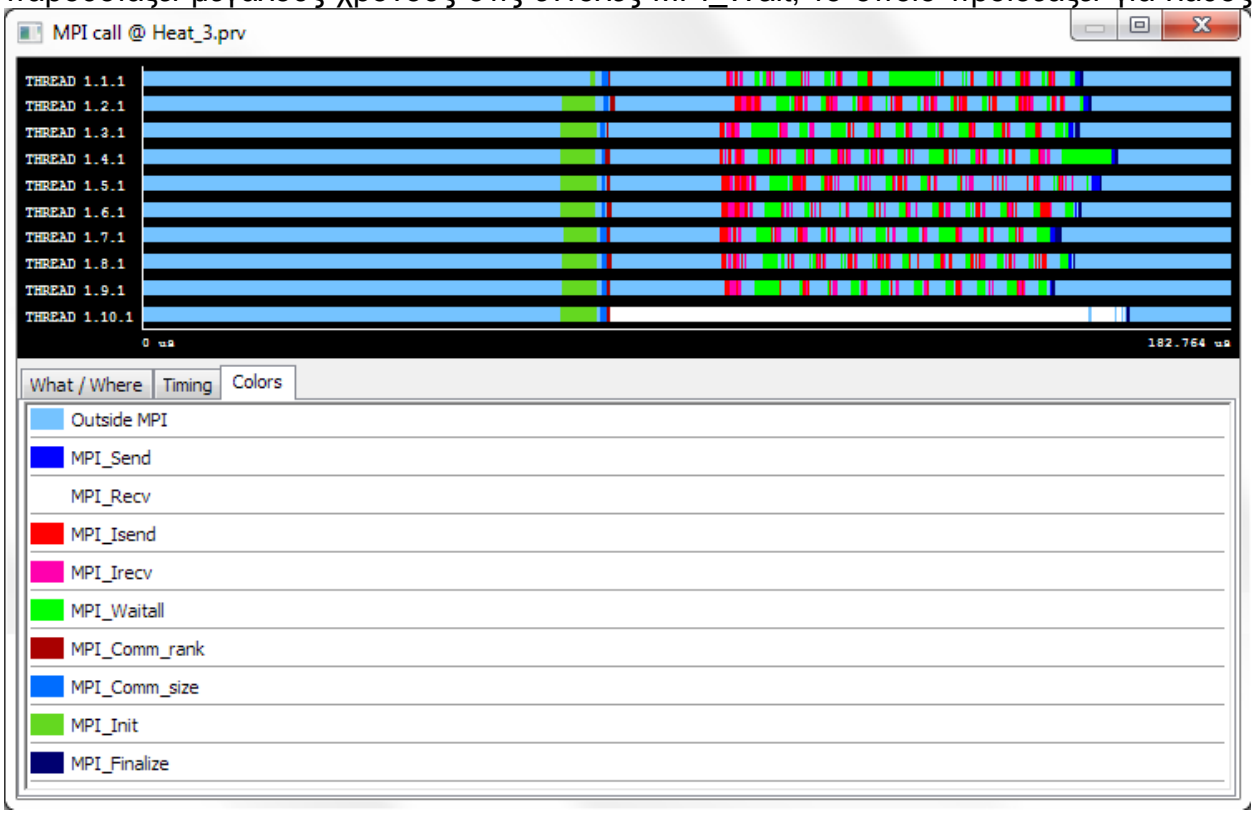
Εικόνα 13: MPI_call.cfg - Υλοποίηση 1η



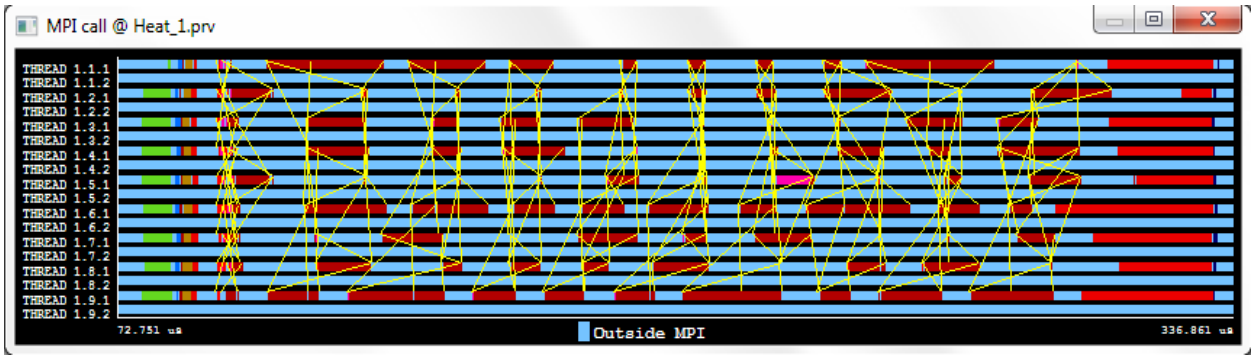
Εικόνα 14: MPI_call.cfg - Υλοποίηση 2η

λειτουργίας. Λαμβάνοντας υπόψη τη κλίμακα στον οριζόντιο άξονα του χρόνου η φαινομενική διαφορά μέχρι το MPI_Init δεν υφίσταται. Η μεγάλη διαφορά για τη δεύτερη υλοποίηση έγκειται στο γεγονός ότι πριν ολοκληρωθεί, δημιουργεί ένα αρχείο με τις τιμές που έχει επεξεργαστεί το πρόγραμμα. Το μέγεθος του αρχείου αυτού είναι τέτοιο ώστε να απαιτείται περίπου ένα δευτερόλεπτο για να δημιουργηθεί. Κάνοντας zoom στις ενεργές περιοχές των προγραμμάτων και ενεργοποιώντας τη γραμμές επικοινωνίας μεταξύ των διεργασιών (*Εικόνες 16*, *17*, *18*), γίνεται φανερό ότι το γενικό μοτίβο λειτουργίας των τριών προγραμμάτων είναι πανομοιότυπο, όπως και είναι αναμενόμενο. Ειδικότερα τώρα, εξετάζοντας κάθε υλοποίηση ξεχωριστά, η πρώτη

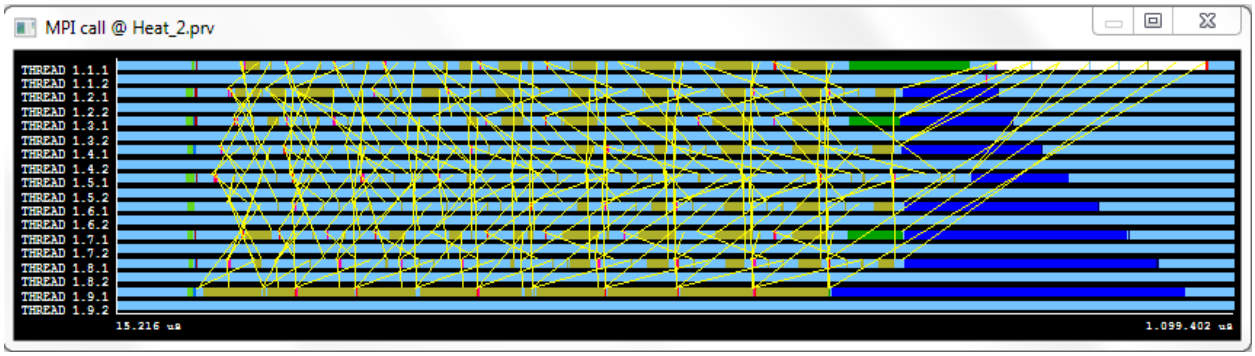
παρουσιάζει μεγάλους χρόνους στις εντολές MPI_Wait, το οποίο προϊδεάζει για λάθος



Εικόνα 15: MPI_call.cfg - Υλοποίηση 3η



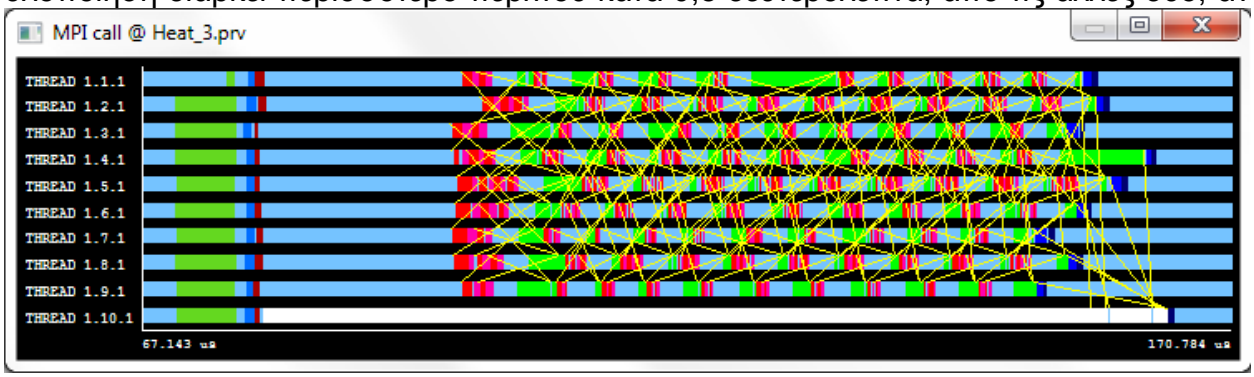
Εικόνα 16: MPI_call.cfg - Υλοποίηση 1η, γραμμές επικοινωνίας



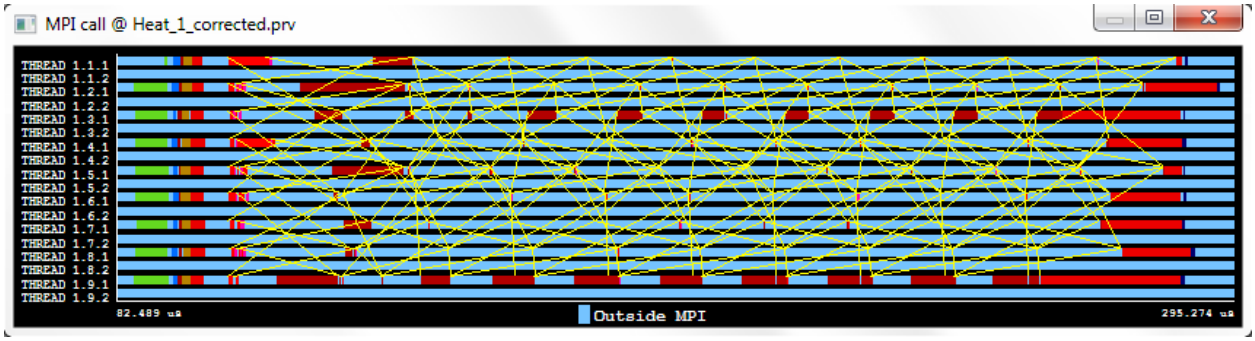
Εικόνα 17: MPI_call.cfg - Υλοποίηση 2η, γραμμές επικοινωνίας

στο κώδικα. Εξετάζοντάς τον αποκαλύπτεται ότι η εντολή MPI_Wait είναι σε λάθος θέση και βρίσκεται πριν από την εντολή υπολογισμών των νέων τιμών του πλέγματος. Μεταφέροντάς την στη σωστή θέση προκύπτει το διάγραμμα της *Εικόνας 19*, με φανερή βελτίωση, αφού έχουν μειωθεί κατά πολύ οι χρόνοι των MPI_Wait. Η δεύτερη

υλοποίηση διαρκεί περισσότερο περίπου κατά 0,8 δευτερόλεπτα, από τις άλλες δύο, αν



Εικόνα 18: MPI_call.cfg - Υλοποίηση 3η, γραμμές επικοινωνίας

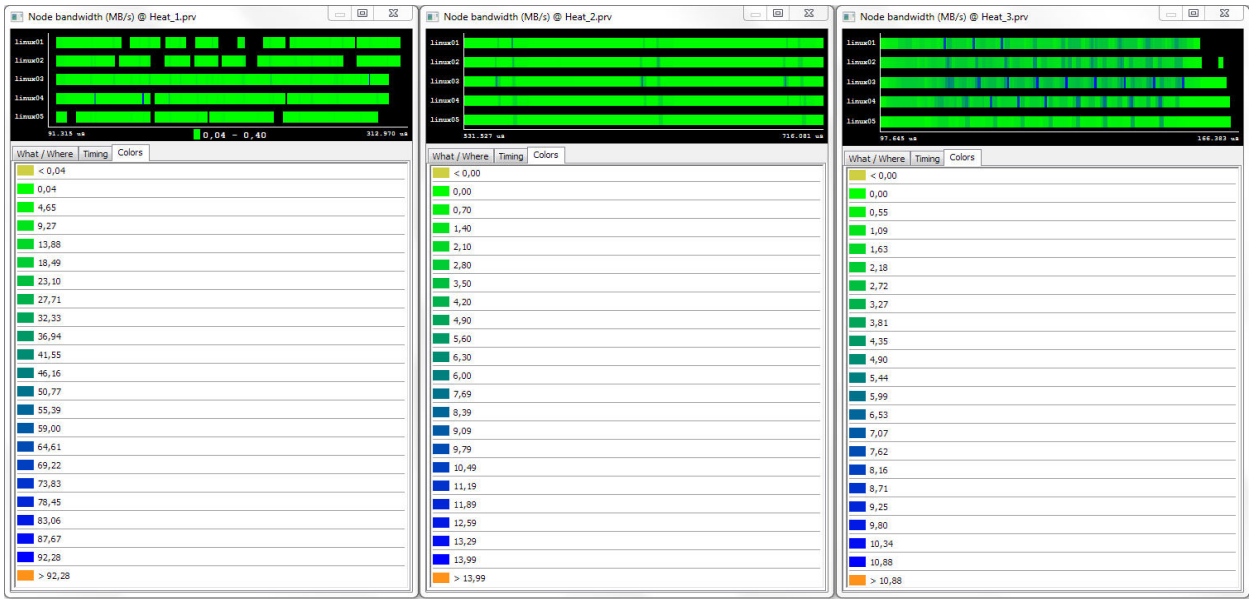


Εικόνα 19: MPI_call.cfg - Υλοποίηση 1η, διορθωμένος κώδικας

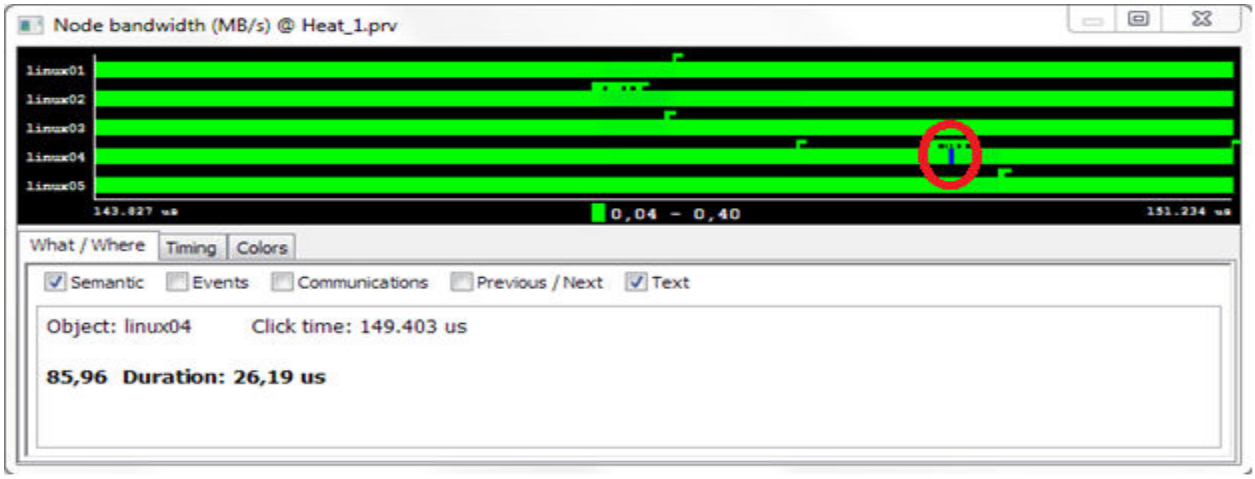
εξαιρέσουμε το χρόνο για τη δημιουργία του αρχείου δεδομένων. Αυτό φαίνεται να οφείλεται στη χρήση της εντολής MPI_Waitany. Εξετάζοντας το κώδικα δεν εντοπίζεται κάποιο λάθος που να δικαιολογεί τους χρόνους που εμφανίζει η MPI_Waitany, αφού οι απαραίτητοι υπολογισμοί γίνονται πριν την εντολή MPI_Waitany, η οποία είναι δομημένη έτσι ώστε να αναγνωρίζει τα request που προέρχονται από MPI_Send με τέτοιο τρόπο, που κάθε διεργασία συνεχίζει με υπολογισμούς σχετικούς με τα δεδομένα που έλαβε. Συμπεραίνεται, οπότε, ότι αυτή είναι η κανονική λειτουργία της εντολής, η οποία δεν είναι αποδοτική. Τέλος εξετάζοντας την τρίτη υλοποίηση παρατηρούνται μικρές καθυστερήσεις στην MPI_Waitall, αλλά εξετάζοντας και σε αυτή τη περίπτωση το κώδικα του προγράμματος δεν υπάρχει κάποιο λάθος στη δομή, οπότε συμπεραίνεται ότι ούτε η MPI_Waitall είναι η καταλληλότερη για χρήση στη συγκεκριμένη περίπτωση. Η διεργασία 10 χρησιμοποιείται μόνο για τη συγκέντρωση των χρόνων εκτέλεσης των υπολοίπων διεργασιών και την εκτύπωση των αποτελεσμάτων, οπότε βρίσκεται επί το πλείστον στη κατάσταση όπου περιμένει να λάβει από τις υπόλοιπες διεργασίες δεδομένα. Επίσης, μία ακόμα παρατήρηση για την τρίτη υλοποίηση είναι ότι δεν γίνεται χρήση OpenMP για τους υπολογισμούς, οπότε και δεν υπάρχουν νήματα παρά μόνο διεργασίες.

node_bandwidth.cfg: Με το συγκεκριμένο cfg τα διαγράμματα που δημιουργούνται για τις τρεις υλοποιήσεις φαίνονται στην *Εικόνα 20*. Τα χρώματα, για κάθε διάγραμμα αντιστοιχούν σε διαφορετικά εύρη τιμών, οπότε είναι απαραίτητες οι πληροφορίες από το info panel για να γίνει η σύγκριση. Η πρώτη υλοποίηση παρουσιάζει πολύ μεγάλη διαφορά στη μέγιστη τιμή εύρους ζώνης, η οποία όμως, εμφανίζεται ελάχιστα στο διάγραμμα. Γενικότερα, και οι τρεις υλοποιήσεις κυμαίνονται περίπου στο ίδιο εύρος τιμών, με διαφορά μόνο μερικών μονάδων. Χρησιμοποιώντας και το MPI_call.cfg μπορεί να γίνει αντιστοίχιση του εύρους ζώνης με τις MPI εντολές. Οι περιοχές με μεγάλες τιμές (μπλε χρώμα), και στις τρεις υλοποιήσεις αντιστοιχούν σε MPI εντολές που έχουν σχέση με αποστολή δεδομένων, όπως MPI_Isend, MPI_Reduce κ.τ.λ., όπως

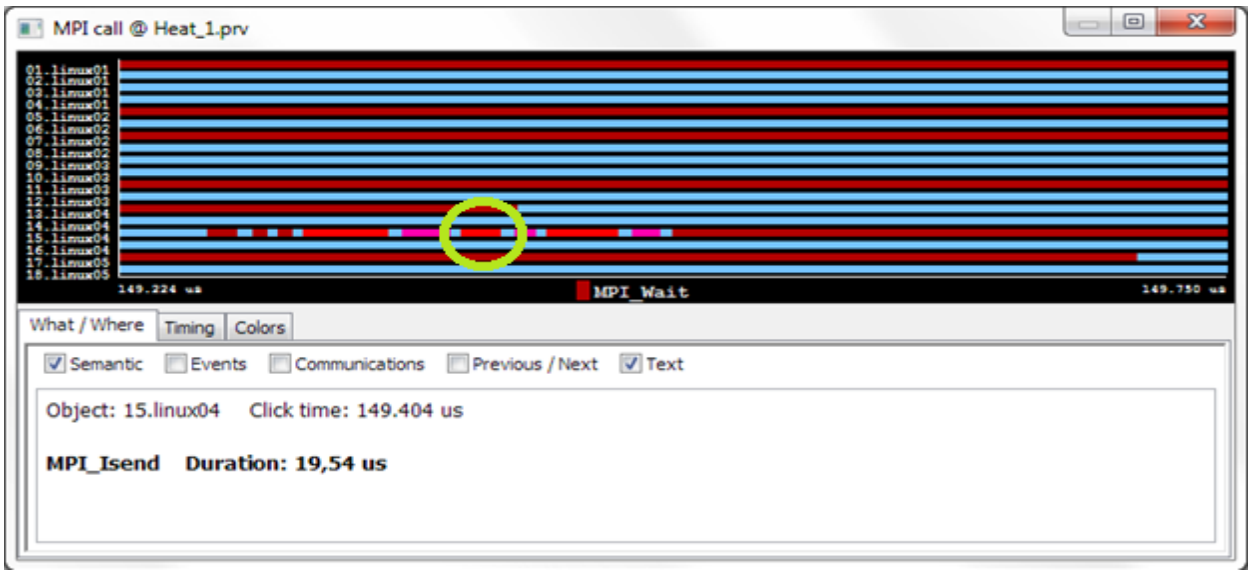
φαίνεται και στο παράδειγμα των *Εικόνων 21* και *22*, όπου ο χρόνος εκτέλεσης είναι ο ίδιος (Click time) και ταυτοποιείται η εντολή MPI_Isend.



Εικόνα 20: node_bandwidth.cfg - όλες οι υλοποιήσεις



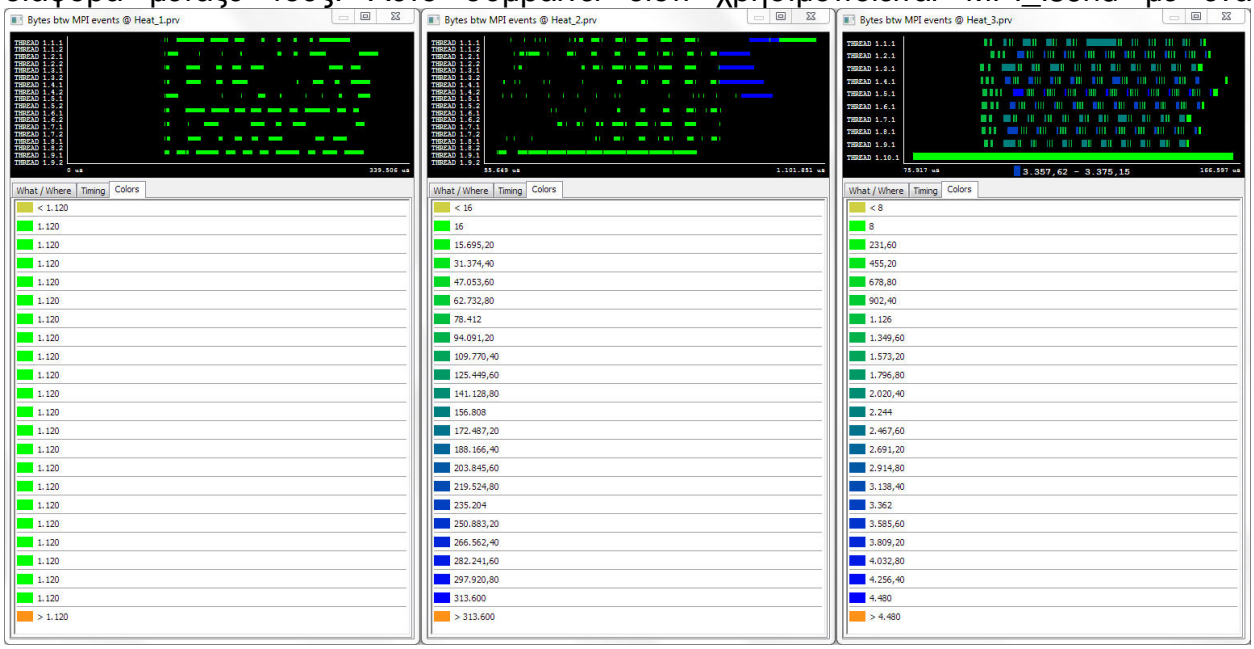
Εικόνα 21: node_bandwidth.cfg - υλοποίηση 1η



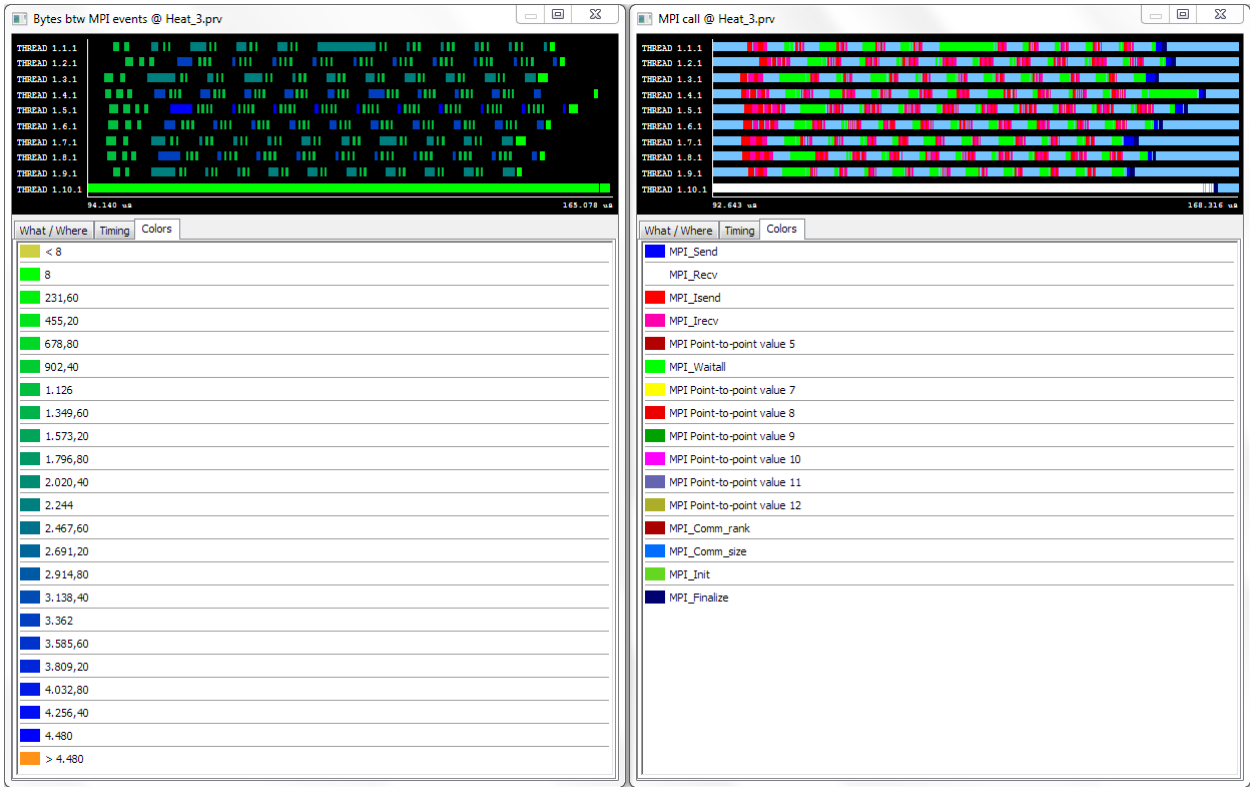
Εικόνα 22: MPI_call.cfg - υλοποίηση 1η

3.3.2.2 Φάκελος mpi/views/advanced

bytes_sr_within_call.cfg: Οι τρεις υλοποιήσεις παρουσιάζουν σημαντικές διαφορές (*Εικόνα 23*). Η πρώτη υλοποίηση έχει συγκεκριμένη τιμή μεγέθους μηνύματος και προκύπτει από το γεγονός ότι χρησιμοποιείται πάντα το ίδιο μέγεθος πίνακα δεδομένων στις εντολές MPI_Isend. Η επανάληψη της μοναδικής τιμής στο info panel συμβαίνει διότι το Paraver χρησιμοποιεί 23 γραμμές και αποχρώσεις για να αποδώσει το συνολικό εύρος τιμών, το οποίο χωρίζει σε κάθε περίπτωση σε ίσα διαστήματα, χρησιμοποιώντας τη μέση τιμή για να χαρακτηρίσει το κάθε διάστημα. Η δεύτερη υλοποίηση παρουσιάζει δύο αποχρώσεις, άρα και δύο τιμές μεγέθους δεδομένων οι οποίες έχουν μεγάλη διαφορά μεταξύ τους. Αυτό συμβαίνει διότι χρησιμοποιείται MPI_Isend με ένα



Εικόνα 23: bytes_sr_within_call.cfg - όλες οι υλοποιήσεις

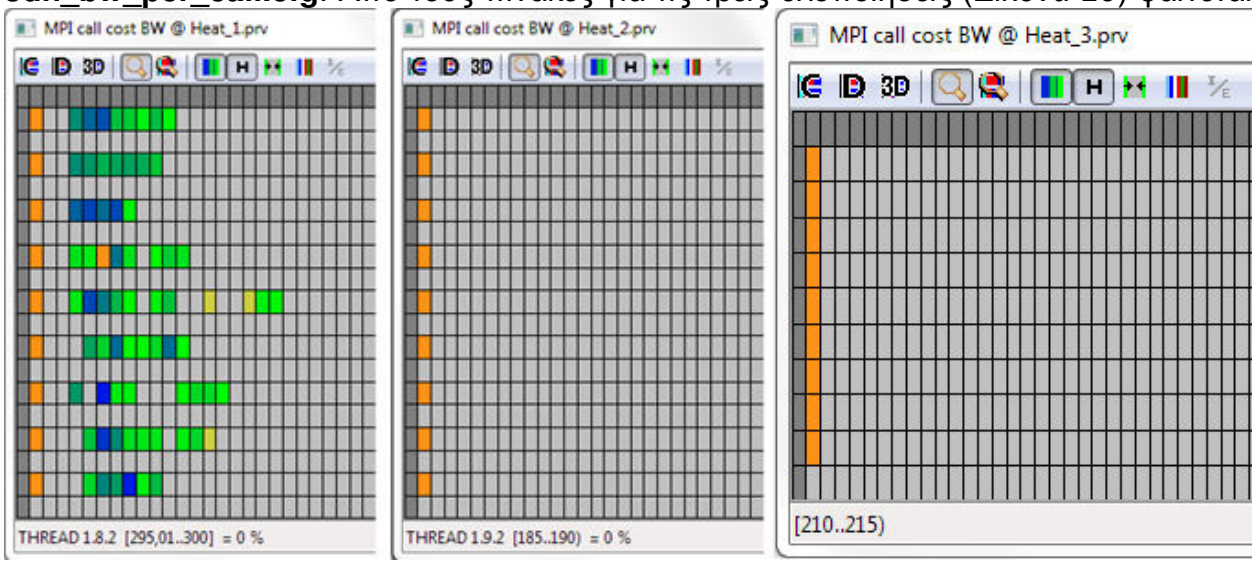


Εικόνα 24: bytes_sr_within_call.cfg και MPI_call.cfg - υλοποίηση 3η

συγκεκριμένο μέγεθος πίνακα δεδομένων για την επικοινωνία των διεργασιών, αλλά και MPI_Send στο τέλος του προγράμματος για να αποστείλει, κάθε διεργασία τον πίνακα δεδομένων της στην κύρια διεργασία, που είναι η 1. Ο πίνακας αυτός είναι πολλές φορές μεγαλύτερος από το πίνακα που ανταλλάζουν οι διεργασίες. Παρατηρείται ακόμα ότι η ανταλλαγή του πίνακα αυτού δεν καταγράφεται για όλες τις διεργασίες. Από τον έλεγχο του κώδικα προκύπτει ότι ό πίνακας που κάνει χρήση το MPI_Send είναι παραμετροποιημένος με μεταβλητές που έχουν σχέση με τη διεργασία, ενώ δε θα έπρεπε, με αποτέλεσμα να μην γίνεται αποστολή δεδομένων για όλες τις διεργασίες. Τέλος, η τρίτη υλοποίηση παρουσιάζει 4 αποχρώσεις, άρα και τέσσερις τιμές μεγέθους μηνύματος. Κάνοντας χρήση και του MPI_call.cfg συμπεραίνεται από τη συσχέτιση των χρόνων εκτέλεσης των δύο γραφημάτων, ότι η μικρότερη τιμή αντιστοιχεί στις εντολές MPI_Isend (*Εικόνα 24*). Οι υπόλοιπες αντιστοιχούν σε εντολές MPI_Waitall, οι οποίες ανάλογα τη διεργασία σχετίζονται με δύο, τρείς ή τέσσερις MPI_Isend εντολές. Η τιμή του μεγέθους μηνύματος για κάθε MPI_Waitall είναι το γινόμενο του μεγέθους μηνύματος της MPI_Isend εντολής επί τον αριθμό των εντολών που της αντιστοιχούν

3.3.2.3 Φάκελος *mpi/analysis/other*

3dh_bw_per_call.cfg: Από τους πίνακες για τις τρεις υλοποιήσεις (*Εικόνα 25*) φαίνεται

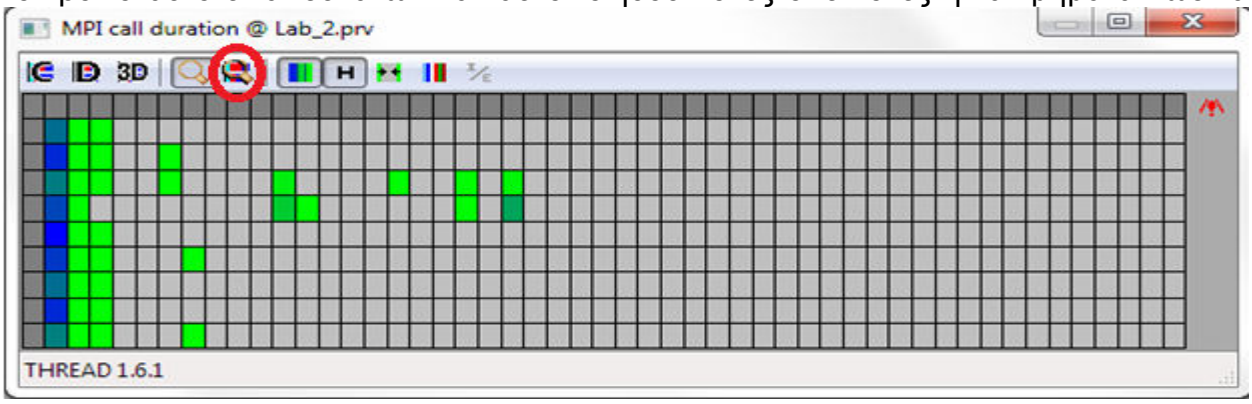


Εικόνα 25: 3dh_bw_per_call.cfg - όλες οι υλοποιήσεις

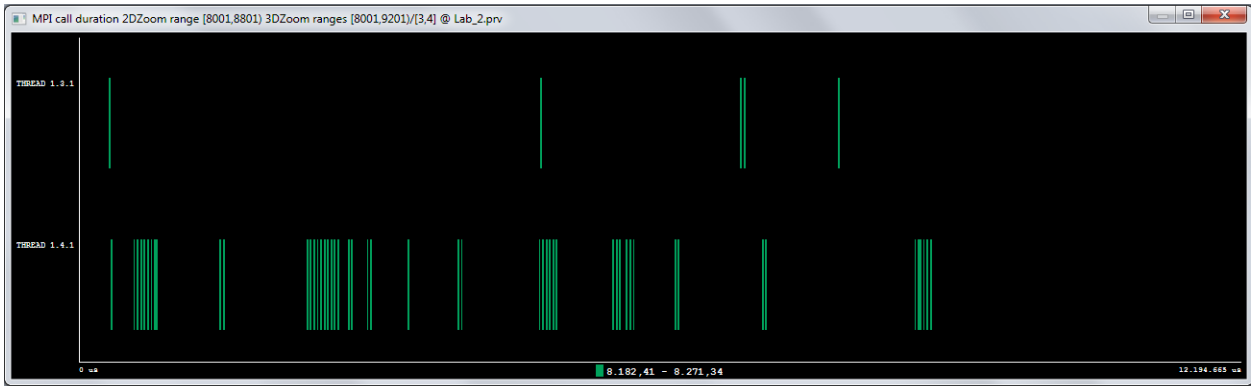
καθαρά η διαφορά στη χρήση του διαθέσιμου εύρους ζώνης. Η πρώτη υλοποίηση παρουσιάζει διακυμάνσεις στο ενεργό εύρος ζώνης, οι οποίες δε παρατηρούνται στις άλλες δύο υλοποιήσεις και πιθανόν οφείλονται στη χρήση των εντολών με πρόθεμα MPI_Cart_. Οι εντολές αυτές διευκρινίζουν την τοπολογία των επεξεργαστών, ώστε να είναι πιο αποδοτική η επικοινωνία μεταξύ των διεργασιών. Για τη συγκεκριμένη υλοποίηση έχει διευκρινιστεί ένα τετραγωνικό πλέγμα, το οποίο όμως δεν αντιστοιχεί στη πραγματική τοπολογία των επεξεργαστών και των πυρήνων τους, στους οποίους εκτελείται το πρόγραμμα. Κάτι τέτοιο όμως δεν προσφέρει στη ταχύτητα του προγράμματος, αφού πιθανόν να αναγκάζει διεργασίες που ανήκουν σε απομακρυσμένους επεξεργαστές να επικοινωνήσουν. Οι κενές σειρές στις δύο πρώτες υλοποιήσεις αντιστοιχούν στα νήματα που έχουν δημιουργηθεί από το OpenMP, που δεν ανταλλάσουν μηνύματα, οπότε και δεν υπάρχουν μετρήσεις για ενεργό εύρος ζώνης.

3dh_duration_per_call.cfg: Χρησιμοποιώντας τη δεύτερη υλοποίηση αλλά για 1000 βήματα, αντί 10, προκύπτει ο πίνακας της *Εικόνα 26*. Πατώντας το κουμπί για τη

επιλογή στηλών και την εμφάνιση του αντίστοιχου διαγράμματος, που είναι σημειωμένο με το κόκκινο κύκλο, και επιλέγοντας τη τελευταία στήλη με τιμές, δημιουργείται το γράφημα της *Εικόνας 27*. Κάνοντας zoom στη πρώτη γραμμή για την διεργασία 3 και στη συνέχεια διπλό κλικ εμφανίζεται ο χρόνος στο παράθυρο πληροφοριών (*Εικόνα 28*). Ο χρόνος αυτός μπορεί να βρεθεί στο γράφημα που παρουσιάζει τις εντολές MPI, το οποίο ανοίγει πατώντας το κουμπί 3D. Κάνοντας το ανάλογο zoom ταυτοποιείται η εντολή MPI_Isend ως η εντολή με την αντίστοιχη διάρκεια (*Εικόνα 29*, η εντολή χαρακτηρισμένη με κόκκινο χρώμα και τη μεγαλύτερη διάρκεια για τη διεργασία 3). Με το τρόπο αυτό είναι δυνατών να ταυτοποιηθούν όλες οι εντολές ή τα τμήματα κώδικα



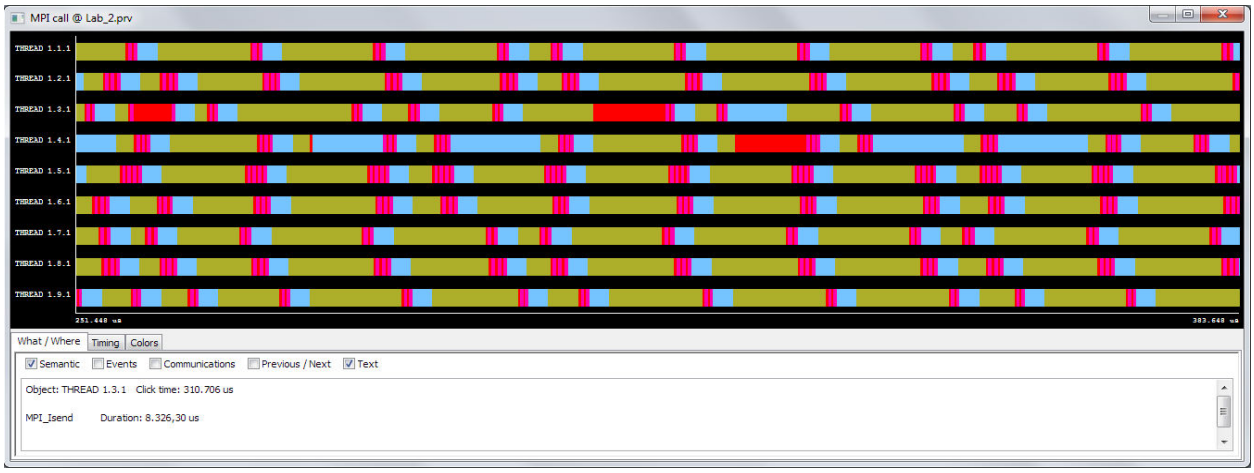
Εικόνα 26: 3dh_duration_per_call.cfg - υλοποίηση 2η



Εικόνα 27: Γράφημα από επιλεγμένες τιμές

Εικόνα 28: Zoom από το γράφημα τις εικόνας 27

που δημιουργούν καθυστερήσεις και να τροποποιηθούν για να βελτιωθούν οι επιδόσεις του προγράμματος. Στο συγκεκριμένο παράδειγμα έγινε χρήση 1000 βημάτων στο πρόγραμμα, ώστε να γίνει εμφανές ότι η διαδικασία αυτή δεν επηρεάζεται από τον αριθμό των εντολών και είναι δυνατόν να εφαρμοστεί και για μεγάλα προγράμματα.



Εικόνα 29: MPI_call.cfg

3.3.3 Πρόγραμμα υπολογισμού εμβαδού με τη μέθοδο των τραπεζίων

Το πρόγραμμα αυτό υπολογίζει το εμβαδό της συνάρτησης $f(x) = x^2$ μεταξύ δύο τιμών του x. Το πρόγραμμα έχει υλοποιηθεί με πέντε διαφορετικούς τρόπους. Καταρχήν σειριακά και με τέσσερις διαφορετικούς τρόπους χρήσης παραλληλίας με MPI. Οι διαφορετικοί τρόποι αυτοί είναι:

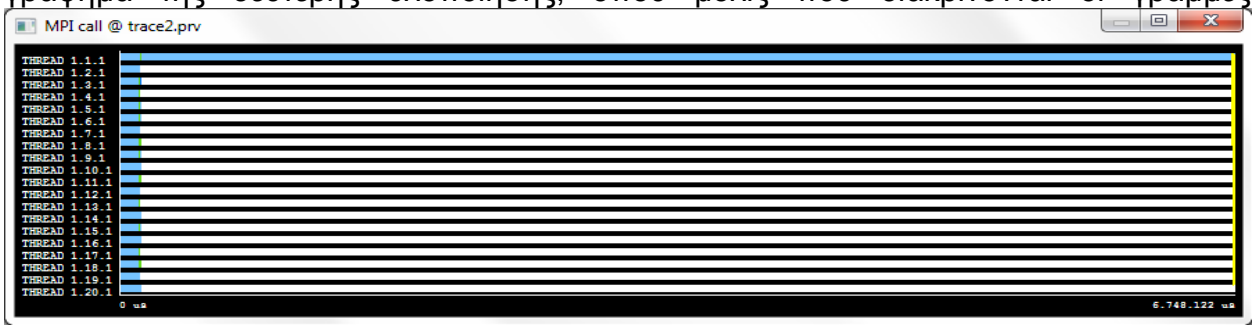
1: Η διεργασία 0 θεωρείται κύρια. Οι μεταβλητές του προβλήματος δεν μπορούν να καταχωρηθούν από το χρήστη κατά τη λειτουργία του προγράμματος και για να αλλάξουν πρέπει να γίνει αλλαγή στο κώδικα. Κάθε διεργασία υπολογίζει ένα κομμάτι του προβλήματος και στέλνει το αποτέλεσμα στην κύρια διεργασία με χρήση MPI_Send και MPI_Recv. Η κύρια διεργασία υπολογίζει το τελικό αποτέλεσμα και το τυπώνει.

2: Η διεργασία 0 θεωρείται κύρια. Οι μεταβλητές του προβλήματος καταχωρούνται από το χρήστη κατά τη λειτουργία του προγράμματος, στην κύρια διεργασία, η οποία τις στέλνει στις υπόλοιπες με χρήση MPI_Send και MPI_Recv. Κάθε διεργασία υπολογίζει ένα κομμάτι του προβλήματος και στέλνει το αποτέλεσμα στην κύρια διεργασία με χρήση MPI_Send και MPI_Recv. Η κύρια διεργασία υπολογίζει το τελικό αποτέλεσμα και το τυπώνει.

3: Η διεργασία 0 θεωρείται κύρια. Οι μεταβλητές του προβλήματος καταχωρούνται από το χρήστη κατά τη λειτουργία του προγράμματος, στην κύρια διεργασία, η οποία τις στέλνει στις υπόλοιπες με χρήση MPI_Bcast για κάθε μεταβλητή ξεχωριστά. Κάθε διεργασία υπολογίζει ένα κομμάτι του προβλήματος. Γίνεται χρήση MPI_Reduce για να συγκεντρωθούν όλα τα δεδομένα στην κύρια διεργασία. Η κύρια διεργασία υπολογίζει το τελικό αποτέλεσμα και το τυπώνει.

4: Η διεργασία 0 θεωρείται κύρια. Οι μεταβλητές του προβλήματος καταχωρούνται από το χρήστη κατά τη λειτουργία του προγράμματος, στην κύρια διεργασία, η οποία τις στέλνει στις υπόλοιπες με χρήση MPI_Bcast και χρήση MPI_Datatype, ώστε να αποσταλούν όλες οι μεταβλητές μαζί. Κάθε διεργασία υπολογίζει ένα κομμάτι του προβλήματος. Γίνεται χρήση MPI_Reduce για να συγκεντρωθούν όλα τα δεδομένα στην κύρια διεργασία. Η κύρια διεργασία υπολογίζει το τελικό αποτέλεσμα και το τυπώνει.

Ο χρόνος εκτέλεσης των παραπάνω υλοποιήσεων είναι της τάξης των δεκάτων του δευτερολέπτου. Στις περιπτώσεις που απαιτείται καταχώρηση δεδομένων από το χρήστη, η χρονική διάρκεια του προγράμματος φτάνει σε μερικά δευτερόλεπτα. Ο επιπλέον αυτός χρόνος αποτυπώνεται στα γραφήματα αλλοιώνοντας τους χρόνους εκτέλεσης και δυσκολεύοντας την ανάλυση και σύγκριση. Στην *Εικόνα 30* φαίνεται το γράφημα της δεύτερης υλοποίησης, όπου μόλις που διακρίνονται οι γραμμές



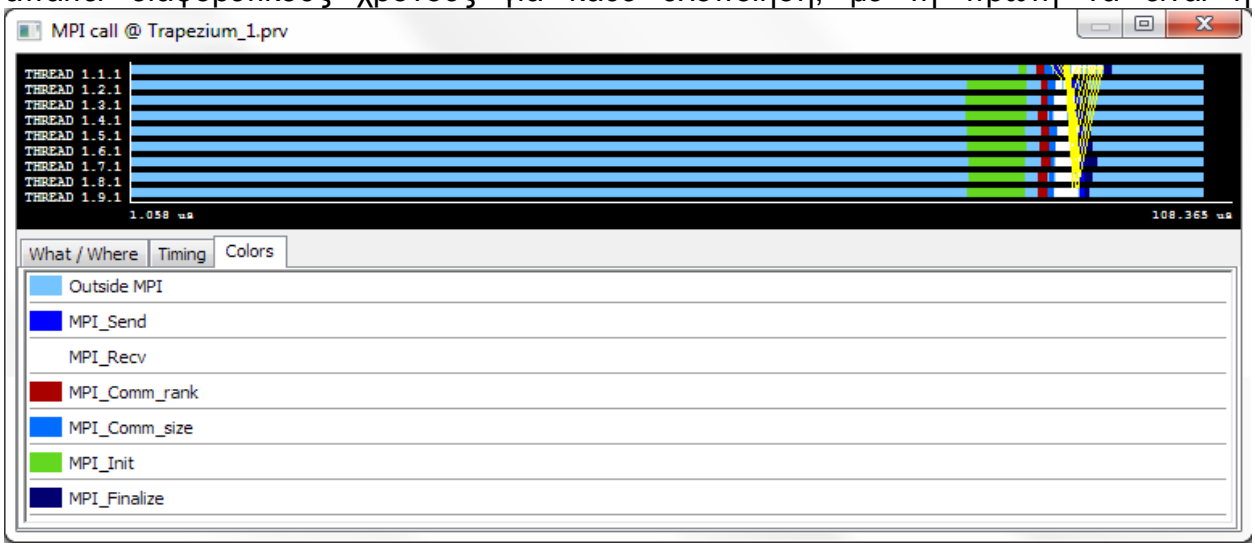
Εικόνα 30: MPI_call.cfg - Με το χρόνο καταχώρησης δεδομένων

επικοινωνίας μεταξύ των διεργασιών στο τέλος του γραφήματος. Για καλύτερη αποτύπωση της λειτουργίας των υλοποιήσεων έχει τροποποιηθεί ο αρχικός κώδικας ώστε να μην χρειάζεται καταχώρηση δεδομένων από το χρήστη. Έτσι, όμως δεν υπάρχει ανάγκη σύγκρισης της πρώτης υλοποίησης με τη δεύτερη, καθώς διαφέρουν μόνο στο τρόπο καταχώρησης των δεδομένων. Κατά συνέπεια η σύγκριση της πρώτης υλοποίησης με τη τρίτη και τέταρτη θα καλυφθεί με τη σύγκριση της δεύτερης με αυτές. Σε κάθε περίπτωση έχει χρησιμοποιηθεί ως αρχική τιμή του άξονα x το 0, ως τελική το 100 και ο υπολογισμός γίνεται με χρήση 1024 τραπεζίων. Χρησιμοποιήθηκαν 9 διεργασίες με 2 διεργασίες σε κάθε επεξεργαστή.

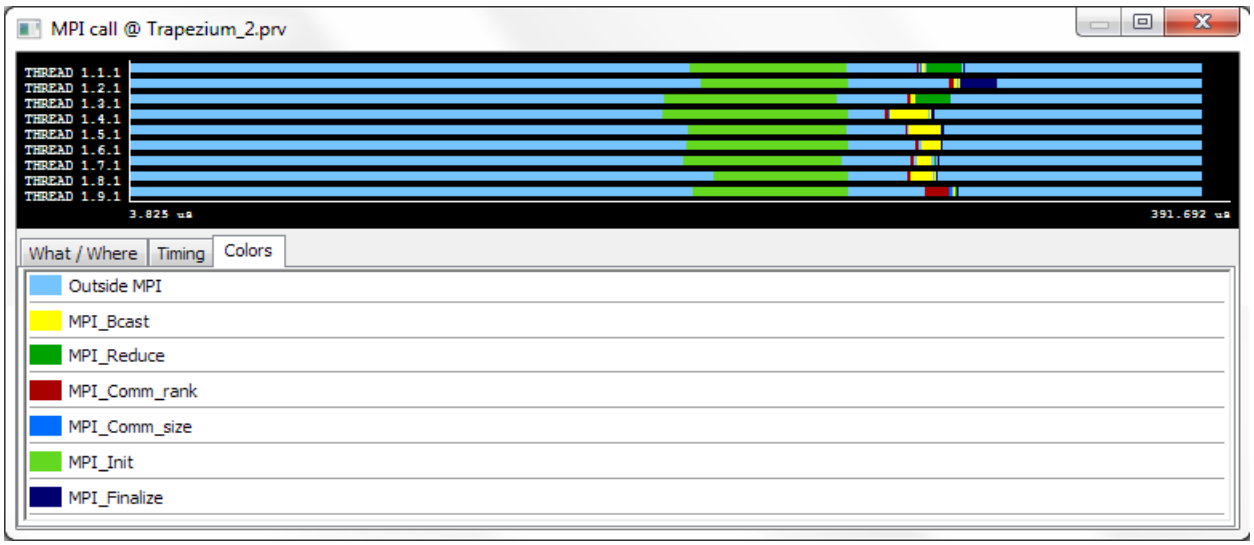
3.3.4 Σύγκριση υλοποιήσεων προγράμματος υπολογισμού εμβαδού με τη μέθοδο των τραπεζίων

3.3.4.1 Φάκελος *mpi/views*

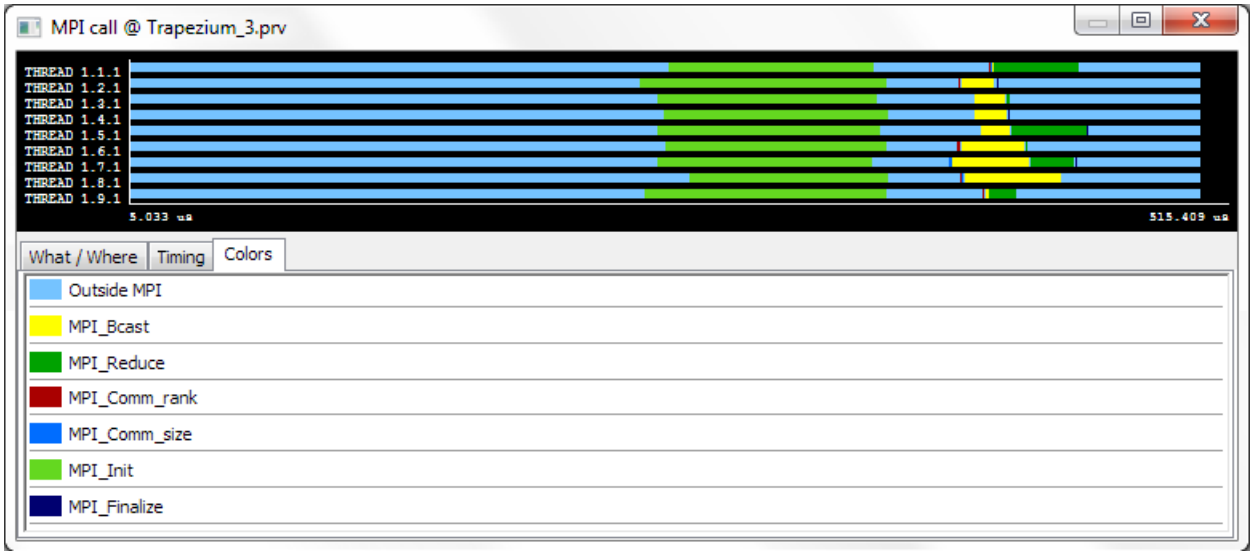
MPI_call.cfg: Τα γραφήματα για τις τρεις υλοποιήσεις φαίνονται αντίστοιχα στις *Εικόνες 31*, *32*, και *33*. Από τη κλίμακα χρόνου, παρατηρείται ότι η εκτέλεση του προγράμματος απαιτεί διαφορετικούς χρόνους για κάθε υλοποίηση, με τη πρώτη να είναι η



Εικόνα 31: MPI_call.cfg - υλοποίηση 1η



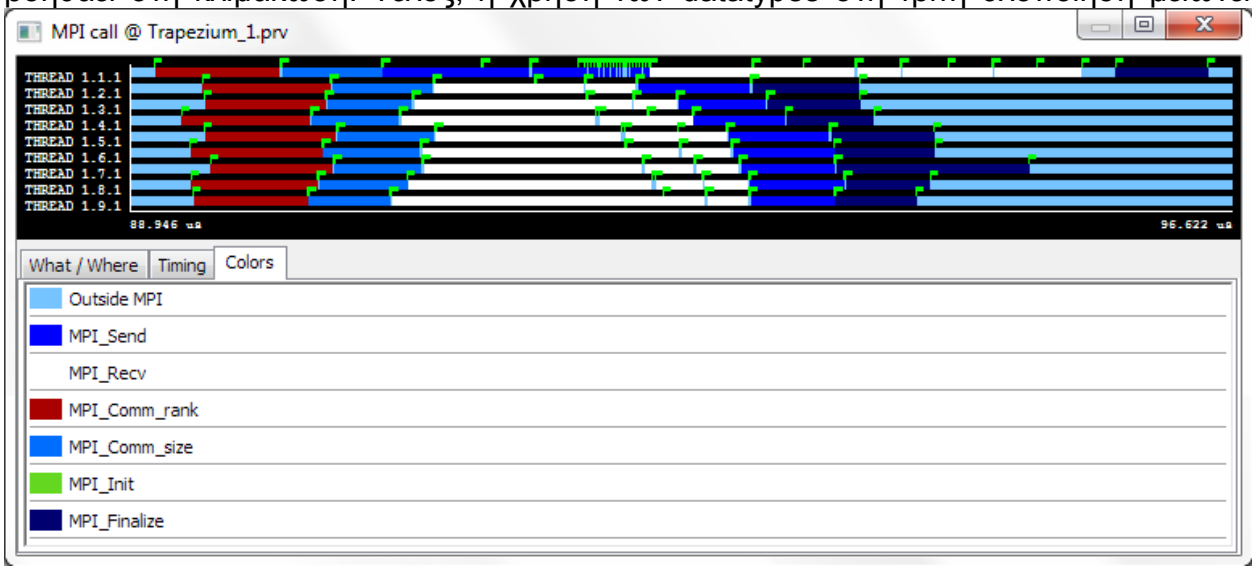
Εικόνα 32: MPI_call.cfg - υλοποίηση 2η



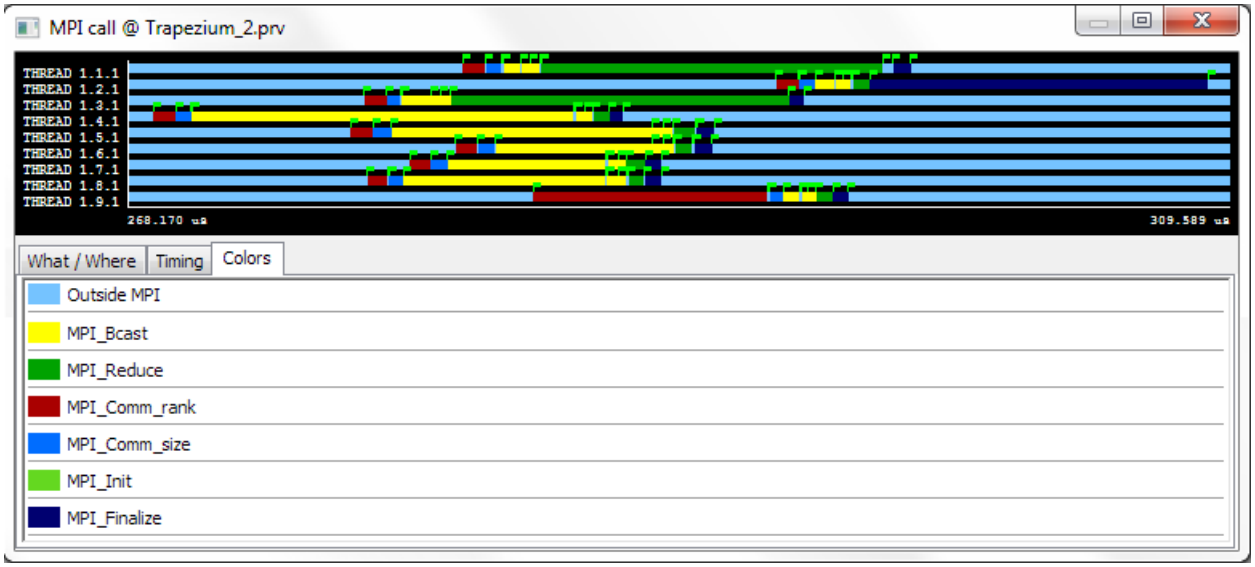
Εικόνα 33: MPI_call.cfg - υλοποίηση 3η

γρηγορότερη και τη τρίτη τη πιο αργή. Παρατηρείται, ακόμα ότι γραμμές επικοινωνίας μεταξύ των διεργασιών υπάρχουν μόνο για τη πρώτη υλοποίηση, διότι κάνει χρήση των εντολών MPI_Send και MPI_Recv που είναι point to point εντολές, ενώ οι υπόλοιπες δύο κάνουν χρήση των MPI_Bcast και MPI_Reduce που είναι collective εντολές. Κάνοντας zoom στις περιοχές των γραφημάτων μετά τα MPI_Init, απενεργοποιώντας τις γραμμές επικοινωνίας και ενεργοποιώντας τους σηματοδότες των εντολών παρατηρούνται καλύτερα οι λεπτομέρειες εκτέλεσης (*Εικόνες 34, 35* και *36*). Για τη πρώτη υλοποίηση παρατηρείται ότι η πρώτη διεργασία είναι επιβαρυμένη με την αποστολή δεδομένων στις υπόλοιπες διεργασίες και την παραλαβή των νέων δεδομένων μετά τους υπολογισμούς κάθε διεργασίας. Στο συγκεκριμένο πρόβλημα το μέγεθος των δεδομένων και ο αριθμός των επεξεργαστών δεν δημιουργούν καθυστερήσεις, έτσι ο χρόνος της υλοποίησης αυτής είναι ο καλύτερος. Όμως, σε μεγάλης κλίμακας προβλήματα όπου τα δεδομένα μπορεί να είναι της τάξης των megabytes και ο αριθμός των επεξεργαστών της τάξης των χιλίων ή και παραπάνω, οπότε περιπλέκεται και η τοπολογία, η χρήση point to point εντολών θα δημιουργούσε μεγάλες καθυστερήσεις και το πρόγραμμα θα ήταν μη αποδοτικό. Συγκρίνοντας με τη δεύτερη υλοποίηση είναι εμφανές ότι ο αριθμός των εντολών προς εκτέλεση με τη χρήση collective εντολών μειώνεται δραματικά, γεγονός που όπως προαναφέρθηκε,

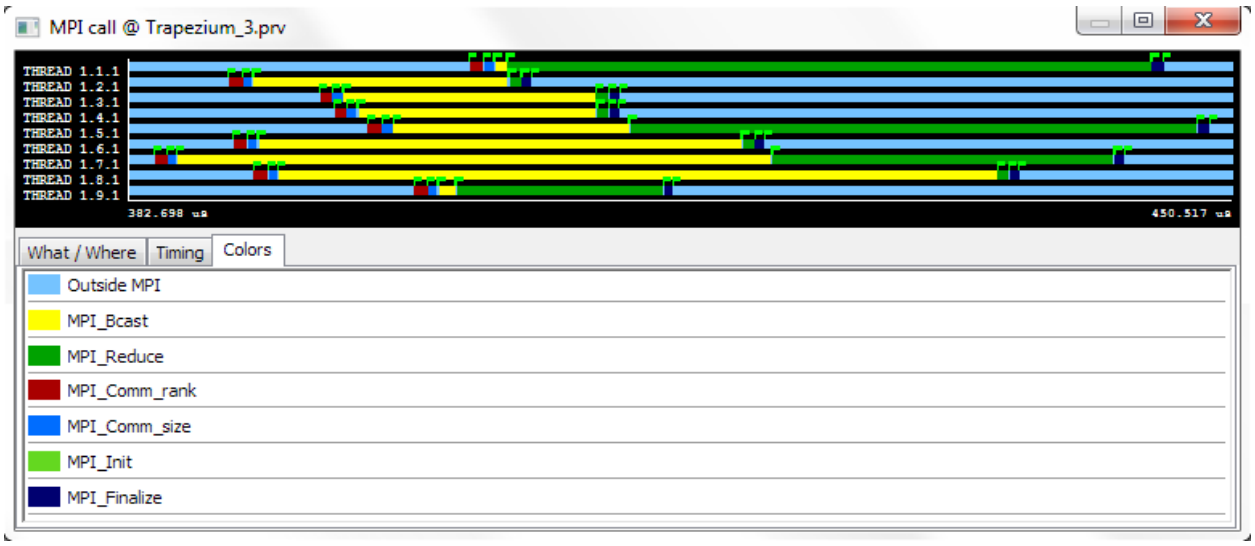
βοηθάει στη κλιμάκωση. Τέλος, η χρήση των datatypes στη τρίτη υλοποίηση μειώνει



Εικόνα 34: MPI_call.cfg - υλοποίηση 1η, zoom και σηματοδότες εντολών



Εικόνα 35: MPI_call.cfg - υλοποίηση 2η, zoom και σηματοδότες εντολών



Εικόνα 36: MPI_call.cfg - υλοποίηση 3η, zoom και σηματοδότες εντολών

ακόμα περισσότερο τον αριθμό των εντολών που απαιτούνται για την επικοινωνία των διεργασιών. Παρόλο που στο συγκεκριμένο παράδειγμα η πρώτη υλοποίηση εκτελείται γρηγορότερα από τις υπόλοιπες, από την ανάλυση προκύπτει ότι δεν είναι η αποδοτικότερη γενικά και δεν πρέπει να προτιμάται.

4. ΜΕΛΕΤΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

4.1 Πρόγραμμα προς μελέτη

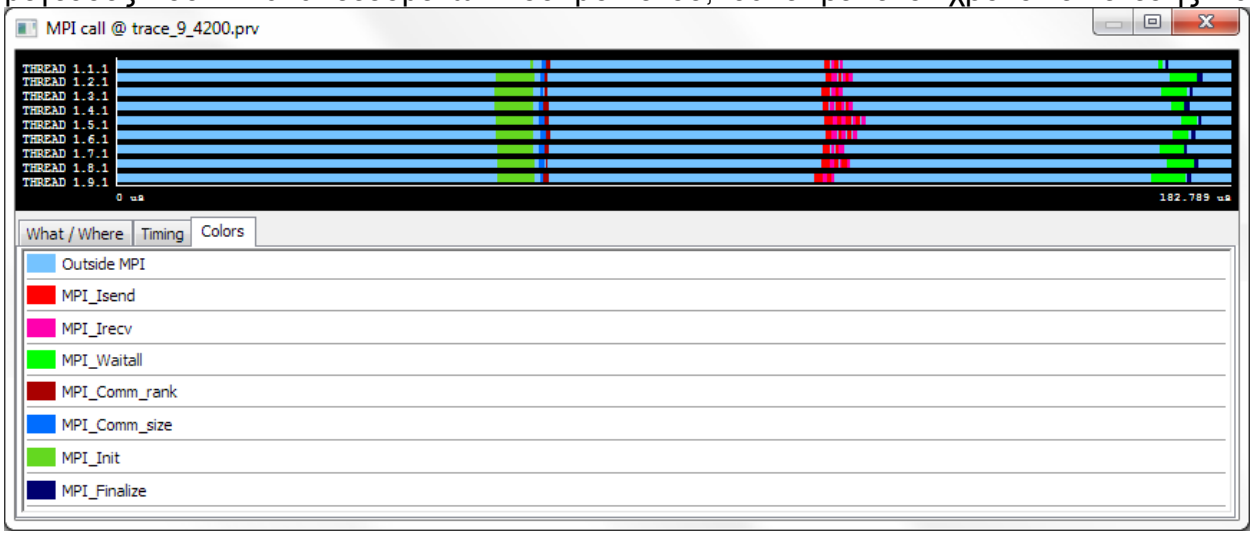
Για τη μελέτη κλιμάκωσης θα χρησιμοποιηθεί μια απλοποιημένη μορφή της τρίτης υλοποίησης του προγράμματος προσομοίωσης μετάδοσης θερμότητας. Συγκεκριμένα, τα δεδομένα του μοντέλου προσομοίωσης βρίσκονται σε ένα κεντρικό τετραγωνικό πλέγμα (grid), που υλοποιείται με τη χρήση ενός δισδιάστατου πίνακα, τον οποίο μοιράζονται εξίσου οι διεργασίες. Οι πυρήνες των επεξεργαστών, σε καθένα από τους οποίους εκτελείται και από μία διεργασία, θεωρείται ότι είναι τοποθετημένοι επίσης σε ένα τετραγωνικό πλέγμα, όπου κάθε ένας επικοινωνεί με τέσσερις γειτονικούς, εκτός από τους πυρήνες που βρίσκονται στις ακμές του πλέγματος, οι οποίοι επικοινωνούν με λιγότερους. Για κάθε σημείο του πλέγματος δεδομένων υπολογίζεται μία νέα τιμή με τη χρήση stencil πέντε σημείων. Αφού το κεντρικό πλέγμα διαμοιράζεται στις διεργασίες, υπάρχουν σημεία για τα οποία απαιτείται επικοινωνία μεταξύ των διεργασιών και ανταλλαγή δεδομένων ώστε να είναι δυνατός ο υπολογισμός του stencil Τα σημεία αυτά ανταλλάσσονται μεταξύ των διεργασιών μέσω επικοινωνίας με MPI non-blocking εντολές. Σε μία περίπτωση προστίθεται και χρονική εξέλιξη του μοντέλου που υλοποιείται με επανάληψη της παραπάνω διαδικασίας.

4.2 Περιπτώσεις και παράμετροι εκτέλεσης

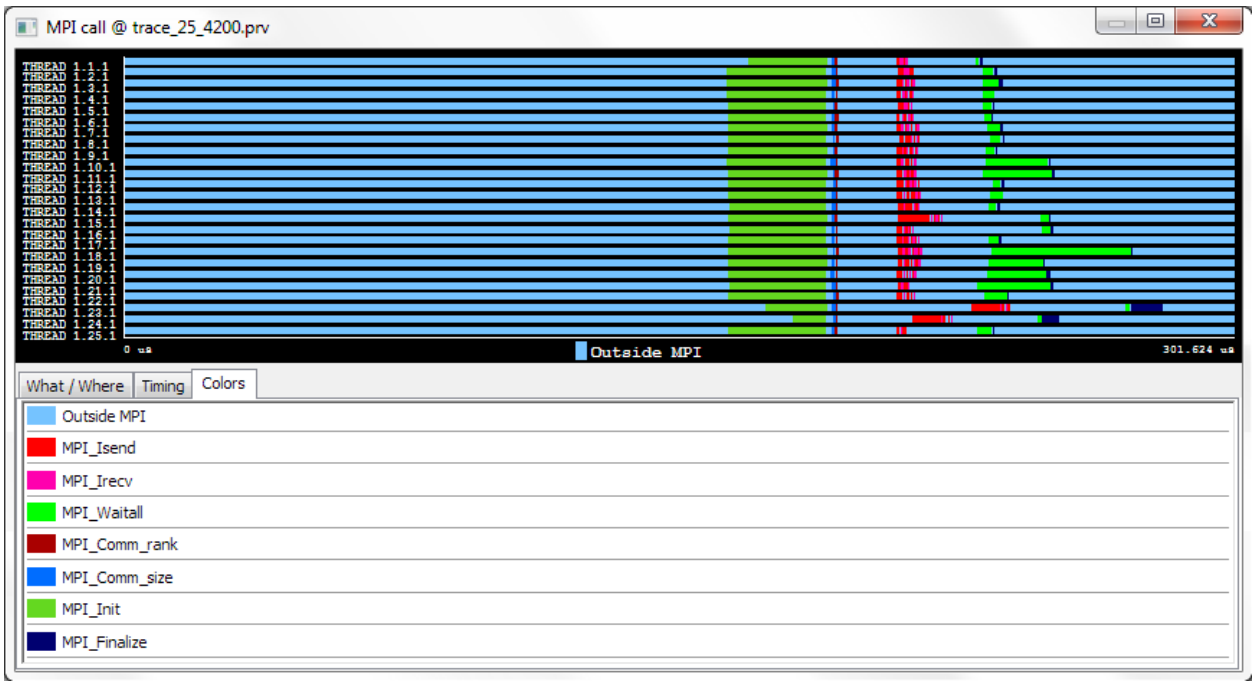
4.2.1 Σταθερός αριθμός υπολογισμών - Μεταβαλλόμενος αριθμός διεργασιών

Αρχικά, εκτελείται το πρόγραμμα τρεις φορές, όπου διατηρείται σταθερό το μέγεθος του κεντρικού grid, του οποίου η πλευρά είναι 4200, και μεταβάλλεται ο αριθμός των διεργασιών. Το πρόγραμμα εκτελείται για 9, 25 και 49 διεργασίες, οπότε το μέγεθος της πλευράς του grid κάθε διεργασίας είναι 1400, 840 και 600 αντίστοιχα.

Όπως φαίνεται και από τα γραφήματα του MPI_call.cfg για τις τρεις εκτελέσεις, *Εικόνες 37*, *38* και *39* αντίστοιχα, ο συνολικός χρόνος εκτέλεσης του προγράμματος αυξάνεται με την αύξηση των διεργασιών, με πολύ χρόνο να αναλώνεται στην εκκίνηση του MPI. Συγκρίνοντας τους χρόνους εκτέλεσης των προγραμμάτων εξαιρώντας το χρόνο πριν και κατά τη διάρκεια του MPI_Init, παρατηρείται ότι είναι περίπου ίδιοι και στις τρεις περιπτώσεις, με μικρή βελτίωση από τις 9 διεργασίες στις 25, αλλά χωρίς βελτίωση από τις 25 διεργασίες στις 49. Με την αύξηση των διεργασιών και τη διατήρηση του μεγέθους του πίνακα δεδομένων του μοντέλου, θα έπρεπε οι χρόνοι εκτέλεσης να

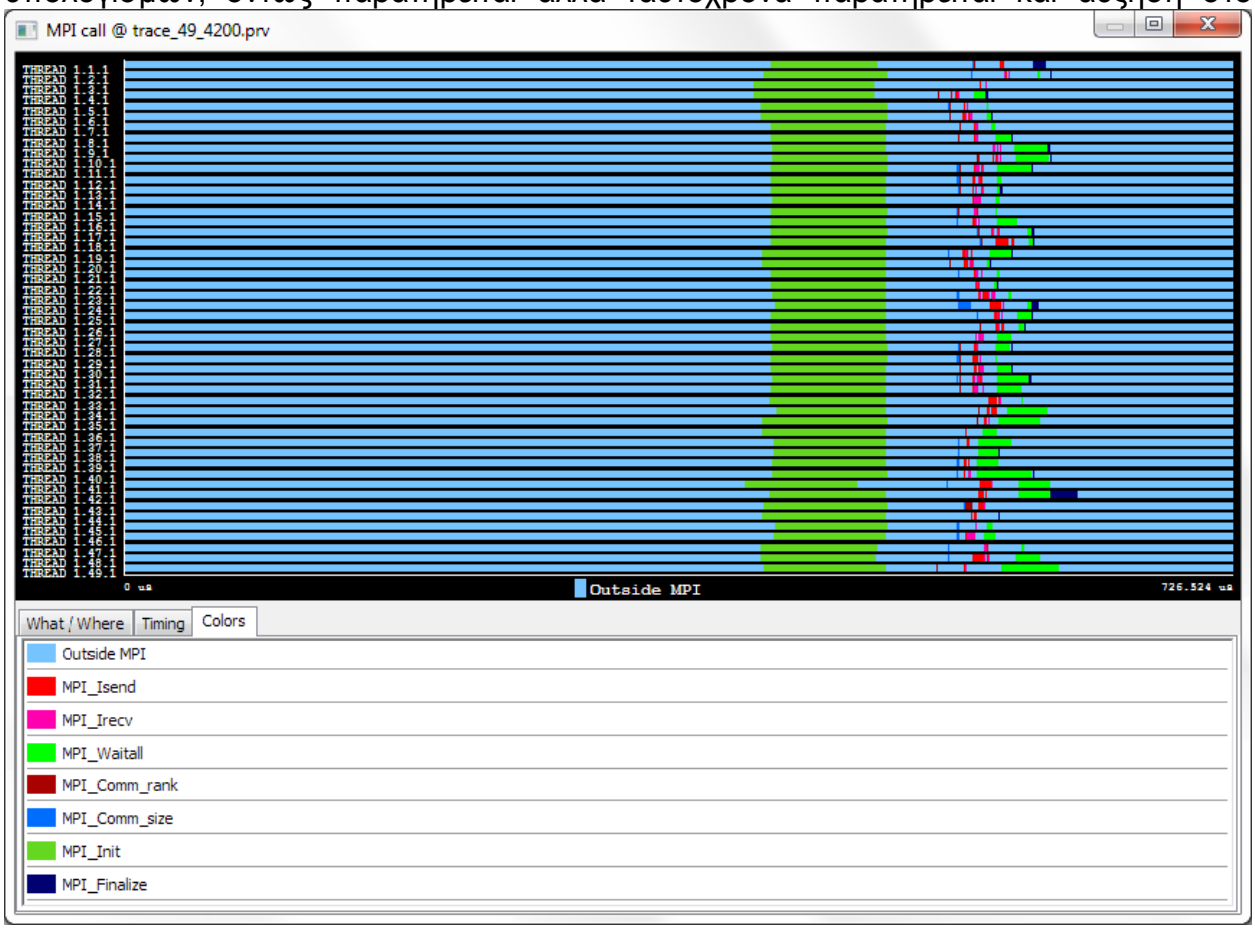


Εικόνα 37: MPI_call.cfg - 9 διεργασίες, 4200 grid



Εικόνα 38: MPI_call.cfg - 25 διεργασίες, 4200 grid

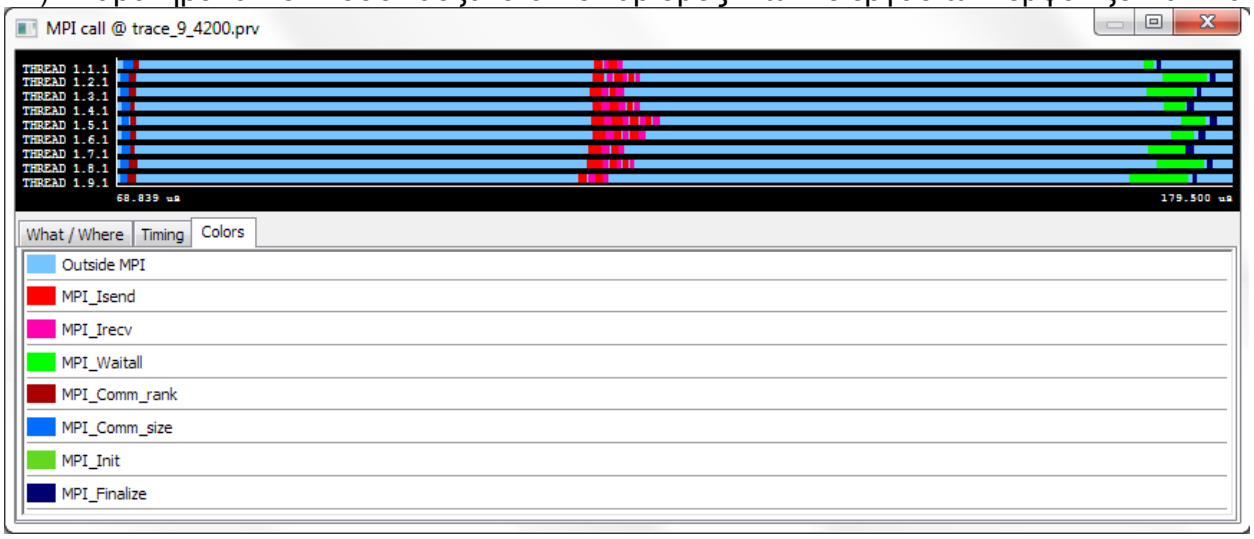
βελτιώνονται, αφού σε κάθε διεργασία ανατίθενται λιγότεροι υπολογισμοί όσο αυξάνεται ο αριθμός των διεργασιών. Μεταξύ MPI_Comm_rank και του πρώτου MPI_Isend γίνεται η αρχικοποίηση των πινάκων και μεταξύ MPI_Irecv και MPI_Waitall υπολογίζονται οι νέες τιμές του πίνακα μέσω του stencil των πέντε σημείων, όπου η μείωση του χρόνου υπολογισμών, όντως παρατηρείται αλλά ταυτόχρονα παρατηρείται και αύξηση στο



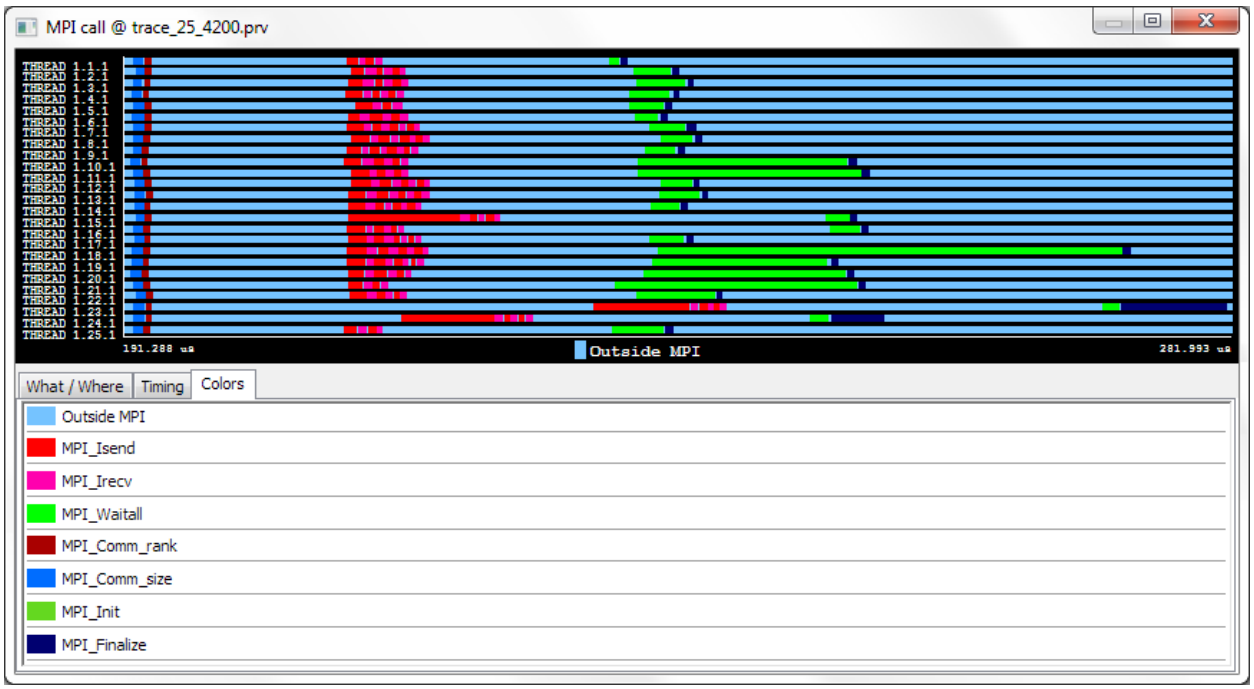
Εικόνα 39: MPI_call.cfg - 49 διεργασίες, 4200 grid

χρόνο που απαιτείται για την επικοινωνία των διεργασιών.

Κάνοντας zoom στις περιοχές από το MPI_Comm_rank και μετά, (*Εικόνες 40*, *41* και *42*) παρατηρείται ότι όσο αυξάνεται ο αριθμός των διεργασιών εμφανίζονται και



Εικόνα 40: MPI_call.cfg - 9 διεργασίες, 4200 grid, zoom



Εικόνα 41: MPI_call.cfg - 25 διεργασίες, 4200 grid, zoom

μεγαλύτερης διάρκειας MPI_Send, MPI_Recv και MPI_Waitall που καθυστερούν το πρόγραμμα. Με τη χρήση του Info Panel προκύπτουν οι τιμές των *Πινάκων 1, 2* και *3* για τους χρόνους (σε nanosecond) αρχικοποίησης των πινάκων, υπολογισμού των νέων τιμών και επικοινωνίας για κάθε διεργασία αντίστοιχα. Παρατηρείται ότι με την αύξηση των διεργασιών ελαττώνονται οι χρόνοι αρχικοποίησης και υπολογισμών, αλλά αυξάνονται οι χρόνοι επικοινωνίας. Από τις μέσες τιμές για κάθε εκτέλεση προκύπτει ότι από τις 9 στις 25 διεργασίες ο συνολικός χρόνος υπολογισμών και επικοινωνίας υποδιπλασιάζεται, αλλά από τις 25 διεργασίες στις 49 ο συνολικός χρόνος ελαττώνεται κατά το 1/3 μόνο. Το αυξανόμενο κόστος επικοινωνίας μεταξύ των διεργασιών, δεν επιτρέπει την επ' αόριστον βελτίωση όσο αυξάνεται ο αριθμός διεργασιών, αλλά υπάρχει άνω όριο, το οποίο εξαρτάται από την αναλογία υπολογισμών και διεργασιών και είναι διαφορετικό για κάθε πρόβλημα.

Πίνακας 1 | 9 διεργασίες

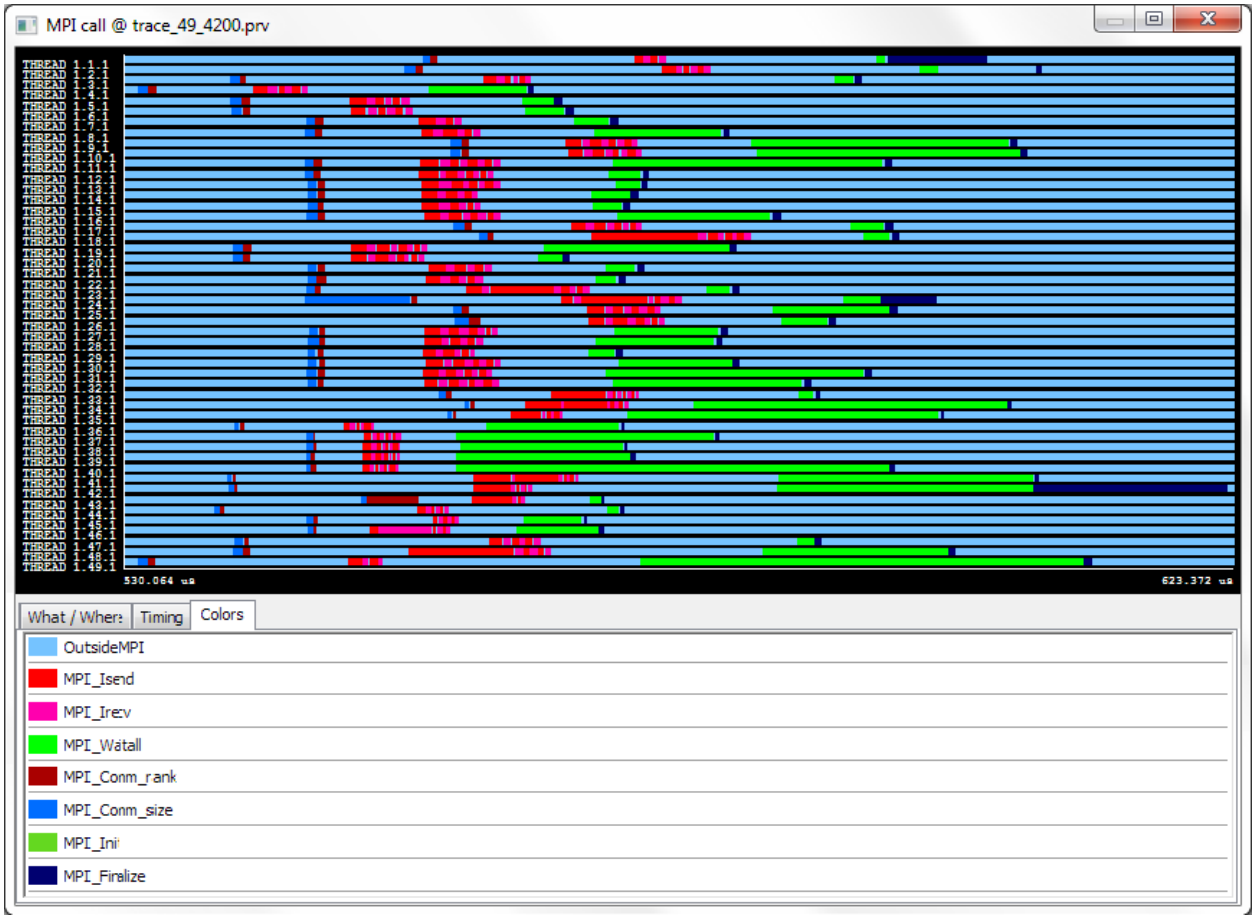
Διεργασία	Αρχικοποίηση	Υπολογισμοί	Επικοινωνία
1	45.094,07	51.697,25	857,37
2	45.287,59	51.787,79	4.455,70
3	44.968,24	51.721,86	4.678,63
4	45.206,83	52.057,96	1.940,98
5	44.853,55	51.802,64	2.462,71
6	45.090,27	51.949,74	2.412,23
7	44.826,74	51.845,78	3.738,16
8	44.712,16	51.719,93	4.640,77
9	43.812,98	51.813,32	5.841,57
Μέση τιμή	44.872,49	51.821,81	3.447,57
Άθροισμα	100.141,87		

Πίνακας 2 | 25 διεργασίες

Διεργασία	Αρχικοποίηση	Υπολογισμοί	Επικοινωνία
1	16.092,16	18.622,22	840,49
2	16.168,67	18.610,80	2.966,00
3	16.076,75	18.614,76	4.111,80
4	16.049,41	18.608,20	3.299,24
5	16.493,28	18.705,51	2.727,90
6	16.043,61	18.642,04	1.903,50
7	16.082,97	18.638,23	3.046,34
8	16.328,97	18.762,63	2.767,97
9	16.082,20	18.601,58	2.457,45
10	15.999,76	18.721,92	17.187,29
11	16.257,06	18.659,48	18.291,89
12	16.152,80	18.895,44	2.421,35
13	16.152,80	18.747,63	3.323,90
14	16.147,20	18.627,09	2.375,54
15	16.134,98	26.632,10	1.948,73
16	16.063,55	34.691,00	2.525,61
17	16.157,95	18.663,37	2.749,86
18	16.134,24	18.777,98	37.858,11
19	16.122,85	18.719,65	14.409,27
20	16.201,82	18.899,38	16.568,89
21	16.023,94	18.594,94	19.926,71
22	16.065,41	18.714,10	14.409,27
23	36.089,85	30.630,28	1.555,31
24	20.243,95	22.714,12	1.555,94
25	15.859,32	18.789,76	4.275,46
Μέση τιμή	17.089,02	20.291,37	7.420,15
Άθροισμα	44.800,54		

Πίνακας 3 | 49 διεργασίες

Διεργασία	Αρχικοποίηση	Υπολογισμοί	Επικοινωνία
1	16.475	17.737,75	745,43
2	20.162,83	17.498,71	1.620,18
3	19.987,02	25.479,96	1.609,10
4	8.228,72	10.061,21	8.254,12
5	8.284,30	9.503,93	2.605,07
6	8.420,08	9.552,50	3.144,50
7	8.141,80	9.505,91	3.001,85
8	8.300,21	9.645,84	10.752,71
9	8.249,42	9.511,79	21.696,18
10	8.324,60	9.675,28	21.993,31
11	8.274,01	9.621,21	22.602,57
12	8.265,54	9.490,23	2.861,98
13	8.182,15	9.520,82	1.954,20
14	8.223,81	9.520,82	3.158,49
15	8.223,81	9.490,97	2.283,04
16	8.294,72	9.865,00	12.843,99
17	8.335,73	17.591,52	2.741,39
18	8.112,36	9.508,24	2.180,85
19	8.352,34	9.657,47	15.636,63
20	8.342,56	9.506,95	2.038,59
21	8.589,92	9.754,03	2.339,04
22	8.410,48	9.492,60	1.763,41
23	12.214,00	9.812,70	1.922,05
24	12.012,06	13.534,81	3.153,84
25	9.992,02	9.498,80	9.684,90
26	9.113,52	9.629,77	3.982,36
27	8.328,96	9.704,60	8.797,64
28	8.221,27	9.491,62	10.100,83
29	8.313,04	9.487,18	2.211,72
30	8.479,08	9.925,46	9.438,01
31	8.261,85	9.492,47	21.752,62
32	8.375,37	9.578,08	15.979,60
33	8.393,37	13.479,32	1.255,03
34	4.319,57	5.450,74	26.370,38
35	4.461,56	5.424,06	26.160,06
36	8.422,00	9.512,99	11.132,11
37	3.981,02	4.780,07	21.667,07
38	3.981,93	4.869,89	13.829,13
39	3.927,67	4.817,23	14.560,16
40	3.928,72	4.775,89	36.485,92
41	19.878,36	16.773,71	21.638,23
42	19.851,26	20.755,92	21.358,79
43	4.465,95	5.423,86	959,56
44	16.405,45	13.515,49	985,82
45	9.623,28	5.432,78	4.932,85
46	4.404,09	5.642,58	6.835,97
47	20.089,29	21.517,24	1.475,04
48	13.364,50	17.647,81	15.555,76
49	16.155,07	21.515,58	37.255,41
Μέση τιμή	9.656	10.871	10.149
Άθροισμα	30.676,26		



Εικόνα 42: MPI_call.cfg - 49 διεργασίες, 4200 grid, zoom

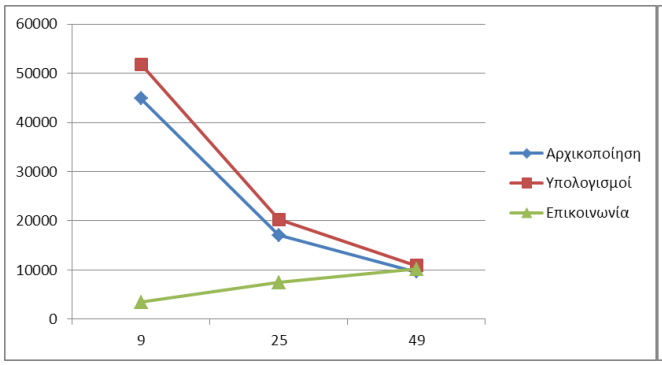
Η γραφική παράσταση του *Σχήματος 1* προκύπτει από τις τιμές του *Πίνακα 4*, όπου ο οριζόντιος άξονας είναι ο αριθμός διεργασιών και ο κατακόρυφος είναι ο χρόνος (σε nanosecond). Γίνεται αμέσως αντιληπτό ότι η αύξηση του κόστους επικοινωνίας αποτελεί φραγμό στη βελτίωση της απόδοσης του προγράμματος, αφού οι χρόνοι υπολογισμού ραγδαία καταλήγουν να είναι της ίδιας τάξης μεγέθους με το κόστος επικοινωνίας.

Πίνακας 4 | Μέσες τιμές

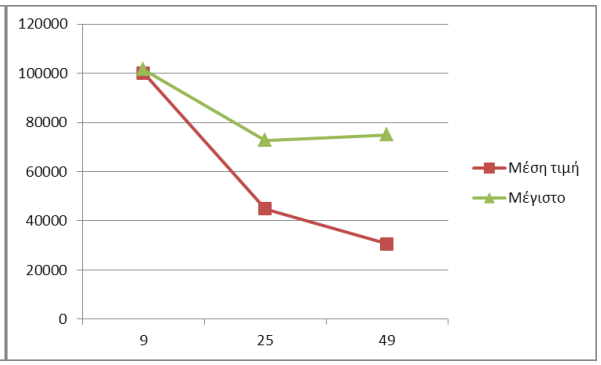
Διεργασίες	9	25	49
Αρχικοποίηση	44872,49	17089,02	9656,034
Υπολογισμοί	51821,81	20291,37	10871,09
Επικοινωνία	3447,569	7420,153	10149,13

Πίνακας 5 | Συνολικοί χρόνοι

Διεργασίες	9	25	49
Μέση τιμή	100141,9	44800,54	30676,26
Μέγιστο	101531,1	72770,33	74926,06



Σχήμα 1: Μέσες τιμές



Σχήμα 2: Συνολικοί χρόνοι

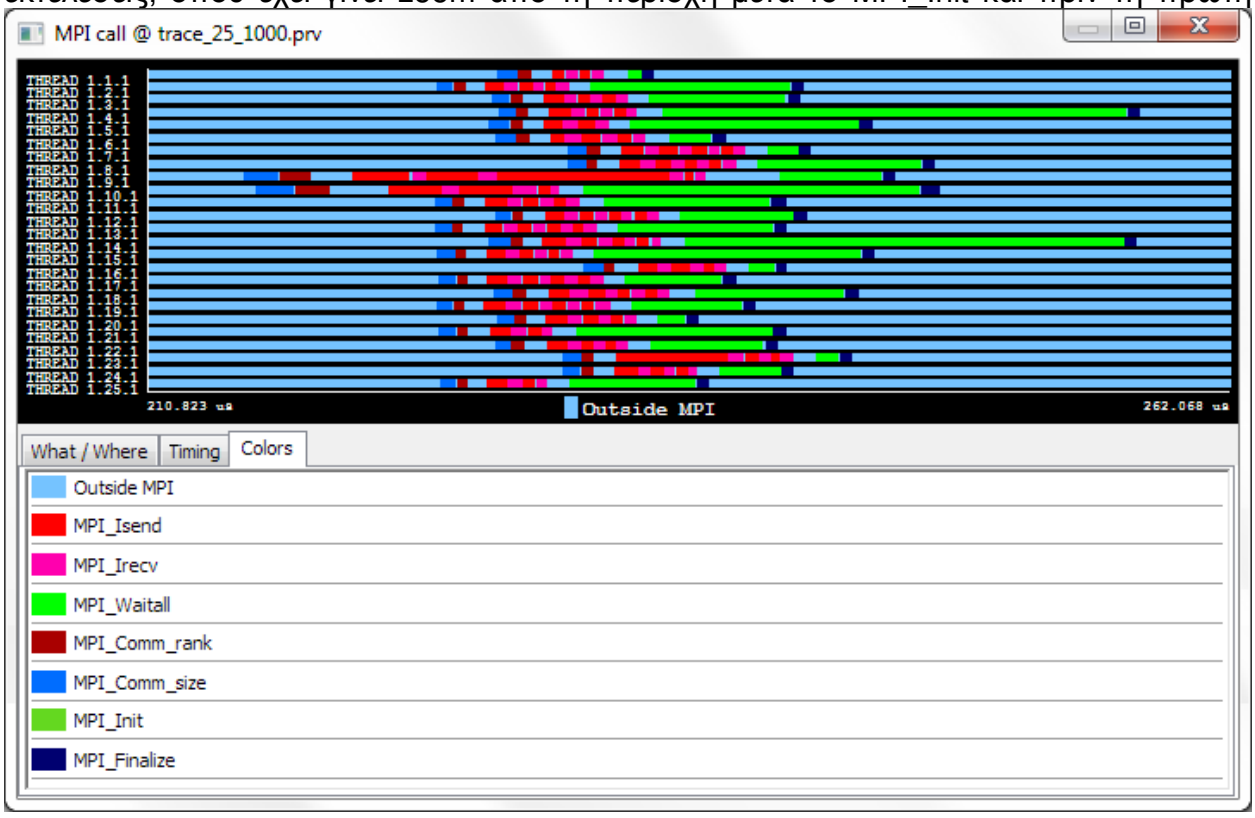
Από το *Πίνακα 5* προκύπτει η γραφική παράσταση του *Σχήματος 2* η οποία προδίδει το γεγονός ότι με την αύξηση του αριθμού διεργασιών οι χρόνοι αποκτούν διασπορά από τη μέση τιμή, η οποία αυξάνεται με την αύξηση του αριθμού διεργασιών. Αυτό είναι σημαντικό διότι και μία μόνο διεργασία να παρουσιάζει μεγάλες τιμές χρόνου είναι αρκετή για να καθυστερήσει την ολοκλήρωση του όλου προγράμματος.

Ένας ακόμα λόγος στον οποίο οφείλεται η μη βελτίωση του χρόνου εκτέλεσης είναι ότι δεν έχει ληφθεί υπόψη η τοπολογία των μηχανών στις οποίες εκτελούνται οι διεργασίες. Έτσι, ενώ η επικοινωνία μεταξύ των διεργασιών είναι τοπική και κάθε μία επικοινωνεί μόνο με τις γειτονικές τις και όχι με όλες, η απόσταση μεταξύ των μηχανών στις οποίες εκτελούνται δύο γειτονικές διεργασίες μπορεί να είναι τέτοια που να δημιουργεί καθυστερήσεις, αφού δύο μηχανές υπάρχει περίπτωση να επικοινωνούν μεταξύ τους μέσω μίας τρίτης. Οι πιθανότητες εμφάνισης του φαινομένου αυτού αυξάνονται όσο αυξάνεται ο αριθμός των διεργασιών και απαιτούνται περισσότερες μηχανές για την εκτέλεση του προγράμματος. Αν γινόταν χρήση τοπολογίας επεξεργαστών τετραγωνικού πλέγματος όπως έχει θεωρηθεί κατά το σχεδιασμό, θα υπήρχε πιθανόν δυνατότητα βελτίωσης του χρόνου εκτέλεσης και για 49 διεργασίες, αλλά τελικά οι αύξηση στο κόστος επικοινωνίας μεταξύ των διεργασιών θα οριοθετούσε και πάλι το βαθμό βελτίωσης του χρόνου εκτέλεσης του προγράμματος.

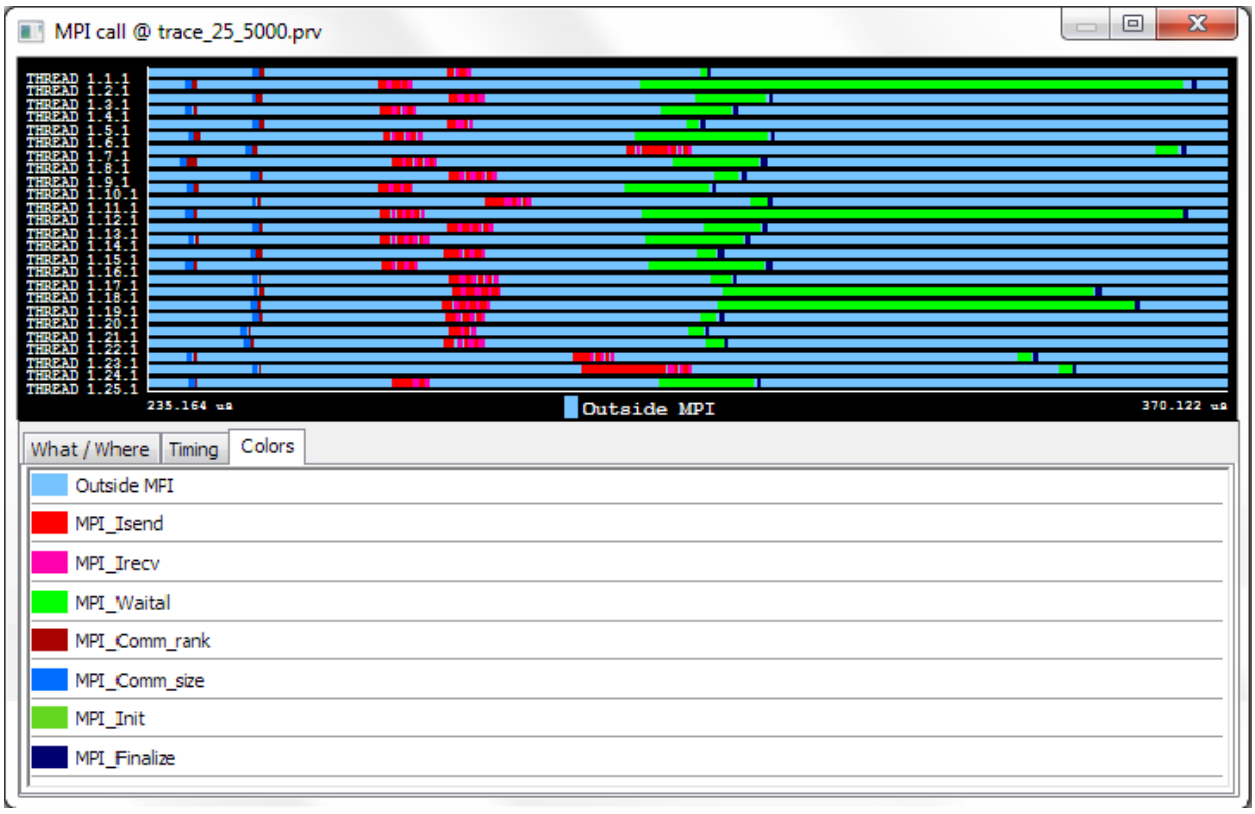
4.2.2 Μεταβαλλόμενος αριθμός υπολογισμών - Σταθερός αριθμός διεργασιών

Σε αυτή τη περίπτωση εκτελείται πάλι το πρόγραμμα τρεις φορές, αλλά διατηρείται σταθερός ο αριθμός διεργασιών, στις 25 διεργασίες, και μεταβάλλεται το μέγεθος του κεντρικού grid. Οι τιμές του grid για κάθε εκτέλεση είναι 1000, 5000 και 10000. Έτσι το μέγεθος του grid κάθε διεργασίας είναι 40, 200 και 400 αντίστοιχα.

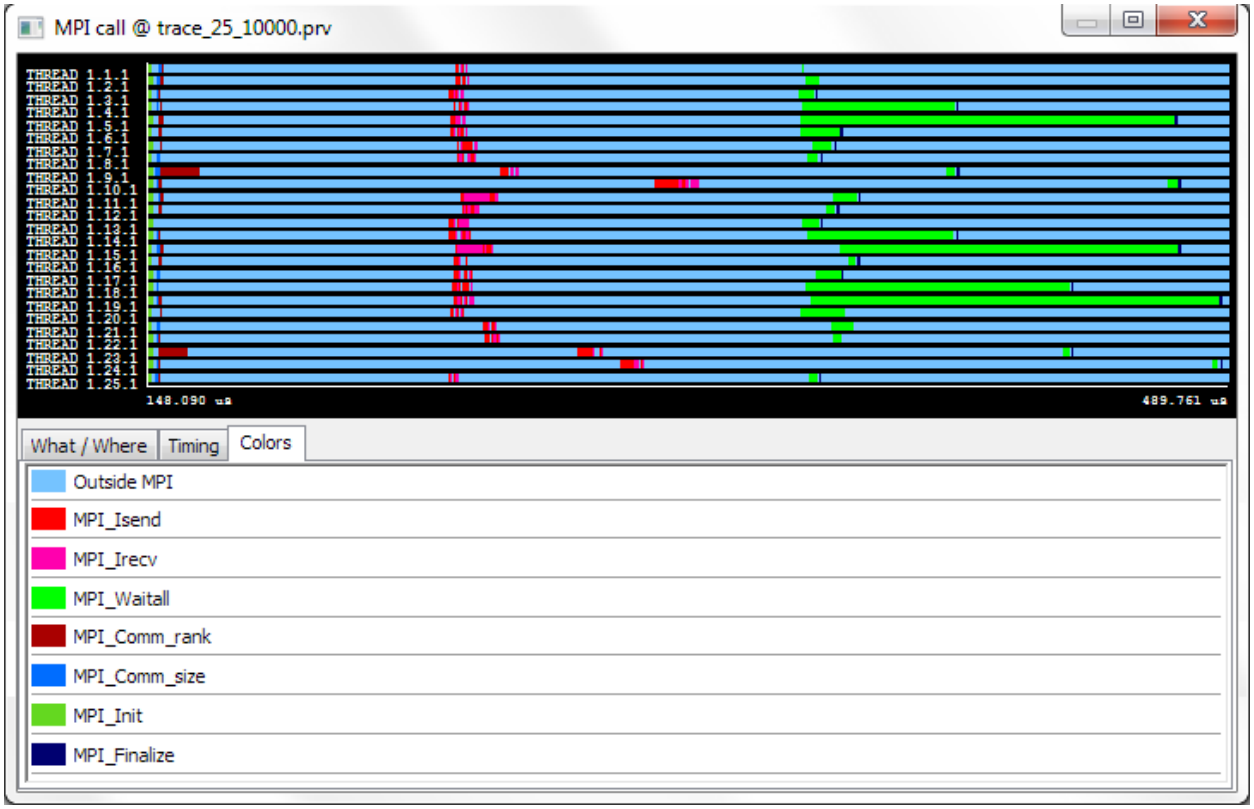
Στις *Εικόνες 43*, *44* και *45* φαίνονται τα γραφήματα για το MPI_call.cfg για τις τρεις εκτελέσεις, όπου έχει γίνει zoom από τη περιοχή μετά το MPI_Init και πριν τη πρώτη



Εικόνα 43: MPI_call.cfg - Μέγεθος πλευράς grid 1000, 25 διεργασίες, zoom



Εικόνα 44: MPI_call.cfg - Μέγεθος πλευράς grid 5000, 25 διεργασίες, zoom



Εικόνα 45: MPI_call.cfg - Μέγεθος πλευράς grid 10000, 25 διεργασίες, zoom

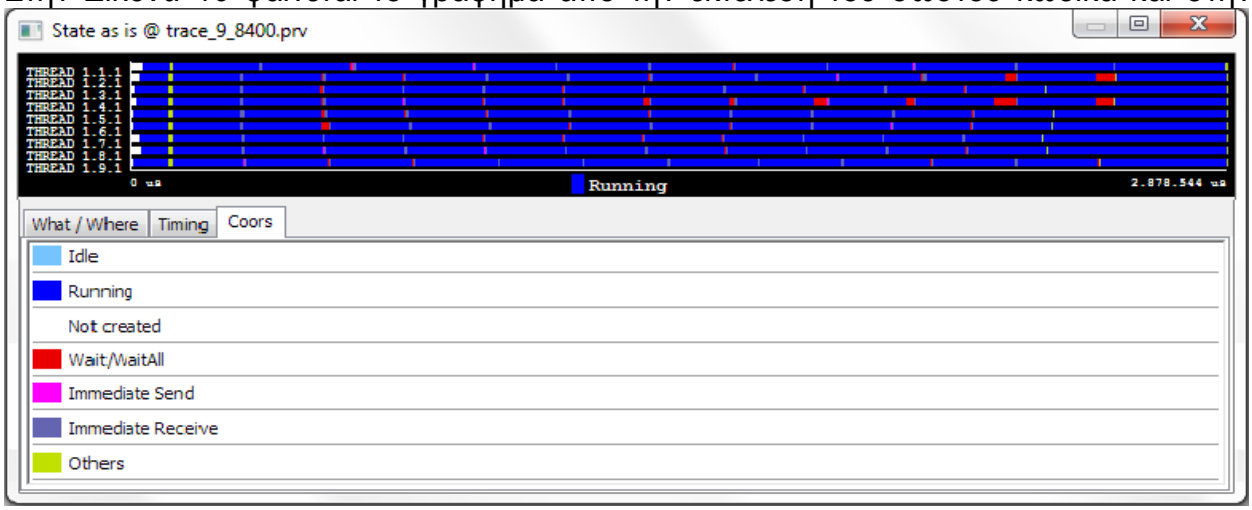
MPI εντολή, έως και το MPI_Finalize. Οι υπολογισμοί που κάνει κάθε διεργασία είναι, μεταξύ του MPI_Comm_rank και του πρώτου MPI_Isend, αρχικοποίηση των πινάκων και μεταξύ του τελευταίου MPI_Irecv και του MPI_Waitall, υπολογισμός των νέων τιμών του grid μέσω του stencil πέντε σημείων. Αύξηση του grid από x' σε $x'+x$ έχει ως

αποτέλεσμα την αύξηση των υπολογισμών κατά $x^2 + 2*x*x'$, αλλά αύξηση των δεδομένων προς ανταλλαγή μεταξύ των διεργασιών κατά x. Για το λόγο αυτό παρατηρείται έντονη αύξηση των χρόνων υπολογισμού αλλά ίδιους χρόνους για την αποστολή των δεδομένων. Η αύξηση των χρόνων υπολογισμού όμως έχει αντίκτυπο στους χρόνους των MPI_Waitall, οι οποίοι παρατηρώντας και τις τιμές στον άξονα του χρόνου, προκύπτει ότι αυξάνονται αναλογικά με τους χρόνους υπολογισμού. Στη περίπτωση αυτή η συμπεριφορά του προγράμματος είναι η αναμενόμενη και δεν παρουσιάζονται ιδιοτροπίες ή άλλου είδους προβλήματα.

4.2.3 Εσκεμμένο λάθος στο κώδικα του προγράμματος

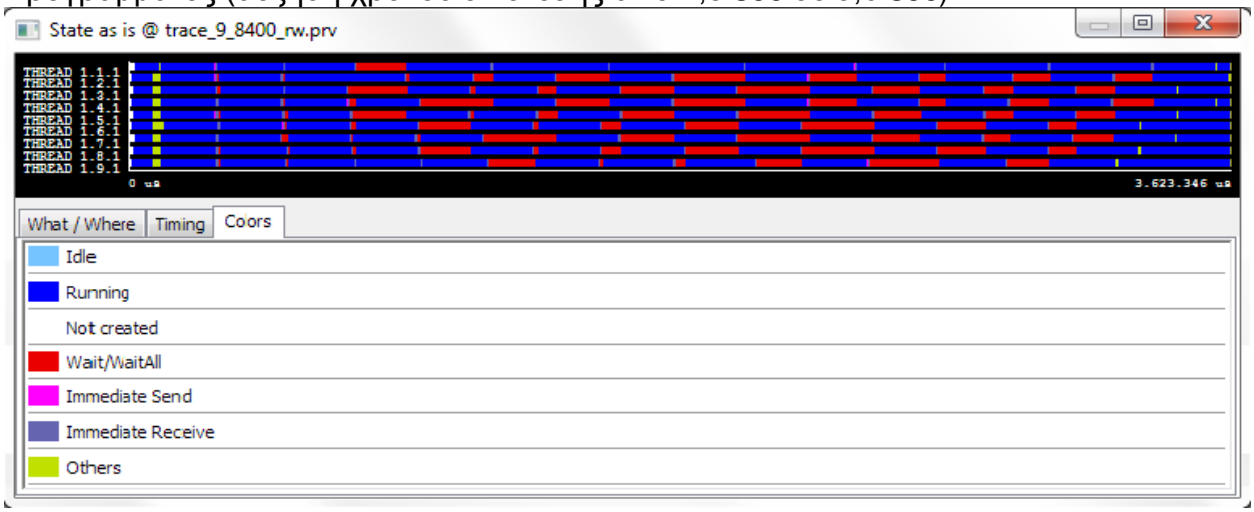
Το πρώτο λάθος που δημιουργήθηκε στο κώδικά είναι η τοποθέτηση της εντολής MPI_Waitall πριν από τον υπολογισμό των νέων τιμών του εσωτερικού του πίνακα δεδομένων. Επίσης έχει προστεθεί στο πρόγραμμα χρονική εξέλιξη 10 βημάτων, ώστε να αποτυπώνονται καλύτερα οι διαφορές στα γραφήματα του Paraver. Το πρόγραμμα εκτελέστηκε για μέγεθος κεντρικού πλέγματος 8400 και 9 διεργασίες.

Στην *Εικόνα 46* φαίνεται το γράφημα από την εκτέλεση του σωστού κώδικα και στην



Εικόνα 46: state_as_is.cfg - Σωστός κώδικας

Εικόνα 47 το γράφημα από την εκτέλεση του λανθασμένου κώδικα. Είναι εμφανής η διαφορά στην απόδοση του προγράμματος, με το λανθασμένο κώδικα να παρουσιάζει μεγάλες καθυστερήσεις που πλησιάζουν το 1/3 του χρόνου εκτέλεσης του σωστού προγράμματος (αύξηση χρόνου εκτέλεσης από 2,8 sec σε 3,6 sec).



Εικόνα 47: state_as_is.cfg - Λανθασμένος κώδικας

5. CONFIGURATION FILES ΣΧΕΤΙΚΑ ΜΕ ΤΟ PAPI

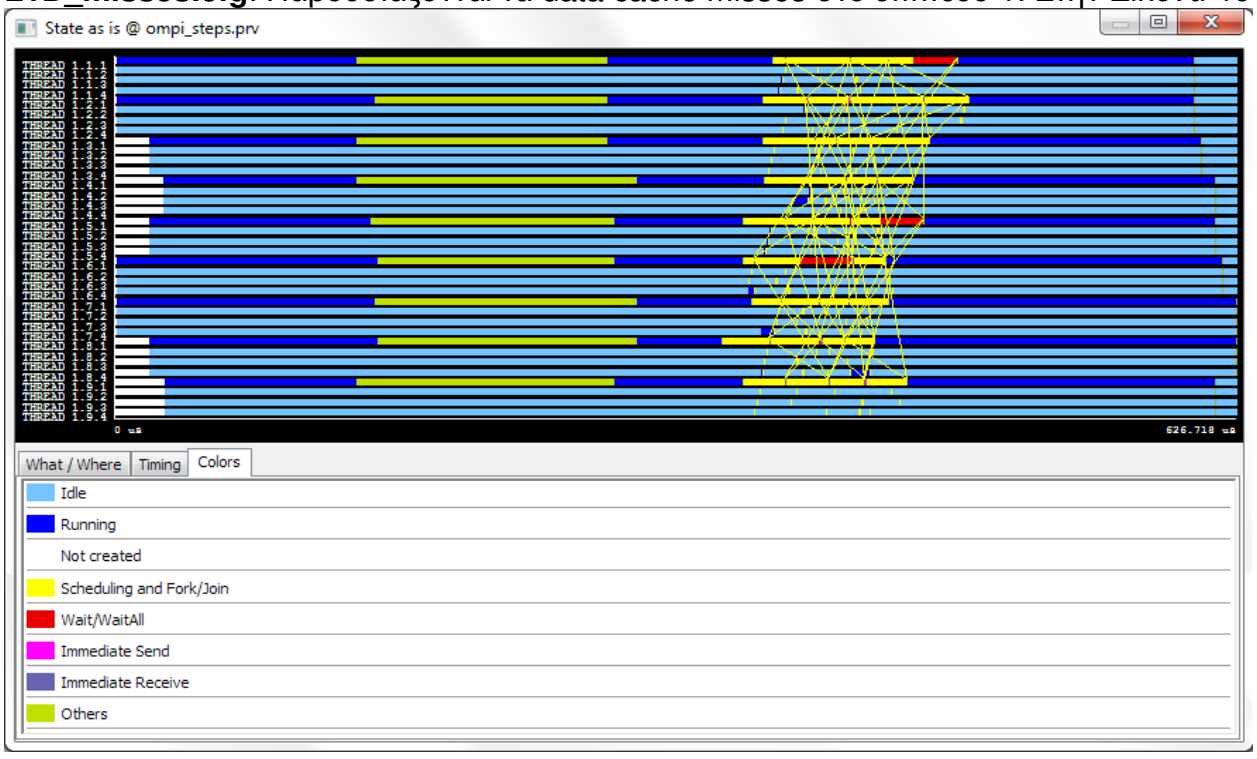
Οι επεξεργαστές των μηχανών στο εργαστήριο Linux του τμήματος Πληροφορικής και Τηλεπικοινωνιών του ΕΚΠΑ, δεν υποστηρίζουν μεγάλη γκάμα από μετρητές και δεν μπορούν να χρησιμοποιηθούν τα configuration files του Paraver που είναι σχετικά με το PAPI. Για το λόγο αυτό τα γραφήματα του Paraver που θα παρουσιαστούν στο κεφάλαιο αυτό, δημιουργήθηκαν με τη χρήση μιας μηχανής εκτός του τμήματος με επεξεργαστή που υποστηρίζει διπλάσιους μετρητές (λεπτομέρειες στο Παράρτημα ΙΙΙ). Το πρόγραμμα που χρησιμοποιήθηκε για τη δημιουργία των traces είναι το ίδιο με αυτό του κεφαλαίου 4, με τη μόνη διαφορά τη χρήση OpenMP για την αρχικοποίηση του πίνακα δεδομένων και για τους υπολογισμούς των νέων τιμών στο εσωτερικό του πίνακα.. Εκτελέστηκε με 9 διεργασίες, μέγεθος πλευράς πίνακα δεδομένων 840 και προστέθηκε χρονική εξέλιξη τριών βημάτων. Οι μετρητές που χρησιμοποιήθηκαν στο extrae.xml είναι:

PAPI_TOT_INS	Instructions completed	PAPI_L1_STM	Level 1 store misses
PAPI_TOT_CYC	Total cycles	PAPI_STL_ICY	Cycles with no instruction issue
PAPI_L1_DCM	Level 1 data cache misses	PAPI_LD_INS	Load instructions
PAPI_L1_LDM	Level 1 load misses	PAPI_SR_INS	Store instructions

Ο μέγιστος αριθμός μετρητών που μπορούν να υποστηριχτούν ταυτόχρονα από το Extrae είναι 6. Για το λόγο αυτό πρέπει να δημιουργούνται διαφορετικά trace αρχεία ανάλογα το configuration file και τους μετρητές που αυτό απαιτεί. Επίσης για καλύτερη κατηγοριοποίηση και παρουσίαση των δεδομένων, θα πρέπει ο χρήστης να εκτιμά κατά περίπτωση, με τη βοήθεια του Info Panel, τις καταλληλότερες τιμές για το εύρος τιμών, το οποίο αναπαρίσταται μέσω των χρωμάτων στα γραφήματα. Αυτό μπορεί να γίνει από το menu στο κεντρικό παράθυρο του Paraver, όπως φαίνεται στην *Εικόνα 4*, και αλλάζοντας τις τιμές στα **Semantic minimum** και **Semantic maximum**.

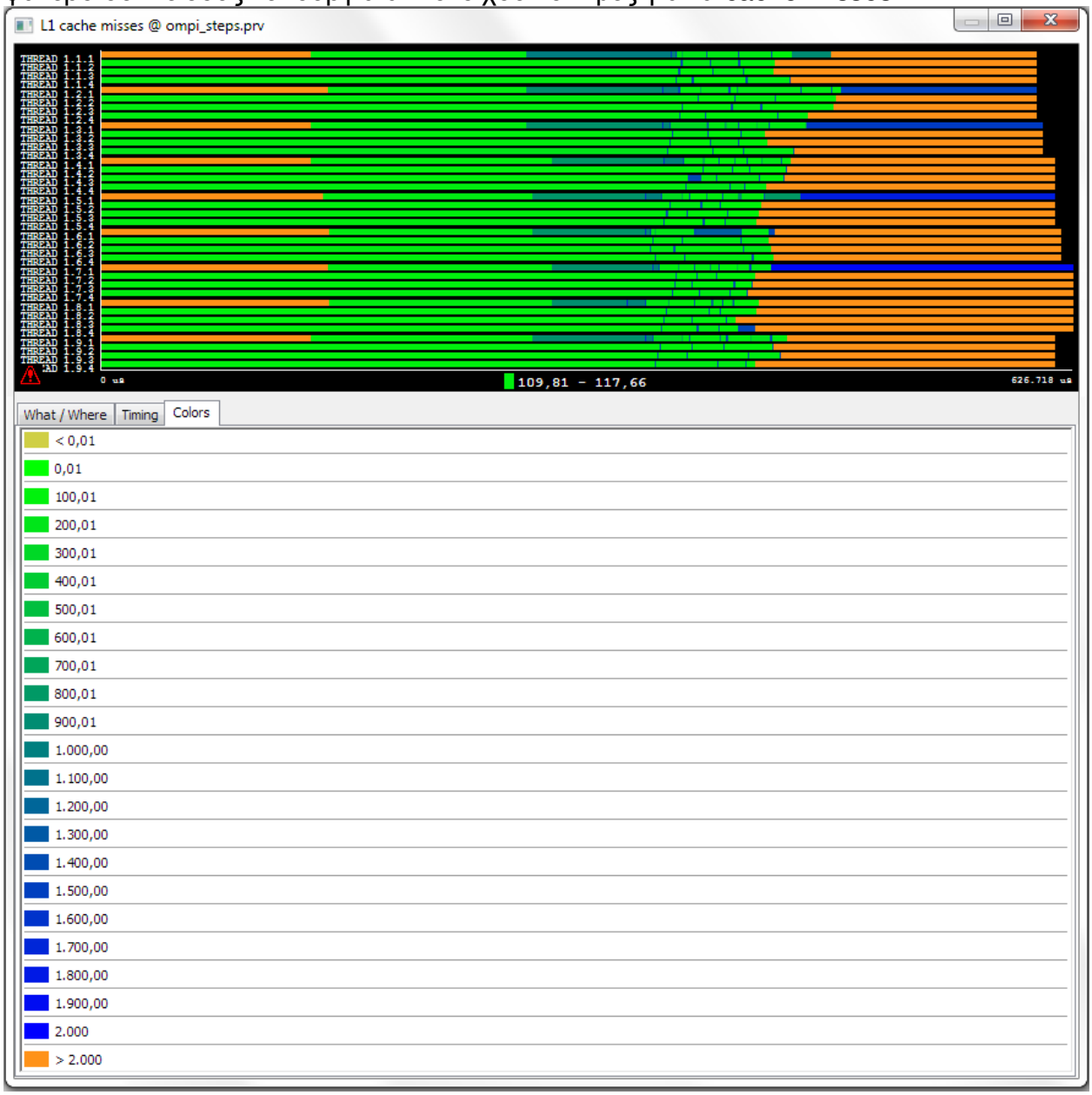
Φάκελος counters_PAPI/architecture

L1D_misses.cfg: Παρουσιάζονται τα data cache misses στο επίπεδο 1. Στην *Εικόνα 48*



Εικόνα 48: state_as_is.cfg

παρουσιάζεται αρχικά το γράφημα για το state_as_is.cfg, διότι έχει ενδιαφέρον η αντιπαράθεσή του με το γράφημα L1D_misses.cfg της *Εικόνας 49*, οπότε και γίνεται φανερό σε τι είδους λειτουργία αντιστοιχούν οι τιμές για τα cache misses.



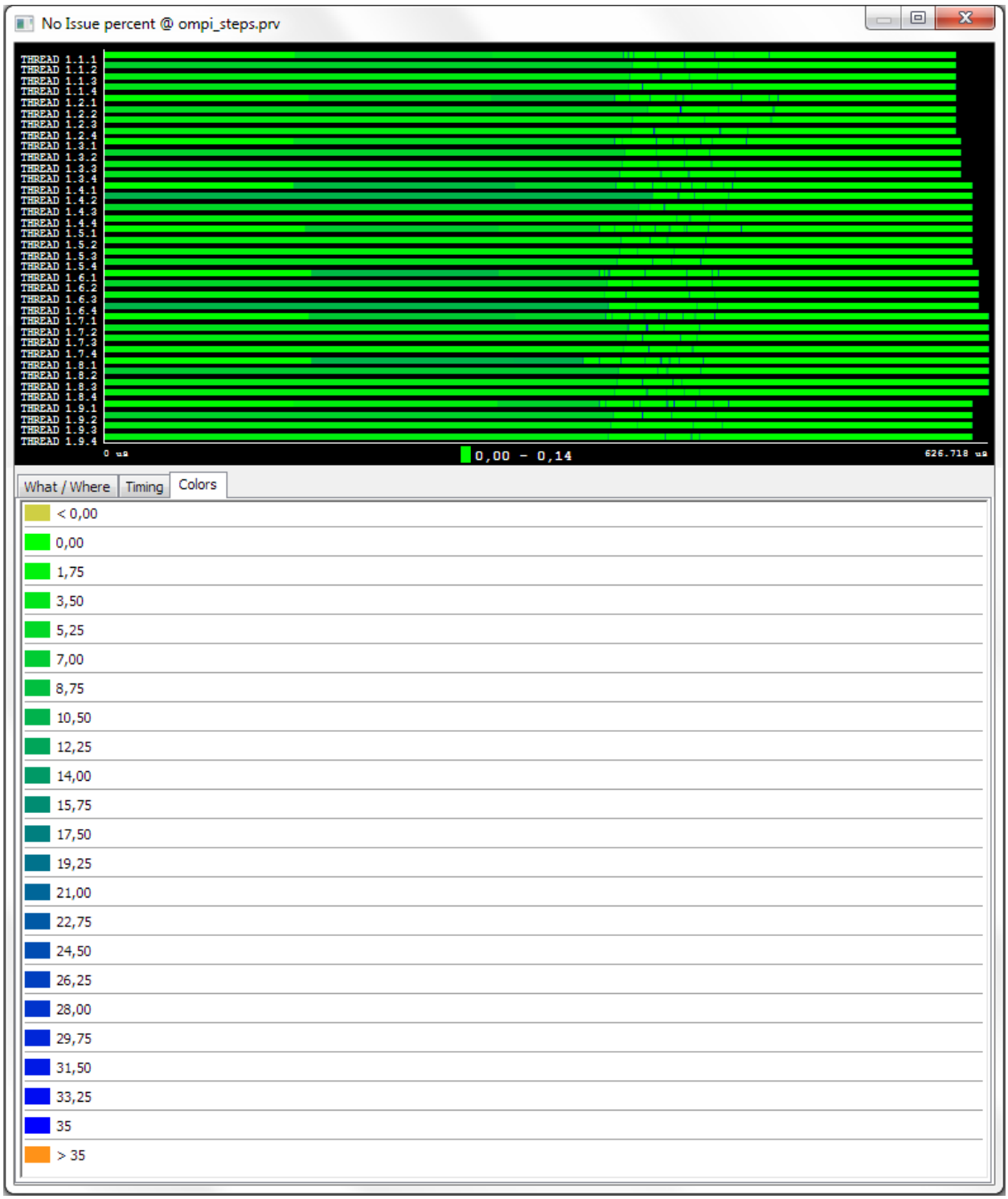
Εικόνα 49: L1D_misses.cfg

L1D_missratio: Παρουσιάζεται ο λόγος των data cache misses προς τον αριθμό των instructions.

L1_Load_misses.cfg: Παρουσιάζονται τα load misses στο επίπεδο 1 της cache. Χρειάζεται η προσθήκη του μετρητή PAPI_L1_LDM στο αρχικό extrae.xml.

L1_Store_misses.cfg: Παρουσιάζονται τα store misses στο επίπεδο 1 της cache. Χρειάζεται η προσθήκη του μετρητή PAPI_L1_STM στο αρχικό extrae.xml.

No_issue_percent.cfg: Παρουσιάζεται το ποσοστό τοις εκατό των κύκλων χωρίς instruction σε σχέση με το συνολικό αριθμό των κύκλων (*Εικόνα 50*). Χρειάζεται η προσθήκη του μετρητή PAPI_STL_ICY στο αρχικό extrae.xml.

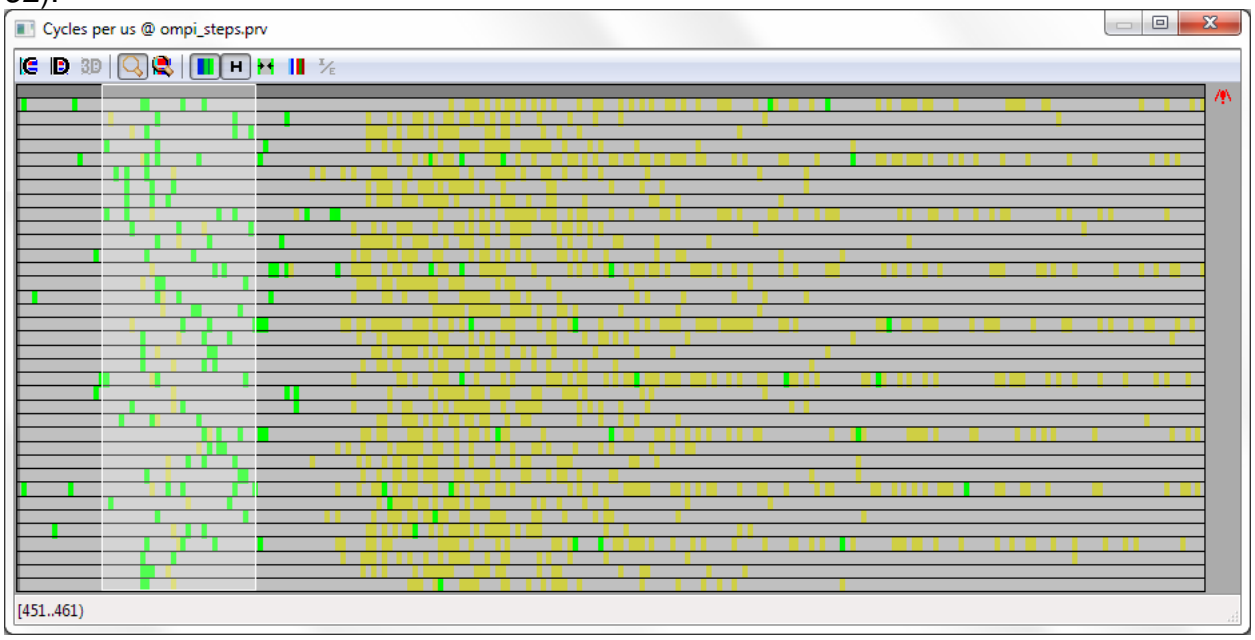


Εικόνα 50: No_issue_percent.cfg

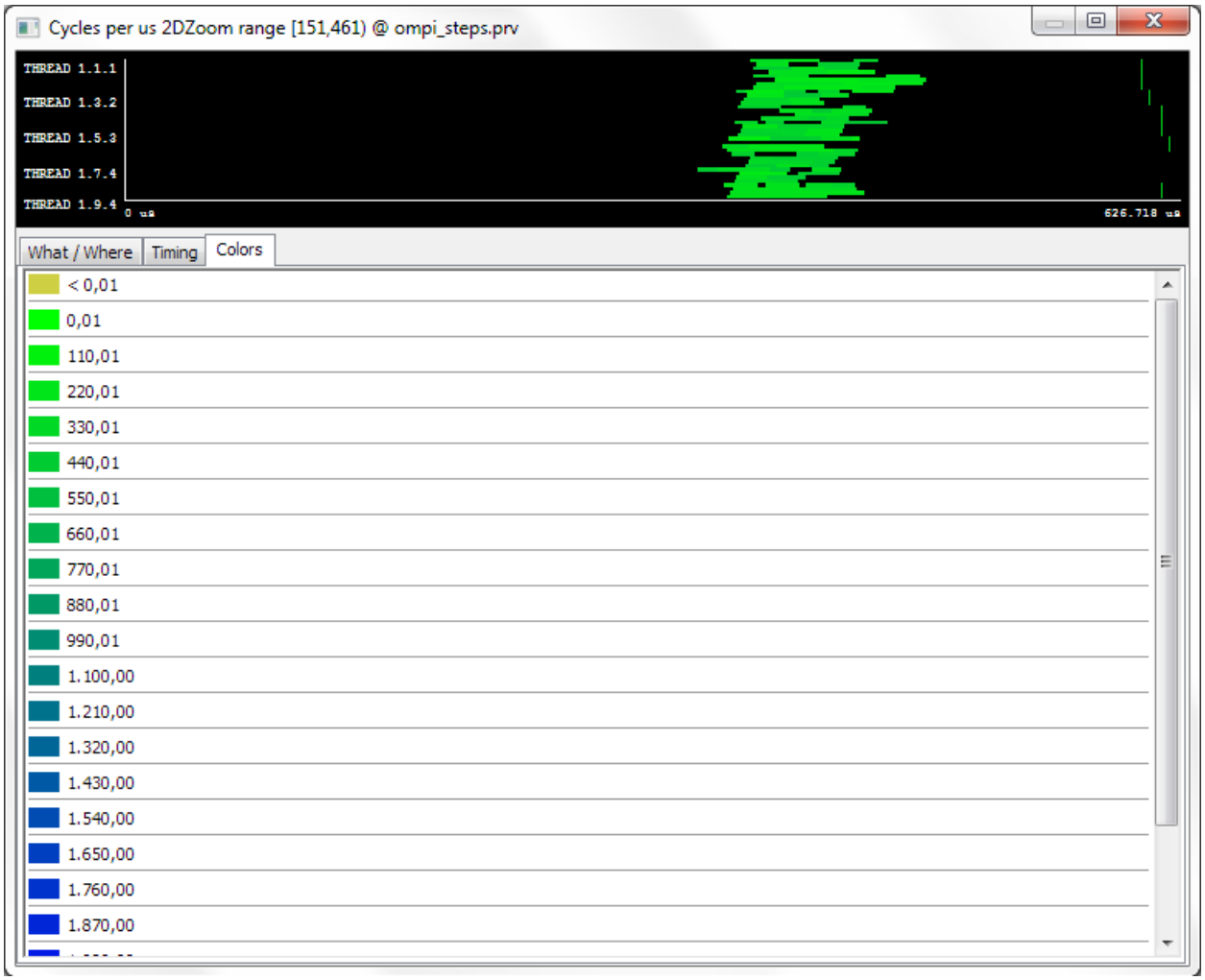
Φάκελος counters_PAPI/performance

3dh_cycles_per_us.cgf: Παρουσιάζεται ο αριθμός των κύκλων ανά μικροsecond. Στον οριζόντιο άξονα του πίνακα κάθε στήλη αντιστοιχεί σε ένα εύρος τιμών. Σε κάθε κελί η τιμή αντιστοιχεί στον αριθμό των περιπτώσεων, κατά την εκτέλεση του προγράμματος, που οι κύκλοι ανά μικροsecond ανήκουν στο συγκεκριμένο εύρος τιμών. Τα κελιά με κίτρινο χρώμα αντιστοιχούν σε πολύ μικρές αποστάσεις μεταξύ των δειγμάτων (μικρότερες από 100 μικροsecond), όπου η ακρίβεια καταγραφής των δεδομένων πιθανών να μην επαρκεί και να τα αλλοιώνει σημαντικά.

Με το κουμπί  και επιλέγοντας στήλες, (*Εικόνα 51*) δημιουργείται το σχετικό γράφημα που αποτυπώνει σε ποιο σημείο τις εκτέλεσης αντιστοιχούν οι τιμές των κελιών (*Εικόνα 52*).



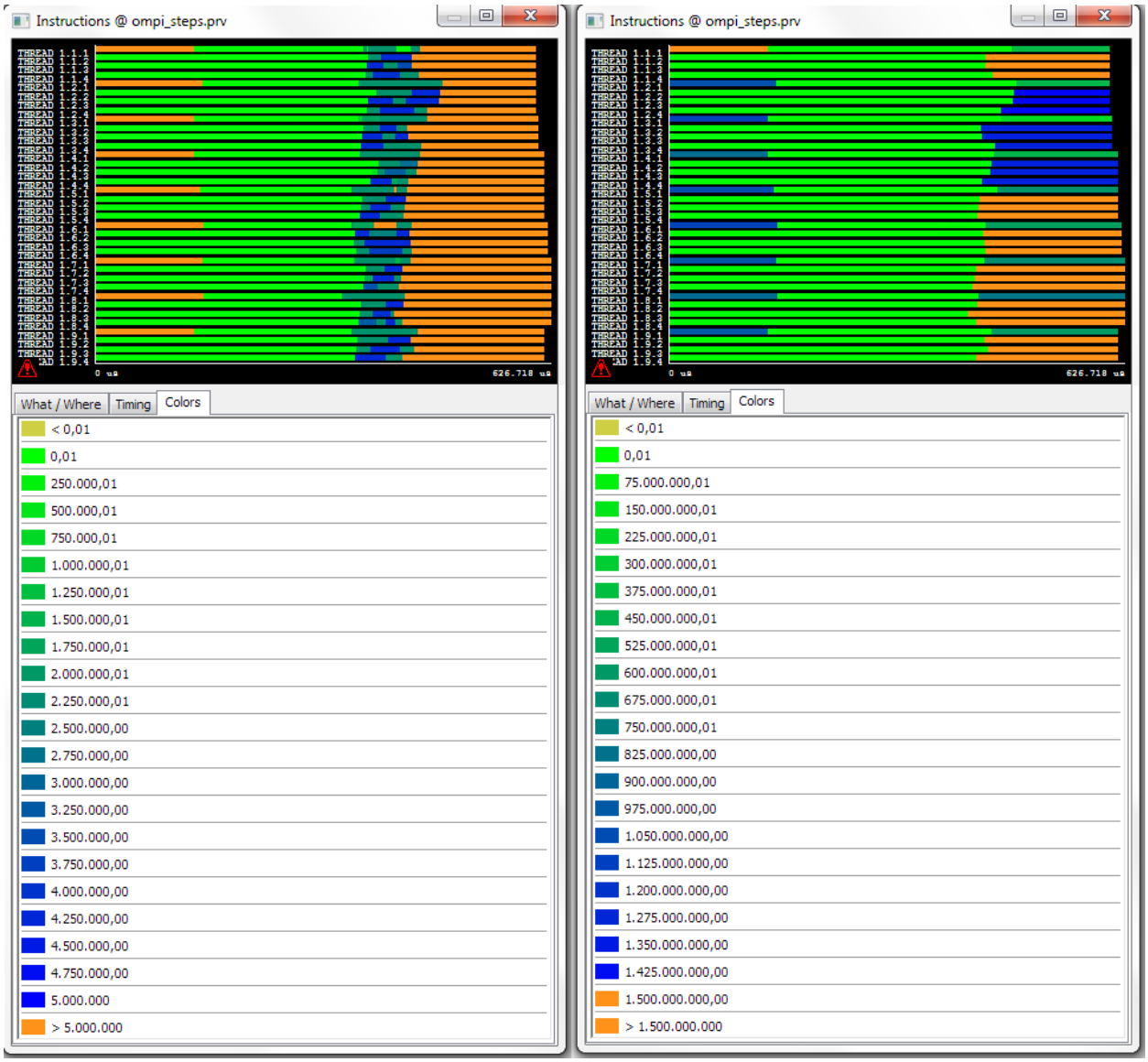
Εικόνα 51: 3dh_cycles_per_us.cfg



Εικόνα 52: Cycles per μικροsecond - 2DZoom

IPC.cfg: Instructions per cycle, εντολές ανά κύκλο.

instructions.cfg: Αριθμός εντολών. Στην *Εικόνα 53* φαίνεται το ίδιο γράφημα με διαφορετική μέγιστη τιμή για το εύρος τιμών το οποίο αναπαρίσταται με χρώματα, όπως φαίνεται από το info panel. Η χρήση του ενός μόνο από τα δύο γραφήματα δεν μπορεί να αποτυπώσει ακριβή εικόνα και είναι απαραίτητη η παραμετροποίηση του γραφήματος ανάλογα την ανάγκη της εκάστοτε μελέτης.



Εικόνα 53: instruction.cfg - Διαφορετική τιμή semantic maximum

L_S_mix.cfg: Αριθμός load και store εντολών ανά 1000 κύκλους. Χρειάζεται η προσθήκη των μετρητών PAPI_LD_INS και PAPI_SR_INS στο αρχικό extrae.xml.

Loads.cfg: Αριθμός load εντολών. Χρειάζεται η προσθήκη του μετρητή PAPI_LD_INS στο αρχικό extrae.xml.

Store.cfg: Αριθμός store εντολών. Χρειάζεται η προσθήκη του μετρητή PAPI_SR_INS στο αρχικό extrae.xml.

6. ΣΥΜΠΕΡΑΣΜΑΤΑ

Το BSC έχει δημιουργήσει δύο πολύ εύχρηστα και αποδοτικά εργαλεία για τη μελέτη παράλληλων προγραμμάτων σχετικά με την απόδοσή τους. Αρχικά, η χρήση του Extrae για τη δημιουργία των trace αρχείων που περιέχουν τις μετρήσεις που απαιτούνται για τη μελέτη ενός προγράμματος είναι απλή και σχετίζεται με τη χρήση δύο αρχείων. Το πρώτο είναι ένα script αρχείο το οποίο εκτελείται μαζί με το προς μελέτη πρόγραμμα και περιέχει τις παραμέτρους εκτέλεσης του Extrae, δηλαδή τις βιβλιοθήκες των οποίων θα κάνει χρήση και έχουν δημιουργηθεί από το κατασκευαστή, αλλά και τη τοποθεσία του αρχείου με τις παραμέτρους λειτουργίας, που παραμετροποιεί ο χρήστης. Το δεύτερο αυτό αρχείο, που παραμετροποιεί ο χρήστης, έχει μορφή xml και περιέχει λεπτομέρειες σχετικά με το τι είδους μετρήσεις θα αποθηκευτούν στα τελικά παραγόμενα αρχεία και η πολυπλοκότητά του εξαρτάται από τις ανάγκες του χρήστη.

Τα παραγόμενα αρχεία από το Extrae φορτώνονται στο δεύτερο εργαλείο, το Paraver, το οποίο οπτικοποιεί τα δεδομένα με τη χρήση γραφημάτων και πινάκων. Παρέχεται, επίσης μια ευρεία γκάμα επιλογών για το είδος των δεδομένων και το τρόπο που παρουσιάζονται μέσω έτοιμων αρχείων, επονομαζόμενων configuration files ή εν συντομία cfgs. Τα αρχεία αυτά περιέχουν τις απαραίτητες ρυθμίσεις ώστε το Paraver να αντλήσει από τα αρχεία trace τα δεδομένα που θέλει ο χρήστης. Επιπλέον, υπάρχει μεγάλος αριθμός επιλογών σχετικά με τις μονάδες, τη κλίμακα, το επίπεδο λειτουργίας (εφαρμογή, διεργασία κ.τ.λ.) και το τμήμα του προγράμματος που θέλει να χρησιμοποιήσει ο χρήστης για τη μελέτη. Έτσι, μπορούν να εξεταστούν σε βάθος όλες οι λεπτομέρειες της συμπεριφοράς ενός προγράμματος κατά την εκτέλεσή του και να αναγνωριστούν επακριβώς τα σημεία και οι λόγοι καθυστέρησης ή χαμηλής απόδοσης.

Ταυτόχρονα με τα παραπάνω, υπάρχει δυνατότητα μέτρησης της λειτουργίας του συστήματος από τους μετρητές του επεξεργαστή μέσω της χρήσης του PAPI. Το Extrae κάνει χρήση του PAPI συλλέγοντας δεδομένα στα αρχεία trace από τους μετρητές που ορίζει ο χρήστης στο xml αρχείο ρυθμίσεων του Extrae. Για το Paraver υπάρχουν configuration files για τη κατηγορία αυτή των μετρήσεων με τις ίδιες δυνατότητες επιλογής μονάδων, κλίμακας κ.τ.λ. από το χρήστη.

Συνοψίζοντας, η χρήση των Extrae και Paraver προσφέρει μια ολοκληρωμένη, λεπτομερής και οπτικοποιημένη πληροφόρηση για τη λειτουργία των παράλληλων προγραμμάτων, που μπορεί να βοηθήσει τους προγραμματιστές να εντοπίσουν γρήγορα τα λανθασμένα και μη αποδοτικά τμήματα ενός κώδικα και να τον βελτιώσουν βοηθώντας στη κλιμάκωση του προγράμματος.

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

BSC	Barcelona Supercomputing Center
cfgs	configuration files
ΕΚΠΑ	Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΠΑΡΑΡΤΗΜΑ Ι - Εγκατάσταση και παραμετροποίηση Extrae

Εγκατάσταση

Πριν γίνει εγκατάστασή του Extrae απαιτούνται εγκαταστάσεις ορισμένων εξαρτήσεων (dependencies). Παρακάτω οι αντίστοιχες εντολές για το terminal:

```
sudo apt-get install libelfg0-dev
sudo apt-get install libdwarf-dev
sudo apt-get install g++
sudo apt-get install libboost-all-dev
sudo apt-get install libgtk2.0-dev
sudo apt-get install gfortran
sudo apt-get install libunwind8-dev
sudo apt-get install libxml++2.6-dev
sudo apt-get install liblasso3-dev
sudo apt-get install binutils-dev
```

Επίσης πρέπει να γίνει εγκατάσταση των παρακάτω:

- Libdwarf
  Πριν προχωρήσετε στη παρακάτω διαδικασία ελέγξτε αν υπάρχουν στο φάκελο `/usr/include` τα αρχεία `libdwarf.h` και `dwarf.h` και στο φάκελο `/usr/lib` το αρχείο `libdwarf.so`. Αν υπάρχουν και τα τρία τότε μπορείτε να προχωρήσετε στην εγκατάσταση του Dyninst. Αν κάποιο από αυτά δεν υπάρχει τότε προχωρήστε στη παρακάτω διαδικασία. Επίσης αν κάποια από τα τρία παραπάνω αρχεία υπάρχουν ήδη δεν χρειάζεται να τα αντικαταστήσετε.
  Λήψη από *www.filewatcher.com/m/libdwarf-20130207.tar.gz.1534527-0.html*
  Μέσω του terminal:
  `cd` <path of downloaded file> (π.χ. `cd /home/john/Downloads`)
  `tar -xvzf libdwarf-20130207.tar.gz`
  `cd` <path of unzipped folder>`/libdwarf`
  `sudo ./configure --enable-shared`
  `sudo make`
  `sudo cp libdwarf.h /usr/include/libdwarf.h` (αν δεν υπάρχει ήδη)
  `sudo cp dwarf.h /usr/include/dwarf.h` (αν δεν υπάρχει ήδη)
  `sudo cp libdwarf.so /usr/lib/libdwarf.so` (αν δεν υπάρχει ήδη)
- DyninstAPI8.1.2
  Λήψη από *www.dyninst.org/downloads/dyninst-8.x*
  Μέσω του terminal:
  `cd` <path of downloaded file> (π.χ. `cd /home/john/Downloads`)
  `tar -xvzf DyninstAPI-8.1.2.tgz`
  `cd` <path of unzipped folder>
  `sudo ./configure --prefix=`<path to install> (να σημειωθεί το path που θα γίνει η εγκατάσταση καθώς θα χρειαστεί αργότερα)
  `sudo make`
  `sudo make install`
- PAPI
  Λήψη από το *icl.cs.utk.edu/papi/software/index.html*
  Μέσω του terminal:
  `cd` <path of downloaded file> (π.χ. `cd /home/john/Downloads`)
  `tar -xvzf papi-5.3.0.tar.gz`
  `cd` <path of unzipped folder>`/src`
  `sudo ./configure --prefix=`<path to install> (να σημειωθεί το path που θα γίνει η εγκατάσταση καθώς θα χρειαστεί αργότερα)

```
sudo make
sudo make install
```

- MPICH3
  Λήψη από *www.mpich.org/downloads/*
  Μέσω του terminal:
  `cd` <path of downloaded file> (π.χ. cd /home/john/Downloads)
  `tar -xvzf mpich-3.1.3.tar.gz`
  `cd` <path of unzipped folder>
  `sudo ./configure --prefix=`<path to install> (να σημειωθεί το path που θα γίνει η εγκατάσταση καθώς θα χρειαστεί αργότερα)
  `sudo make`
  `sudo make install`

Για την εγκατάσταση του Extrae η λήψη του SOURCE αρχείου γίνεται από:

http://www.bsc.es/computer-sciences/performance-tools/downloads.

`cd` <path of downloaded file> (π.χ. `cd /home/john/Downloads`)
`tar -xvjf extrae-2.5.1.tar.bz2`
`cd` <path of unzipped folder>
`sudo ./configure --prefix=`<path to install> `--with=elf=/usr --with-dwarf=/usr --with-unwind=/usr --with-dyninst=`<path of dyninst installation> `--with-papi=`<path of papi installation> `--with-mpi=`<path of mpi installation> `--enable-openmp`
`sudo make`
`sudo make install`

Παρέχεται η δυνατότητα, μετά την εγκατάσταση, να γίνει ρύθμιση σχετικά με το που έχουν εγκατασταθεί οι βιβλιοθήκες που χρειάζεται το Extrae, αν δεν έχει διευκρινιστεί με την εντολή `./configure`. Αυτό μπορεί να γίνει με τη παρακάτω εντολή:

<path of extrae>`/bin/extrae-post-installation-upgrade.sh`

Το σύστημα θα κάνει κάποιες ερωτήσεις σχετικά με το που είναι εγκατεστημένες διάφορες βιβλιοθήκες δείχνοντας τις αποθηκευμένες τιμές, αν υπάρχουν, και προσφέροντας τη δυνατότητα τροποποίησής τους. Επίσης υπάρχει η δυνατότητα παράθεσης των ρυθμίσεων με τις οποίες έγινε εγκατάσταση του Extrae με την εντολή:

<path of extrae>`/etc/configured.sh`

Παράμετροι λειτουργίας

Οι παράμετροι λειτουργίας του Extrae παρέχονται μέσω ενός αρχείου xml . Το αρχείο αυτό απαρτίζεται από διάφορα nodes με το καθένα από αυτά να έχει το δικό του enable, ώστε να μπορεί να γίνει η παραμετροποίηση όπως ακριβώς θέλει ο χρήστης.

Το πρώτο τμήμα του .xml δεν πρέπει να τροποποιείται και αναφέρεται στην έκδοση:
<?xml version='1.0'?>

Το κεντρικό node ρυθμίζεται από το χρήστη και είναι το παρακάτω:
<trace enabled="yes" home="@sub_PREFIXDIR@" initial-mode="detail" type="paraver" xml-parser-id="@sub_XMLID@">

- Trace enabled="yes" για δημιουργία trace αρχείου.
- home: ο φάκελος που έχει γίνει εγκατάσταση του Extrae. Συνήθως δε τροποποιείται και δείχνει όπου δείχνει και η μεταβλητή περιβάλλοντος EXTRAE_HOME, η οποία καθορίζεται μέσω του terminal.
- initial-mode:
  - detail: συμπεριλαμβάνονται όλες οι λεπτομέρειες.
  - bursts: δεν συμπεριλαμβάνονται στοιχεία για MPI, openmp αλλά μόνο στοιχεία για υπολογισμούς.
- type: το πρόγραμμα στο οποίο θα φορτωθεί το εξαγόμενο αρχείο του Extrae.
- xml-parcer-id: έλεγχος συμβατότητας.

Τα εμφωλευμένα nodes, επίσης, ρυθμίζονται από το χρήστη και είναι τα παρακάτω:

- **<mpi enabled="yes">**
  <counters enabled="yes"/>
  </mpi>
  Συλλογή πληροφοριών στην αρχή και στο τερματισμό των MPI calls.
- **<openmp enabled="yes">**
  <locks enabled="no"/>
  <counters enabled="yes"/>
  </openmp>
  Συλλογή πληροφοριών για τους χρόνους τρεξίματος του OpenMP. Επίσης, αν θα γίνεται συλλογή πληροφοριών για κλειδώματα και από τους μετρητές απόδοσης.
- **<pthread enabled="no">**
  <locks enabled="no"/>
  <counters enabled="yes"/>
  </pthread>
  Πληροφορίες για pthreads.
- **<callers enabled="yes">**
  <mpi enabled="yes">
  1-3
  </mpi>
  <sampling enabled="no">
  1-5
  </sampling>
  </callers>
  Είναι οι διευθύνσεις που βρίσκονται στη στοίβα όσο η εφαρμογή τρέχει. Συνδέουν το αρχείο trace με το πηγαίο κώδικα της εφαρμογής.
- **<user-functions enabled="no" list="/home/bsc41/bsc41273/user-functions.dat" exclude-automatic-functions="no">**
- **<counters enabled="yes"/>**
  </user-functions>

Μέσω αρχείου παρέχεται λίστα με συγκεκριμένες συναρτήσεις για τις οποίες το Extrae θα δημιουργήσει το trace αρχείο. Περισσότερες λεπτομέρειες και τρόποι υλοποίησης αναφέρονται στο user-guide.pdf του Extrae.

- **<counters enabled="yes">**
  <cpu enabled="yes" starting-set-distribution="1">
  <set enabled="yes" domain="all" changeat-globalops="5">
  PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_L1_DCM
  <sampling enabled="no" frequency="100000000">
  PAPI_TOT_CYC
  </sampling>
  </set>
  <set enabled="yes" domain="user" changeat-globalops="5">
  PAPI_TOT_INS,PAPI_FP_INS,PAPI_TOT_CYC
  </set>
  </cpu>
  <network enabled="no"/>
  <resource-usage enabled="no"/>
  <memory-usage enabled="no"/>
  </counters>

  Συλλογή πληροφοριών από τους μετρητές του συστήματος. Απαραίτητη χρήση του PAPI.
- **<storage enabled="no">**
  <trace-prefix enabled="yes">
  TRACE
  </trace-prefix>
  <size enabled="no">
  5
  </size>
  <temporal-directory enabled="yes">
  /scratch
  </temporal-directory>
  <final-directory enabled="yes">
  /gpfs/scratch/bsc41/bsc41273
  </final-directory>
  <gather-mpits enabled="no"/>
  </storage>

  Οδηγίες σχετικές με τη δημιουργία και την αποθήκευση των ενδιάμεσων αρχείων που απαιτούνται για τη δημιουργία του αρχείου trace.
- **<buffer enabled="yes">**
  <size enabled="yes">
  500000
  </size>
  <circular enabled="no"/>
  </buffer>

  Ρυθμίσεις για τον τρόπο λειτουργίας του buffer.
- **<trace-control enabled="no">**
  <file enabled="no" frequency="5M">
  /gpfs/scratch/bsc41/bsc41273/control
  </file>
  <global-ops enabled="no"/>

<remote-control enabled="no">
<signal enabled="no" which="USR1"/>
</remote-control>
</trace-control>
Ρυθμίσεις σχετικές με το τελικό αρχείο trace.

- **<others enabled="no">**
<minimum-time enabled="no">
10M
</minimum-time>
</others>
Ρυθμίσεις που δεν υπάγονται σε καμία από τις υπόλοιπες κατηγορίες. Προς στιγμήν η μόνη επιλογή που υποστηρίζεται είναι η οριοθέτηση ελάχιστου χρόνου υλοποίησης.
- **<bursts enabled="no">**
<threshold enabled="yes">
500u
</threshold>
<mpi-statistics enabled="yes"/>
</bursts>
Συλλογή πληροφοριών μόνο για «εκρήξεις» υπολογισμών. Υπολογισμοί που διαρκούν λιγότερο από το ορισμένο κατώφλι αγνοούνται. Επίσης συλλέγονται στατιστικές πληροφορίες για MPI οι οποίες μπορούν να περαστούν στο αρχείο trace.
- **<cell enabled="no">**
<spu-file-size enabled="yes">
5
</spu-file-size>
<spu-buffer-size enabled="yes">
64
</spu-buffer-size>
<spu-dma-channel enabled="no">
2
</spu-dma-channel>
</cell>
Συλλογή πληροφοριών όταν χρησιμοποιείται η αρχιτεκτονική Cell. Περισσότερες λεπτομέρειες στο user-guide.pdf του Extrae.
- **<sampling enabled="no" type="default" period="50m" variability="10m"/>**
Ρυθμίσεις σχετικά με τη δειγματοληψία των μετρήσεων από τους μετρητές του συστήματος.
- **<dynamic-memory enabled="no"/>**
Ρυθμίσεις για χρήση δυναμικών κλήσεων στη μνήμη (malloc, free, realloc).
- **<input-output enabled="no"/>**
Ρύθμιση για I/O calls.
- **<merge enabled="yes" synchronization="default" tree-fan-out="16" max-memory="512" joint-states="yes" keep-mpits="yes" sort-addresses="yes" overwrite="yes" binary="EXE">**
$TRACENAME$
</merge>
Ρυθμίσεις για το τελικό παραγόμενο αρχείο.

</trace>

Παράμετροι εκτέλεσης

Για την εκτέλεση του Extrae είναι απαραίτητο ένα αρχείο script από το οποίο προσδιορίζεται η μέθοδος που θα χρησιμοποιηθεί. Για τη χρήση του LD_PRELOAD το script αρχείο πρέπει να έχει το παρακάτω περιεχόμενο:

```
#!/bin/bash
export EXTRAE_HOME=<path of extrae installation>
export EXTRAE_CONFIG_FILE=<path of extrae.xml>
source ${EXTRAE_HOME}/etc/extrae.sh
export EXE=$2
export TRACENAME=${EXE}.prv
${EXTRAE_HOME}/bin/extrae $*
```

Για τη χρήση του **Dyninst** το script αρχείο πρέπει να έχει το παρακάτω περιεχόμενο:

```
#!/bin/sh
export EXTRAE_HOME=<path of extrae installation>
export EXTRAE_CONFIG_FILE=<path of extrae.xml>
export LD_PRELOAD=${EXTRAE_HOME}/lib/<ονομα βιβλιοθήκης>
export EXE=$2
export TRACENAME=${EXE}.prv
```

Ανάλογα το πρόγραμμα του οποίου γίνεται επεξεργασία, το <ονομα βιβλιοθήκης> πρέπει να αντικατασταθεί με `libmpitrace.so` αν το πρόγραμμα περιέχει μόνο MPI, `libomptrace.so` αν περιέχει μόνο openMP ή `libompitrace.so` αν περιέχει MPI και openMP. Αντίστοιχα, για λόγους ευχρηστίας, τα αρχεία μπορούν να ονομαστούν `mpitrace.sh`, `omptrace.sh` και `ompitrace.sh`.

ΠΑΡΑΡΤΗΜΑ ΙΙ - Εγκατάσταση και παραμετροποίηση Paraver

Εγκατάσταση

Λήψη από http://www.bsc.es/computer-sciences/performance-tools/downloads
υπάρχουν αντίστοιχες εκδόσεις για Linux και για Windows. Για την εγκατάσταση σε Linux:

`cd` <path of downloaded file> (π.χ. `cd /home/john/Downloads`)

`tar -xvzf wxparaver-4.5.4-linux-x86_64.tar` (ή `tar -xvzf wxparaver-4.5.4-linux-x86_32.tar`)

Δεν χρειάζεται περεταίρω εγκατάσταση. Για ευκολία στη χρήση προτείνεται μετονομασία του φάκελου, που προκύπτει μετά την αποσυμπίεση, σε paraver και μεταφορά του φακέλου εκεί που έχει γίνει εγκατάσταση και των υπολοίπων προγραμμάτων.

Παράμετροι λειτουργίας

Πριν την εκτέλεση του Paraver πρέπει να τεθούν δύο μεταβλητές περιβάλλοντος (environment variables). Οι εντολές είναι:

```
setenv LC_ALL "en_US.UTF8"
set path=(<path of paraver>/bin $path)
```

Για να εκτελεστεί:

```
./wxparaver
```

Σε περίπτωση που κατά τη διάρκεια εκτέλεσης του Paraver στο τερματικό εμφανίζονται μηνύματα όμοια με:

(wxparaver.bin:2298): Gtk-Warning **:Attempting to store changes into '/root/.local/share/recently-used.xbel', but failed: Failed to create file '/root/.local/share/recently-used.xbel.TBGEHY: No such file or directory

τότε από το τερματικό εκτελούμε τη παρακάτω εντολή για να δημιουργήσουμε τους απαραίτητους φακέλους που χρειάζεται το Paraver:

```
sudo mkdir -p /root/.local/share
```

Για να επαληθεύσουμε ότι έχει δημιουργηθεί ο φάκελος εκτελούμε την:

```
sudo ls /root/.local/
```

Το αποτέλεσμα πρέπει να είναι `share`

ΠΑΡΑΡΤΗΜΑ III - PAPI

Το PAPI (Performance Application Programming Interface) προσφέρει πρόσβαση σε μετρητές απόδοσης της μηχανής και του λειτουργικού συστήματος που υπάρχουν στους περισσότερους σύγχρονους επεξεργαστές. Παρέχει πληροφορίες, σε πραγματικό χρόνο, για τη σχέση μεταξύ της απόδοσης των προγραμμάτων και των γεγονότων του επεξεργαστή. Πάνω από 100 γεγονότα μπορούν αν καταμετρηθούν μέσω χρήσης μίας απλής, υψηλού επιπέδου, είτε μίας πιο πολύπλοκής, χαμηλού επιπέδου διεπαφής μέσω C ή Fortran.

Στο παρακάτω πίνακα φαίνονται τα σημαντικότερα utilities του. Βρίσκονται στο φάκελο `bin`, στο φάκελο εγκατάστασης του PAPI.

PAPI Utility Name	Description
papi_avail	provides availability and detail information for PAPI preset events
papi_clockres	provides availability and detail information for PAPI preset events
papi_cost	provides availability and detail information for PAPI preset events
papi_command_line	executes PAPI preset or native events from the command line
papi_decode	decodes PAPI preset events into a csv format suitable for PAPI_encode_events
papi_event_chooser	given a list of named events, lists other events that can be counted with them
papi_mem_info	provides information on the memory architecture of the current processor
papi_native_avail	provides detailed information for PAPI native events

Το papi_avail παρέχει τις σημαντικότερες πληροφορίες, αρχικά σχετικά με τον επεξεργαστή και στη συνέχεια για τους μετρητές τους οποίους αυτός υποστηρίζει μέσω μίας λίστας, όπου αναφέρεται αν τα δεδομένα μετρώνται σε πραγματικό χρόνο (Avail) ή αν προκύπτουν αργότερα από συνδυασμό άλλων μετρήσεων (Deriv). Τα ονόματα των μετρητών, όπως τα παρέχει το papi_avail, μπορούν να χρησιμοποιηθούν στο τμήμα **counters enabled** του extrae.xml (σελίδα 57) για τη παραμετροποίηση του Extrae, σχετικά με το ποιές μετρήσεις και από ποιούς μετρητές θα συμπεριληφθούν στο αρχείο trace. Υπάρχει η δυνατότητα δημιουργίας ομάδων μετρητών, οι οποίες καθορίζονται στο τμήμα **set enabled**. Η παράμετρος **domain** μπορεί να πάρει τις τιμές **kernel**, **user** η **all**, οπότε και συλλέγονται πληροφορίες όταν το πρόγραμμα εκτελείται σε kernel mode, user mode ή σε κάθε περίπτωση αντίστοιχα. Με τη παράμετρο **starting-set-distribution** διανέμονται τα set μετρητών στις διεργασίες και μπορεί να πάρει τις τιμές:

- **N**, όπου **N** είναι ο αριθμός του set με τη σειρά που έχουν καταχωρηθεί. Όλες οι διεργασίες χρησιμοποιούν το set αυτό
- **block**, όπου οι διεργασίες χωρίζονται σε τόσα block όσα και τα set των μετρητών και σε κάθε block αναθέτεται ένα set
- **cyclic**, όπου τα set αναθέτονται κυκλικά στις διεργασίες
- **random**, όπου η ανάθεση των set στις διεργασίες γίνεται τυχαία

Υπάρχει δυνατότητα αλλαγής του set μετρητών από το οποίο γίνονται μετρήσεις με δύο τρόπους. Ο πρώτος τρόπος είναι με τη χρήση της παραμέτρου **changeat-globalops="N"**, όπου N είναι ο αριθμός collective MPI εντολών και ο δεύτερος με τη χρήση της **changeat-time="NX"**, όπου N είναι η τιμή και X μπορεί να είναι s για δευτερόλεπτα, m για millisecond, M για λεπτά κ.τ.λ. Αν δεν υπάρχει κάποια από τις δύο αυτές παραμέτρους, τότε γίνεται χρήση μόνο ενός set μετρητών που πρέπει να έχει διευκρινιστεί με τη παράμετρο starting-set-distribution. και τη χρήση αριθμού για τη τιμή.

Το παρακάτω αποτελεί το αποτέλεσμα από την εκτέλεση του **papi_avail** στο εργαστήριο Linux του τμήματος Πληροφορικής και Τηλεπικοινωνιών του ΕΚΠΑ:

```
Available events and hardware information.
--------------------------------------------------------------------------------
PAPI Version             : 5.3.0.0
Vendor string and code   : GenuineIntel (1)
Model string and code    : Intel(R) Pentium(R) D CPU 2.80GHz (4)
CPU Revision             : 7.000000
CPUID Info               : Family: 15  Model: 4  Stepping: 7
CPU Max Megahertz        : 2792
CPU Min Megahertz        : 2792
Hdw Threads per core     : 1
Cores per Socket         : 2
Sockets                  : 1
CPUs per Node            : 2
Total CPUs               : 2
Running in a VM          : no
Number Hardware Counters : 18
Max Multiplex Counters   : 64
--------------------------------------------------------------------------------

    Name        Code    Avail Deriv Description (Note)
PAPI_L1_DCM  0x80000000  No    No   Level 1 data cache misses
PAPI_L1_ICM  0x80000001  Yes   No   Level 1 instruction cache misses
PAPI_L2_DCM  0x80000002  No    No   Level 2 data cache misses
PAPI_L2_ICM  0x80000003  No    No   Level 2 instruction cache misses
PAPI_L3_DCM  0x80000004  No    No   Level 3 data cache misses
PAPI_L3_ICM  0x80000005  No    No   Level 3 instruction cache misses
PAPI_L1_TCM  0x80000006  No    No   Level 1 cache misses
PAPI_L2_TCM  0x80000007  Yes   No   Level 2 cache misses
PAPI_L3_TCM  0x80000008  No    No   Level 3 cache misses
PAPI_CA_SNP  0x80000009  No    No   Requests for a snoop
PAPI_CA_SHR  0x8000000a  No    No   Requests for exclusive access to shared cache line
PAPI_CA_CLN  0x8000000b  No    No   Requests for exclusive access to clean cache line
PAPI_CA_INV  0x8000000c  No    No   Requests for cache line invalidation
PAPI_CA_ITV  0x8000000d  No    No   Requests for cache line intervention
PAPI_L3_LDM  0x8000000e  No    No   Level 3 load misses
PAPI_L3_STM  0x8000000f  No    No   Level 3 store misses
PAPI_BRU_IDL 0x80000010  No    No   Cycles branch units are idle
PAPI_FXU_IDL 0x80000011  No    No   Cycles integer units are idle
PAPI_FPU_IDL 0x80000012  No    No   Cycles floating point units are idle
PAPI_LSU_IDL 0x80000013  No    No   Cycles load/store units are idle
PAPI_TLB_DM  0x80000014  Yes   No   Data translation lookaside buffer misses
PAPI_TLB_IM  0x80000015  Yes   No   Instruction translation lookaside buffer misses
PAPI_TLB_TL  0x80000016  Yes   No   Total translation lookaside buffer misses
PAPI_L1_LDM  0x80000017  Yes   No   Level 1 load misses
PAPI_L1_STM  0x80000018  No    No   Level 1 store misses
PAPI_L2_LDM  0x80000019  Yes   No   Level 2 load misses
PAPI_L2_STM  0x8000001a  No    No   Level 2 store misses
PAPI_BTAC_M  0x8000001b  No    No   Branch target address cache misses
PAPI_PRF_DM  0x8000001c  No    No   Data prefetch cache misses
PAPI_L3_DCH  0x8000001d  No    No   Level 3 data cache hits
PAPI_TLB_SD  0x8000001e  No    No   Translation lookaside buffer shootdowns
PAPI_CSR_FAL 0x8000001f  No    No   Failed store conditional instructions
PAPI_CSR_SUC 0x80000020  No    No   Successful store conditional instructions
PAPI_CSR_TOT 0x80000021  No    No   Total store conditional instructions
PAPI_MEM_SCY 0x80000022  No    No   Cycles Stalled Waiting for memory accesses
PAPI_MEM_RCY 0x80000023  No    No   Cycles Stalled Waiting for memory Reads
PAPI_MEM_WCY 0x80000024  No    No   Cycles Stalled Waiting for memory writes
PAPI_STL_ICY 0x80000025  No    No   Cycles with no instruction issue
PAPI_FUL_ICY 0x80000026  No    No   Cycles with maximum instruction issue
PAPI_STL_CCY 0x80000027  No    No   Cycles with no instructions completed
PAPI_FUL_CCY 0x80000028  No    No   Cycles with maximum instructions completed
PAPI_HW_INT  0x80000029  No    No   Hardware interrupts
PAPI_BR_UCN  0x8000002a  No    No   Unconditional branch instructions
PAPI_BR_CN   0x8000002b  No    No   Conditional branch instructions
PAPI_BR_TKN  0x8000002c  Yes   No   Conditional branch instructions taken
PAPI_BR_NTK  0x8000002d  Yes   No   Conditional branch instructions not taken
PAPI_BR_MSP  0x8000002e  Yes   No   Conditional branch instructions mispredicted
```

```
PAPI_BR_PRC  0x8000002f  Yes   No   Conditional branch instructions correctly
predicted
PAPI_FMA_INS 0x80000030  No    No   FMA instructions completed
PAPI_TOT_IIS 0x80000031  Yes   No   Instructions issued (Only on model 2 and above)
PAPI_TOT_INS 0x80000032  Yes   No   Instructions completed
PAPI_INT_INS 0x80000033  No    No   Integer instructions
PAPI_FP_INS  0x80000034  Yes   No   Floating point instructions (PAPI_FP_INS counts
only retired x87 uops tagged with 0. If you add other native events tagged with 0,
their counts will be included in PAPI_FP_INS)
PAPI_LD_INS  0x80000035  Yes   No   Load instructions
PAPI_SR_INS  0x80000036  Yes   No   Store instructions
PAPI_BR_INS  0x80000037  Yes   No   Branch instructions
PAPI_VEC_INS 0x80000038  No    No   Vector/SIMD instructions (could include integer)
PAPI_RES_STL 0x80000039  Yes   No   Cycles stalled on any resource
PAPI_FP_STAL 0x8000003a  No    No   Cycles the FP unit(s) are stalled
PAPI_TOT_CYC 0x8000003b  Yes   No   Total cycles
PAPI_LST_INS 0x8000003c  Yes   No   Load/store instructions completed
PAPI_SYC_INS 0x8000003d  No    No   Synchronization instructions completed
PAPI_L1_DCH  0x8000003e  No    No   Level 1 data cache hits
PAPI_L2_DCH  0x8000003f  No    No   Level 2 data cache hits
PAPI_L1_DCA  0x80000040  No    No   Level 1 data cache accesses
PAPI_L2_DCA  0x80000041  No    No   Level 2 data cache accesses
PAPI_L3_DCA  0x80000042  No    No   Level 3 data cache accesses
PAPI_L1_DCR  0x80000043  No    No   Level 1 data cache reads
PAPI_L2_DCR  0x80000044  No    No   Level 2 data cache reads
PAPI_L3_DCR  0x80000045  No    No   Level 3 data cache reads
PAPI_L1_DCW  0x80000046  No    No   Level 1 data cache writes
PAPI_L2_DCW  0x80000047  No    No   Level 2 data cache writes
PAPI_L3_DCW  0x80000048  No    No   Level 3 data cache writes
PAPI_L1_ICH  0x80000049  No    No   Level 1 instruction cache hits
PAPI_L2_ICH  0x8000004a  No    No   Level 2 instruction cache hits
PAPI_L3_ICH  0x8000004b  No    No   Level 3 instruction cache hits
PAPI_L1_ICA  0x8000004c  Yes   No   Level 1 instruction cache accesses
PAPI_L2_ICA  0x8000004d  No    No   Level 2 instruction cache accesses
PAPI_L3_ICA  0x8000004e  No    No   Level 3 instruction cache accesses
PAPI_L1_ICR  0x8000004f  No    No   Level 1 instruction cache reads
PAPI_L2_ICR  0x80000050  No    No   Level 2 instruction cache reads
PAPI_L3_ICR  0x80000051  No    No   Level 3 instruction cache reads
PAPI_L1_ICW  0x80000052  No    No   Level 1 instruction cache writes
PAPI_L2_ICW  0x80000053  No    No   Level 2 instruction cache writes
PAPI_L3_ICW  0x80000054  No    No   Level 3 instruction cache writes
PAPI_L1_TCH  0x80000055  No    No   Level 1 total cache hits
PAPI_L2_TCH  0x80000056  Yes   No   Level 2 total cache hits
PAPI_L3_TCH  0x80000057  No    No   Level 3 total cache hits
PAPI_L1_TCA  0x80000058  No    No   Level 1 total cache accesses
PAPI_L2_TCA  0x80000059  Yes   No   Level 2 total cache accesses
PAPI_L3_TCA  0x8000005a  No    No   Level 3 total cache accesses
PAPI_L1_TCR  0x8000005b  No    No   Level 1 total cache reads
PAPI_L2_TCR  0x8000005c  No    No   Level 2 total cache reads
PAPI_L3_TCR  0x8000005d  No    No   Level 3 total cache reads
PAPI_L1_TCW  0x8000005e  No    No   Level 1 total cache writes
PAPI_L2_TCW  0x8000005f  No    No   Level 2 total cache writes
PAPI_L3_TCW  0x80000060  No    No   Level 3 total cache writes
PAPI_FML_INS 0x80000061  No    No   Floating point multiply instructions
PAPI_FAD_INS 0x80000062  No    No   Floating point add instructions
PAPI_FDV_INS 0x80000063  No    No   Floating point divide instructions
PAPI_FSQ_INS 0x80000064  No    No   Floating point square root instructions
PAPI_FNV_INS 0x80000065  No    No   Floating point inverse instructions
PAPI_FP_OPS  0x80000066  No    No   Floating point operations
PAPI_SP_OPS  0x80000067  No    No   Floating point operations; optimized to count
scaled single precision vector operations
PAPI_DP_OPS  0x80000068  No    No   Floating point operations; optimized to count
scaled double precision vector operations
PAPI_VEC_SP  0x80000069  No    No   Single precision vector/SIMD instructions
PAPI_VEC_DP  0x8000006a  No    No   Double precision vector/SIMD instructions
PAPI_REF_CYC 0x8000006b  No    No   Reference clock cycles
-----------------------------------------------------------------------
Of 108 possible events, 23 are available, of which 0 are derived.

avail.c                                   PASSED
```

Το παρακάτω αποτελεί το αποτέλεσμα από την εκτέλεση του **papi_avail** στον επεξεργαστή που χρησιμοποιήθηκε για το κεφάλαιο 5:

```
Available events and hardware information.
--------------------------------------------------------------------------------
PAPI Version             : 5.3.0.0
Vendor string and code   : GenuineIntel (1)
Model string and code    : Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz (58)
CPU Revision             : 9.000000
CPUID Info               : Family: 6  Model: 58  Stepping: 9
CPU Max Megahertz        : 3201
CPU Min Megahertz        : 1600
Hdw Threads per core     : 1
Cores per Socket         : 4
Sockets                  : 1
NUMA Nodes               : 1
CPUs per Node            : 4
Total CPUs               : 4
Running in a VM          : no
Number Hardware Counters : 11
Max Multiplex Counters   : 64
--------------------------------------------------------------------------------

    Name        Code    Avail Deriv Description (Note)
PAPI_L1_DCM  0x80000000  Yes   No   Level 1 data cache misses
PAPI_L1_ICM  0x80000001  Yes   No   Level 1 instruction cache misses
PAPI_L2_DCM  0x80000002  Yes   Yes  Level 2 data cache misses
PAPI_L2_ICM  0x80000003  Yes   No   Level 2 instruction cache misses
PAPI_L3_DCM  0x80000004  No    No   Level 3 data cache misses
PAPI_L3_ICM  0x80000005  No    No   Level 3 instruction cache misses
PAPI_L1_TCM  0x80000006  Yes   Yes  Level 1 cache misses
PAPI_L2_TCM  0x80000007  Yes   No   Level 2 cache misses
PAPI_L3_TCM  0x80000008  Yes   No   Level 3 cache misses
PAPI_CA_SNP  0x80000009  No    No   Requests for a snoop
PAPI_CA_SHR  0x8000000a  No    No   Requests for exclusive access to shared cache line
PAPI_CA_CLN  0x8000000b  No    No   Requests for exclusive access to clean cache line
PAPI_CA_INV  0x8000000c  No    No   Requests for cache line invalidation
PAPI_CA_ITV  0x8000000d  No    No   Requests for cache line intervention
PAPI_L3_LDM  0x8000000e  No    No   Level 3 load misses
PAPI_L3_STM  0x8000000f  No    No   Level 3 store misses
PAPI_BRU_IDL 0x80000010  No    No   Cycles branch units are idle
PAPI_FXU_IDL 0x80000011  No    No   Cycles integer units are idle
PAPI_FPU_IDL 0x80000012  No    No   Cycles floating point units are idle
PAPI_LSU_IDL 0x80000013  No    No   Cycles load/store units are idle
PAPI_TLB_DM  0x80000014  Yes   Yes  Data translation lookaside buffer misses
PAPI_TLB_IM  0x80000015  Yes   No   Instruction translation lookaside buffer misses
PAPI_TLB_TL  0x80000016  No    No   Total translation lookaside buffer misses
PAPI_L1_LDM  0x80000017  Yes   No   Level 1 load misses
PAPI_L1_STM  0x80000018  Yes   No   Level 1 store misses
PAPI_L2_LDM  0x80000019  No    No   Level 2 load misses
PAPI_L2_STM  0x8000001a  Yes   No   Level 2 store misses
PAPI_BTAC_M  0x8000001b  No    No   Branch target address cache misses
PAPI_PRF_DM  0x8000001c  No    No   Data prefetch cache misses
PAPI_L3_DCH  0x8000001d  No    No   Level 3 data cache hits
PAPI_TLB_SD  0x8000001e  No    No   Translation lookaside buffer shootdowns
PAPI_CSR_FAL 0x8000001f  No    No   Failed store conditional instructions
PAPI_CSR_SUC 0x80000020  No    No   Successful store conditional instructions
PAPI_CSR_TOT 0x80000021  No    No   Total store conditional instructions
PAPI_MEM_SCY 0x80000022  No    No   Cycles Stalled Waiting for memory accesses
PAPI_MEM_RCY 0x80000023  No    No   Cycles Stalled Waiting for memory Reads
PAPI_MEM_WCY 0x80000024  No    No   Cycles Stalled Waiting for memory writes
PAPI_STL_ICY 0x80000025  Yes   No   Cycles with no instruction issue
PAPI_FUL_ICY 0x80000026  No    No   Cycles with maximum instruction issue
PAPI_STL_CCY 0x80000027  No    No   Cycles with no instructions completed
PAPI_FUL_CCY 0x80000028  No    No   Cycles with maximum instructions completed
PAPI_HW_INT  0x80000029  No    No   Hardware interrupts
PAPI_BR_UCN  0x8000002a  Yes   Yes  Unconditional branch instructions
PAPI_BR_CN   0x8000002b  Yes   No   Conditional branch instructions
PAPI_BR_TKN  0x8000002c  Yes   Yes  Conditional branch instructions taken
PAPI_BR_NTK  0x8000002d  Yes   No   Conditional branch instructions not taken
PAPI_BR_MSP  0x8000002e  Yes   No   Conditional branch instructions mispredicted
```

```
PAPI_BR_PRC  0x8000002f  Yes   Yes  Conditional branch instructions correctly
predicted
PAPI_FMA_INS 0x80000030  No    No   FMA instructions completed
PAPI_TOT_IIS 0x80000031  No    No   Instructions issued
PAPI_TOT_INS 0x80000032  Yes   No   Instructions completed
PAPI_INT_INS 0x80000033  No    No   Integer instructions
PAPI_FP_INS  0x80000034  Yes   Yes  Floating point instructions
PAPI_LD_INS  0x80000035  Yes   No   Load instructions
PAPI_SR_INS  0x80000036  Yes   No   Store instructions
PAPI_BR_INS  0x80000037  Yes   No   Branch instructions
PAPI_VEC_INS 0x80000038  No    No   Vector/SIMD instructions (could include integer)
PAPI_RES_STL 0x80000039  No    No   Cycles stalled on any resource
PAPI_FP_STAL 0x8000003a  No    No   Cycles the FP unit(s) are stalled
PAPI_TOT_CYC 0x8000003b  Yes   No   Total cycles
PAPI_LST_INS 0x8000003c  No    No   Load/store instructions completed
PAPI_SYC_INS 0x8000003d  No    No   Synchronization instructions completed
PAPI_L1_DCH  0x8000003e  No    No   Level 1 data cache hits
PAPI_L2_DCH  0x8000003f  Yes   Yes  Level 2 data cache hits
PAPI_L1_DCA  0x80000040  No    No   Level 1 data cache accesses
PAPI_L2_DCA  0x80000041  Yes   No   Level 2 data cache accesses
PAPI_L3_DCA  0x80000042  Yes   Yes  Level 3 data cache accesses
PAPI_L1_DCR  0x80000043  No    No   Level 1 data cache reads
PAPI_L2_DCR  0x80000044  Yes   No   Level 2 data cache reads
PAPI_L3_DCR  0x80000045  Yes   No   Level 3 data cache reads
PAPI_L1_DCW  0x80000046  No    No   Level 1 data cache writes
PAPI_L2_DCW  0x80000047  Yes   No   Level 2 data cache writes
PAPI_L3_DCW  0x80000048  Yes   No   Level 3 data cache writes
PAPI_L1_ICH  0x80000049  No    No   Level 1 instruction cache hits
PAPI_L2_ICH  0x8000004a  Yes   No   Level 2 instruction cache hits
PAPI_L3_ICH  0x8000004b  No    No   Level 3 instruction cache hits
PAPI_L1_ICA  0x8000004c  No    No   Level 1 instruction cache accesses
PAPI_L2_ICA  0x8000004d  Yes   No   Level 2 instruction cache accesses
PAPI_L3_ICA  0x8000004e  Yes   No   Level 3 instruction cache accesses
PAPI_L1_ICR  0x8000004f  No    No   Level 1 instruction cache reads
PAPI_L2_ICR  0x80000050  Yes   No   Level 2 instruction cache reads
PAPI_L3_ICR  0x80000051  Yes   No   Level 3 instruction cache reads
PAPI_L1_ICW  0x80000052  No    No   Level 1 instruction cache writes
PAPI_L2_ICW  0x80000053  No    No   Level 2 instruction cache writes
PAPI_L3_ICW  0x80000054  No    No   Level 3 instruction cache writes
PAPI_L1_TCH  0x80000055  No    No   Level 1 total cache hits
PAPI_L2_TCH  0x80000056  No    No   Level 2 total cache hits
PAPI_L3_TCH  0x80000057  No    No   Level 3 total cache hits
PAPI_L1_TCA  0x80000058  No    No   Level 1 total cache accesses
PAPI_L2_TCA  0x80000059  Yes   Yes  Level 2 total cache accesses
PAPI_L3_TCA  0x8000005a  Yes   No   Level 3 total cache accesses
PAPI_L1_TCR  0x8000005b  No    No   Level 1 total cache reads
PAPI_L2_TCR  0x8000005c  Yes   Yes  Level 2 total cache reads
PAPI_L3_TCR  0x8000005d  Yes   Yes  Level 3 total cache reads
PAPI_L1_TCW  0x8000005e  No    No   Level 1 total cache writes
PAPI_L2_TCW  0x8000005f  Yes   No   Level 2 total cache writes
PAPI_L3_TCW  0x80000060  Yes   No   Level 3 total cache writes
PAPI_FML_INS 0x80000061  No    No   Floating point multiply instructions
PAPI_FAD_INS 0x80000062  No    No   Floating point add instructions
PAPI_FDV_INS 0x80000063  Yes   No   Floating point divide instructions
PAPI_FSQ_INS 0x80000064  No    No   Floating point square root instructions
PAPI_FNV_INS 0x80000065  No    No   Floating point inverse instructions
PAPI_FP_OPS  0x80000066  Yes   Yes  Floating point operations
PAPI_SP_OPS  0x80000067  Yes   Yes  Floating point operations; optimized to count
scaled single precision vector operations
PAPI_DP_OPS  0x80000068  Yes   Yes  Floating point operations; optimized to count
scaled double precision vector operations
PAPI_VEC_SP  0x80000069  Yes   Yes  Single precision vector/SIMD instructions
PAPI_VEC_DP  0x8000006a  Yes   Yes  Double precision vector/SIMD instructions
PAPI_REF_CYC 0x8000006b  Yes   No   Reference clock cycles
-----------------------------------------------------------------------
Of 108 possible events, 50 are available, of which 17 are derived.

avail.c                                    PASSED
```

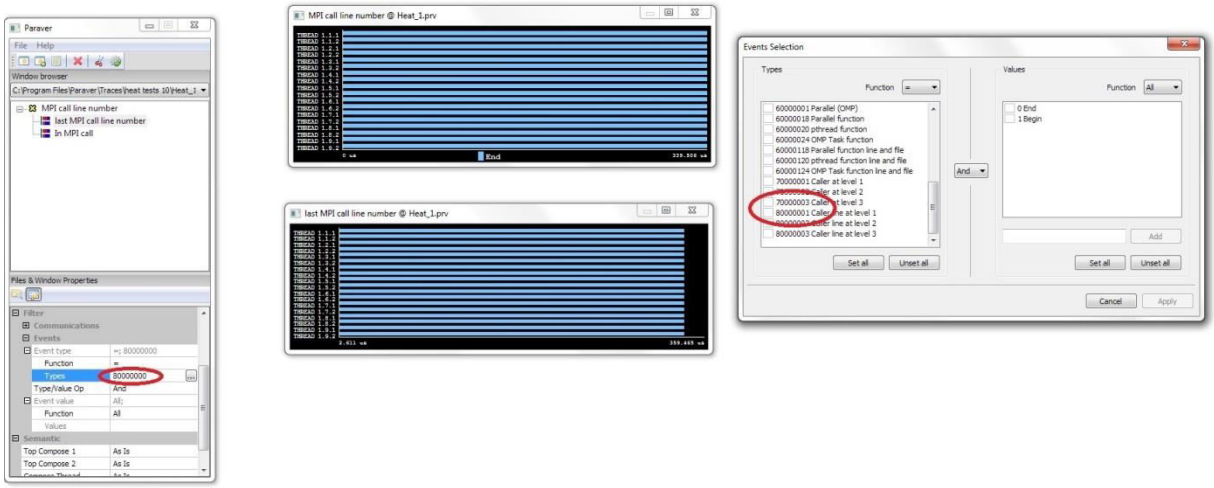
ΠΑΡΑΡΤΗΜΑ IV

Επικοινωνία με το Barcelona Supercomputing Center

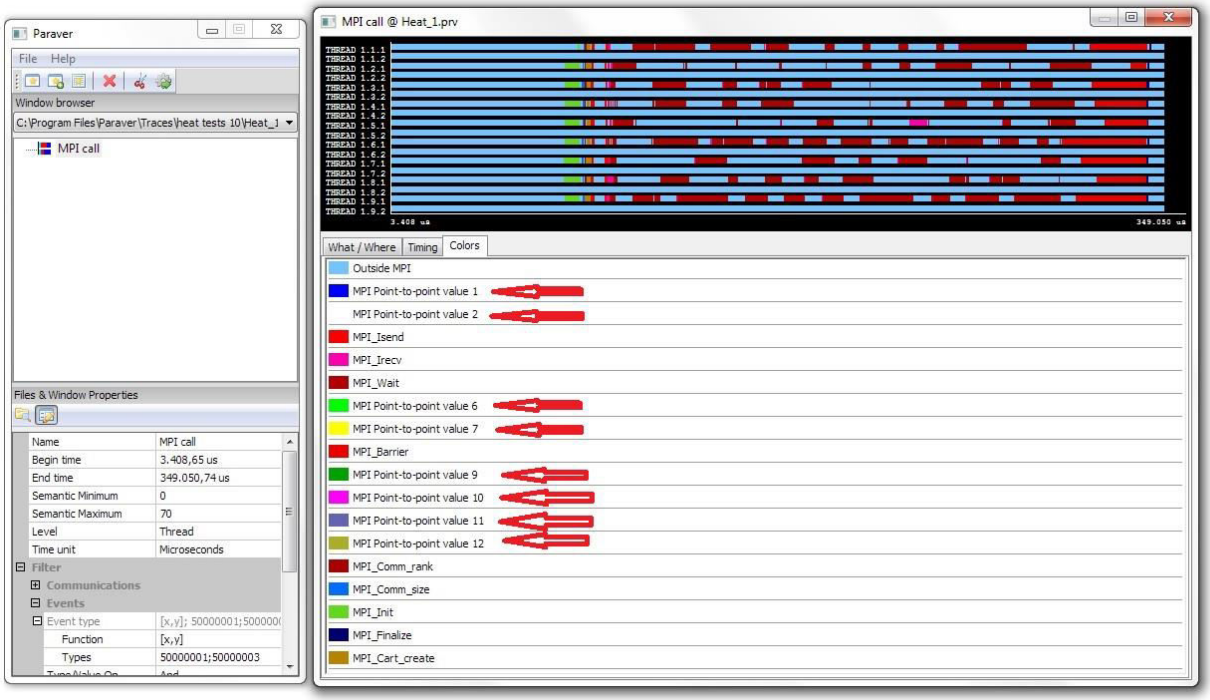
Μετά από ανεπιτυχή προσπάθεια χρήσης του configuration file **From_where_mpi_calls**, το οποίο παρουσιάζει τη γραμμή κώδικα στην οποία γίνεται η κλήση κάθε εντολής, πραγματοποιήθηκε επικοινωνία με το Barcelona Supercomputing Center και την Judit Gimenez, η οποία εργάζεται ως Tools Group Manager. Έγινε, επίσης, ερώτηση σχετική με το τρόπο που παρουσιάζονται οι διαφορετικές καταστάσεις ενός γραφήματος βάση των χρωμάτων στο Info Panel. Παρακάτω αναλυτικά η συνομιλία μέσω e-mail όπου με μπλε χρώμα είναι το αρχικό mail και με μαύρο η απάντηση. Μετά το κείμενο φαίνονται και οι συνημμένες εικόνες:

Θέμα: Re: Help Needed with Paraver
Από: "Judit Gimenez" <judit@bsc.es>
Ημερομηνία: Δευ, Φεβρουάριος 16, 2015 12:57
Προς: "Ioannis Monios" <ymonios@phys.uoa.gr>

```
Hello Ioannis,

Find my answers below.

El 16/02/15 10:45, Ioannis Monios escribió:
> Dear Madam,
>
> My name is Ioannis Monios and I am a post graduate student at the the
> National and Kapodistrian University of Athens studying Electronic
> Automation. I am sending you this e-mail because as my dissertation
> project I work on Extrae and Paraver and I have some questions on Paraver
> and I would be grateful if you could help me with them.
> First, when i try to use the configuration file
> /cfgs/mpi/views/advanced/From_where_mpi_calls.cfg the "last MPI call line
> number" cfg does not display any values. I have checked the "Event"
> options through the filter and the event type number did not exist in the
> list from where to choose (I have attached the picture
> from_where_mpi_calls.jpg and the extrae.xml for Extrae that I am using, i
> hope they help). Is there a problem with the configurations or could it be
> that the hardware does not support this function?
As we can see that there are other events for the callers, I do not
think there is a problem on the Extrae installation but on the cfg. You
can use
$PARAVER_HOME/cfgs/General/link_to_source/by_call_stack/MPI_caller_line.cfg
that should work ok. We will fix this error for future distributions.

> Moreover, when using the /cfgs/mpi/MPI_call.cfg, is there a way to exclude
> from the colors tab of the info panel the lines that do not correspond to
> commands that are used by the program (I have attached MPI_call.jpg as an
> example)?
> Thank you for your time.
I am sorry, currently this is not possible but we will consider adding
this feature.

Thanks for your mail.
Best regards,
judit

> Yours faithfully,
> Ioannis Monios
>
```



Εικόνα 54: from_where_mpi_calls.jpg



Εικόνα 55: MPI_call.jpg

Το cfg που προτείνεται ως εναλλακτική δεν λειτουργεί όπως θα έπρεπε λόγω της μη υποστήριξης των σχετικών counters που απαιτεί το PAPI για τη συγκεκριμένη λειτουργία.

ΑΝΑΦΟΡΕΣ

[1] http://www.bsc.es/
[2] http://icl.cs.utk.edu/papi/
[3] An Introduction to Parallel Programming, Peter Pacheco
[4] Designing and Building Parallel Programs, Ian Foster
[5] Χρησιμοποιήθηκαν οι εργασίες με θέμα το πρόγραμμα προσομοίωσης μετάδοσης θερμότητας:
Heat Simulation - Αγρανιώτης Δημήτριος-Ευθυμιάτου Στεφανία
HEAT2D - Βαρυθυμιάδης Νικόλαος-Κάσσος Δημήτρης
Εργασία περιόδου Σεπτεμβρίου 2014 - Λεβογιάννης Μάριος