

Ετήσια Τεχνική Έκθεση

Έτος 2014



ΘΑΛΗΣ – Πολυτεχνείο Κρήτης

Πλατφόρμα προηγμένων μαθηματικών μεθόδων
και λογισμικού για την επίλυση προβλημάτων
πολλαπλών πεδίων (multi physics, multidomain)
σε σύγχρονες υπολογιστικές αρχιτεκτονικές:
Εφαρμογή σε προβλήματα Περιβαλλοντικής
Μηχανικής και Ιατρικής (MATENVMED- MIS 379416)

Δράση 4.1

Συγκερασμός αριθμητικών μεθόδων και
λογισμικού

Περιεχόμενα

1	Σκοπός	3
2	Μεθοδολογία	3
2.1	Επέκταση του ΠΕΠ για την Υποστήριξη της Κλάσης Προβλημάτων Ενδιαφέροντος του Έργου	3
2.1.1	Μέθοδος Schwarz για Προβλήματα με Επικαλυπτώμενα Υποχωρία	4
2.1.2	Υβριδική Αιτιοκρατική/Στοχαστική Μέθοδος για Προβλήματα MDMP	5
2.1.3	Μέθοδος Geometric Contraction (GEO)	6
2.1.4	Μέθοδος ROB	7
2.2	Αξιοποίηση Σύγχρονων Μέσων Αποθήκευσης	8
3	Πειραματικά Αποτελέσματα	8
3.1	Επέκταση του ΠΕΠ για την Υποστήριξη της Κλάσης Προβλημάτων Ενδιαφέροντος του Έργου	8
3.1.1	Μέθοδος Schwarz για Προβλήματα με Επικαλυπτώμενα Υποχωρία	8
3.1.2	Υβριδική Αιτιοκρατική/Στοχαστική Μέθοδος για Προβλήματα MDMP	11
3.1.3	Μέθοδος Geometric Contraction (GEO)	12
3.2	Αξιοποίηση Σύγχρονων Μέσων Αποθήκευσης	16
4	Παραδοτέα	18
5	Συνεργασίες	19
6	Μελλοντικές Δράσεις	19

1 Σκοπός

Η φυσική πραγματικών και μεγάλων προβλημάτων επιβάλλει αυτά να αντιμετωπιστούν με πολλά και διαφορετικά μοντέλα, ώστε το αρχικό πρόβλημα να προσομοιωθεί και να λυθεί με το βέλτιστο δυνατό τρόπο. Η μεθοδολογία χαλάρωσης της διεπαφής αντιμετωπίζει σύνθετα προβλήματα και ορίζεται σε ένα αρκετά υψηλό επίπεδο με αποτέλεσμα να υποστηρίζει την ετερογένεια σε όλα τα επίπεδα υλοποίησης του προβλήματος. Τα Περιβάλλοντα Επίλυσης Προβλημάτων (ΠΕΠ) σχεδιάζονται και υλοποιούνται εδώ και αρκετά χρόνια και υποστηρίζουν υψηλό επίπεδο αφαίρεσης και ετερογένειας τόσο στο επίπεδο φυσικής και μαθηματικών όσο και στο επίπεδο υλοποίησης, προγραμματισμού, μηχανών εκτέλεσης, υπολογισμού και οπτικοποίησης της λύσης. Οφείλουν να εκμεταλλεύονται οποιαδήποτε επιπλέον δυνατότητα παρέχεται από το υλικό, όπως για παράδειγμα γρήγορα μέσα μόνιμης αποθήκευσης (flash storage).

Η παρούσα δράση σχεδιάζει με σαφήνεια μια αρχιτεκτονική αναφοράς, η οποία φιλοδοξεί να θέσει τις βάσεις για μια ενοποιημένη προσέγγιση αντιμετώπισης των σύνθετων προβλημάτων που απασχολούν το έργο MATENVMED. Οι στόχοι που απασχόλησαν τις ερευνητικές ομάδες κατά το 2014 ήταν οι ακόλουθοι:

- Επέκταση του ΠΕΠ για την υποστήριξη της κλάσης προβλημάτων ενδιαφέροντος του έργου.
- Αξιοποίηση σύγχρονου υλικού αποθήκευσης (flash storage).

Το υπόλοιπο της παρούσης Τεχνικής Έκθεσης είναι οργανωμένο ως εξής: Στην παράγραφο 2 παρουσιάζουμε τα βασικά στοιχεία της μεθοδολογίας που ακολουθήσαμε και στην παράγραφο 3 τα σημαντικότερα πειραματικά αποτελέσματα. Στην παράγραφο 4 αναφερόμαστε στα παραδοτέα που παρήχθησαν στα πλαίσια της δράσης. Στην παράγραφο 5 περιγράφουμε τις συνεργασίες που αναπτύχθηκαν, ενώ η παράγραφος 6 ολοκληρώνει την έκθεση με τη συζήτηση πιθανών επεκτάσεων της υποδομής που παράχθηκε στα πλαίσια της δράσης.

2 Μεθοδολογία

2.1 Επέκταση του ΠΕΠ για την Υποστήριξη της Κλάσης Προβλημάτων Ενδιαφέροντος του Έργου

Ο βασικός στόχος των επεκτάσεών μας είναι η σχεδίαση και υλοποίηση ενός ανοιχτού, βελτιωμένου περιβάλλοντος μετα-υπολογισμού το οποίο θα υποστηρίζει προβλήματα πολλαπλών χωρίων / πολλαπλών τελεστών (Multi-Domain /



Multi-Physics – MDMP), χωρίς να απαιτούνται αλλαγές στα χαμηλά επίπεδα του FEniCS (δημιουργία συστημάτων, γραμμική επίλυση συστημάτων κλπ). Η πλατφόρμα μας αξιοποιεί και επεκτείνει το Python API του Dolphin, καθώς η σύνταξη της Python είναι πλησιέστερη στη UFL, ενώ ταυτόχρονα είναι καταλληλότερη και για ταχεία ανάπτυξη κώδικα. Η υποστήριξη για προβλήματα MDMP υλοποιείται πάνω από την προϋπάρχουσα λειτουργικότητα, είτε ως Python modules αξιοποιώντας υπάρχουσες δομές δεδομένων και κλάσεις, είτε ως εξωτερικές, δυναμικά διασυνδεόμενες βιβλιοθήκες C++, οι οποίες παρουσιάζονται ως Python modules με χρήση του SWIG [4].

Η πλατφόρμα στοχεύει στην υποστήριξη μεγάλου εύρους προβλημάτων με τον ίδιο γενικό σχεδιασμό. Πιο συγκεκριμένα, υποστηρίζονται περίπλοκα σχήματα για τα υποχωρία και τις διεπαφές μεταξύ τους. Επίσης υποστηρίζονται 2-διάστατες και 3-διάστατες γεωμετρίες.

Βασική προϋπόθεση για την υποστήριξη MDMP προβλημάτων είναι η δυνατότητα χρήσης διαφορετικών διαφορικών τελεστών, διαφορετικής διακριτοποίησης και ενδεχομένως και διαφορετικών επιλυτών σε κάθε υποχωρίο. Το FEniCS υποστηρίζει ήδη τον ορισμό ανεξάρτητων υποχωρίων. Η πλατφόρμα μας αξιοποιεί την υπάρχουσα υποδομή και χτίζει επάνω της.

2.1.1 Μέθοδος Schwarz για Προβλήματα με Επικαλυπτόμενα Υποχωρία

Υλοποιήθηκε η επαναληπτική μέθοδος additive Schwarz [5, chapter 2.1] για προσαρμογή των διεπαφών σε προβλήματα με μερικώς επικαλυπτόμενα χωρία και χρησιμοποιήθηκε στα πλαίσια του FEniCS για την υποστήριξη των αντίστοιχων MDMP προβλημάτων.

Η γεωμετρία, οι διεπαφές, η διακριτοποίηση, οι συνοριακές τιμές και οι εξισώσεις που ισχύουν σε κάθε υποχωρίο περιγράφονται με χρήση UFL και Dolphin σε ένα ξεχωριστό αρχείο για κάθε υποχωρίο. Αυτή η οργάνωση αντιμετωπίζει τα υποχωρία ως διακριτές μονάδες κώδικα, διευκολύνοντας την παράλληλη ή καταναεμημένη επίλυση των υποπροβλημάτων που αφορούν διαφορετικά υποχωρία.

Όλοι οι τύποι δεδομένων που χρησιμοποιούνται είναι αντικείμενα είτε της Python είτε του FEniCS. Δεν υπάρχουν εξαρτήσεις από εξωτερικό λογισμικό.

Κάθε αντικείμενο που αντιπροσωπεύει ένα υποχωρίο πρέπει να επαναορίσει (override) έναν αριθμό μεθόδων που καλούνται εμμέσως πριν από κάθε εκτέλεση του επιλυτή.

`init()` Η μέθοδος αυτή διατηρεί τον ορισμό UFL του υποχωρίου. Θέτει ως `attributes` του αντικειμένου που αντιστοιχεί στο υποχωρίο το χώρο συναρτήσεων του υποχωρίου καθώς και τη γραμμική και δι-γραμμική μορφή της μερικής διαφορικής εξίσωσης.

`neighbors()` Παρέχει πληροφορία στον επιλυτή σχετικά με τα άλλα υποχωρία που επικαλύπτονται με το τρέχον, έτσι ώστε ο επιλυτής να ανανεώνει αυτόματα τις τιμές στις διεπαφές μετά από κάθε επανάληψη.

`boundaries()` Παρέχει πληροφορία στον επιλυτή σχετικά με τα τυχόν εξωτερικά σύνορα του υποχωρίου.

Το αρχικό σημείο εισόδου στον επαναληπτικό επιλυτή είναι η μέθοδος `solve()`. Λαμβάνει ως παραμέτρους ένα αντικείμενο που περιγράφει την επιθυμητή διαμόρφωση του περιβάλλοντος επίλυσης (μέγιστες επαναλήψεις, επιθυμητή ακρίβεια κλπ) και μια λίστα Python με τα υποχωρία που έχει ορίσει ο χρήστης.

Μετά από κάθε επανάληψη, για κάθε ανεξάρτητη επίλυση υποχωρίου, το σύστημα υπολογίζει έναν `interpolant` ο οποίος προσεγγίζει την περιγραφή της λύσης στις διεπαφές με γειτονικά υποχωρία. Ο εκάστοτε `interpolant` αξιοποιείται ώστε τα γειτονικά `subdomains` να προσαρμόσουν τις εκτιμήσεις των λύσεων στις διεπαφές πριν την επόμενη επανάληψη. Οι `interpolants` αποτελούν και το μοναδικό τρόπο με τον οποίο ανταλλάσσεται (ελάχιστη) πληροφορία ανάμεσα στα κατά τα άλλα ανεξάρτητα υποπροβλήματα επίλυσης στα υποχωρία.

Τέλος, μετά από κάθε επανάληψη και για κάθε ανεξάρτητη επίλυση υποχωρίου, ο αλγόριθμος ελέγχει ένα σύνολο από κριτήρια τερματισμού (σύγκλιση ή/και μέγιστος αριθμός επαναλήψεων).

Στην παράγραφο 3.1.1 παρουσιάζεται η χρήση της υποδομής Schwarz στο FEniCS για την επαναληπτική επίλυση προβλήματος μοντέλου που αποτελείται από δύο υποχωρία τριών διαστάσεων, αλλά και πραγματικού προβλήματος περιβαλλοντικής μηχανικής σε 2 διαστάσεις.

2.1.2 Υβριδική Αιτιοκρατική/Στοχαστική Μέθοδος για Προβλήματα MDMP

Εισάγαμε επεκτάσεις στο FEniCS για την υποστήριξη υβριδικής αιτιοκρατικής/στοχαστικής μεθόδου επίλυσης προβλημάτων MDMP. Πιο συγκεκριμένα, ένα στοχαστικό βήμα [10] εκτελείται για την εκτίμηση των τιμών στις διεπαφές υποχωρίων, ενώ αξιοποιείται η υπάρχουσα υποστήριξη του FEniCS στο επίπεδο των μεθόδων παρεμβολής και επίλυσης. Αυτή η αρχιτεκτονική απομονώνει το στοχαστικό βήμα της εκτίμησης των τιμών στις διεπαφές από την επίλυση αυτή καθαυτή, και επιτρέπει την υλοποίηση του στοχαστικού βήματος σε οποιαδήποτε υπολογιστική συσκευή (συμπεριλαμβανομένων CPUs, GPUs ή ακόμα και FPGAs), έτσι ώστε να αξιοποιηθεί ο ευρείας κλίμακας παραλληλισμός που είναι διαθέσιμος στις μεθόδους Monte-Carlo. Η υλοποίησή μας περιλαμβάνει μια έκδοση PosixThreads για CPUs, καθώς και μια έκδοση OpenCL [8] για οποιαδήποτε συσκευή υποστηρίζει OpenCL (CPUs, GPUs και FPGAs).

Η λειτουργικότητα υλοποιείται σε C/C++ και προσφέρεται, με χρήση SWIG, στο Dolphin με τη μορφή μιας κλάσης Python MC, η οποία παρέχει τη μέθοδο

MC::montecarlo(). Η τελευταία δέχεται ως παράμετρο την περιγραφή της διεπαφής και επιστρέφει εκτιμήσεις για τις τιμές επάνω στη διεπαφή. Υποστηρίζονται προβλήματα έως και 3 διαστάσεων.

Πιο συγκεκριμένα, η μέθοδος montecarlo() δέχεται της ίδιες παραμέτρους με την κλάση DirichletBC του FEniCS και επιπλέον μια περιγραφή του αρχικού χωρίου και του υποχωρίου ενδιαφέροντος. Με χρήση των μεθόδων της κλάσης DirichletBC λαμβάνουμε τα σημεία στη διεπαφή και καλούμε τη νέα μέθοδο montecarlo() για αυτά. Η κλήση επιστρέφει τις εκτιμώμενες τιμές όλων των σημείων (κόμβων) στη διεπαφή, με τη μορφή ενός νέου αντικειμένου DirichletBC, το οποίο μπορεί να χρησιμοποιηθεί οπουδήποτε στον υπόλοιπο κώδικα.

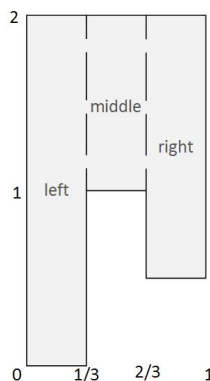
Στην παράγραφο 3.1.2 παρουσιάζεται η χρήση της υποδομής υποστήριξης υβριδικής αιτιοκρατικής/στοχαστικής μεθόδου στο FEniCS για την επίλυση υποπροβλήματος PDE με αρχική στοχαστική εκτίμηση των τιμών στη διεπαφή.

2.1.3 Μέθοδος Geometric Contraction (GEO)

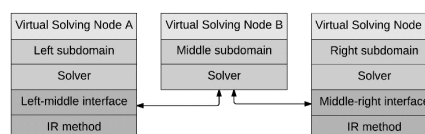
Ακολουθιακή Υλοποίηση της GEO Η μέθοδος geometric (GEO) contraction [9] υλοποιείται σαν ένα πρόγραμμα FEniCS γραμμένο στη γλώσσα UML. Αξιοποιείται η υποστήριξη του Dolfin για την εισαγωγή κλάσεων χρήσιμων για τη δημιουργία των υπο-πεδίων του προβλήματος και τη δημιουργία πλεγμάτων (τριγωνικά στοιχεία) σε αυτά τα υπο-πεδία. Στη συνέχεια δηλώνουμε και εφαρμόζουμε τις οριακές συνθήκες, καθώς επίσης και τις αρχικές μαντεψιές στις διεπαφές των υπο-πεδίων. Το πρόβλημα PDE πρέπει να εκφραστεί σαν ένα παραμετρικό (variational) πρόβλημα και στη συνέχεια να οριστεί στο πρόγραμμα. Μετά τον υπολογισμό της λύσης, πραγματοποιείται και ο υπολογισμός της παραγώγου. Η δημιουργία κατάλληλων συναρτήσεων για την λήψη των τιμών της λύσης και της παραγώγου στα σημεία των διεπαφών (όρια των υπο-προβλημάτων), ο υπολογισμός των νέων χαλαρωμένων τιμών και το πέρασμά τους πίσω στα υπο-προβλήματα σαν ανανεωμένες τιμές για τις διεπαφές ήταν οι κυριότερες προκλήσεις της υλοποίησης της GEO.

Παράλληλη Υλοποίηση της GEO Η μέθοδος GEO είναι εγγενώς παραλληλίστη. Στην παράλληλη υλοποίησή της κάθε κόμβος επιλύει ένα υπο-πεδίο. Οι υπολογισμένες τιμές της λύσης και της παραγώγου στα σημεία των διεπαφών από κάθε κόμβο (που επιλύει ένα υπο-πεδίο που έχει παραπάνω από ένα γειτονικά υπο-πεδία) στέλνονται σαν ένα μήνυμα RabbitMQ στους κόμβους που επιλύουν τα γειτονικά υπο-πεδία. Μια νέα επανάληψη ξεκινά μόλις οι κόμβοι που διαχειρίζονται τα γειτονικά υπο-πεδία τελειώσουν το βήμα της επικοινωνίας σχετικά με την υπολογιζόμενη λύση. Αυτή η αρχιτεκτονική, για ένα πρόβλημα μοντέλο με τρία υπο-πεδία, απεικονίζεται στο Σχήμα 1. Δεν εμπεριέχει ξεχωριστούς κόμβους για να διαχειρίζονται τις διεπαφές εξυπηρετεί στη μείωση του αριθμού

των μηνυμάτων που ανταλλάσσονται, σε μια προσπάθεια να ελαχιστοποιηθεί ο συνολικός χρόνος επικοινωνίας.



(α') Τοπολογία υποχωρίων



(β') Κατανεμημένη υλοποίηση GEO

Εικόνα 1: Αρχιτεκτονική κατανεμημένης επίλυσης για τη μέθοδο GEO σε πρόβλημα-μοντέλο με 3 υποχωρία

Στην παράγραφο 3.1.3 δίνεται ενδεικτικός κώδικας ακολουθιακής και παράλληλης υλοποίησης της GEO.

2.1.4 Μέθοδος ROB

Ακολουθιακή Υλοποίηση της ROB Η μέθοδος ROB [9] βασίζεται σε Robin συνθήκες στις διεπαφές ώστε να δώσει πληροφορία από τη διεπαφή του ενός υπο-πεδίου στο άλλο. Υλοποιείται με τρόπο παρόμοιο με αυτόν που περιγράφηκε για τη GEO. Η τοπική ΜΔΕ στα υποπεδία λύνεται χρησιμοποιώντας Robin συνθήκες στις διεπαφές, αντιστοιχίζοντας ένα συνδυασμό από Dirichlet και Neumann δεδομένα από τα γειτονικά υποπεδία

Παράλληλη Υλοποίηση της ROB Η επίλυση του κάθε υποπεδίου είναι ανεξάρτητη από των υπολοίπων, οπότε η μέθοδος ROB προσφέρεται για παραλληλοποίηση. Παρόμοια και με την παράλληλη υλοποίηση της GEO, οι υπολογισμένες τιμές της λύσης και της παραγώγου στα σημεία των διεπαφών από κάθε κόμβο (που επιλύει ένα υπο-πεδίο που έχει παραπάνω από ένα γειτονικά υπο-πεδία) στέλνονται σαν ένα μήνυμα RabbitMQ, έτσι ώστε η νέα επανάληψη να ξεκινήσει και στον άλλο κόμβο. Και πάλι το σχήμα δεν εμπεριέχει ξεχωριστούς κόμβους για να διαχειρίζονται τις διεπαφές και εξυπηρετεί στη μείωση του αριθμού των μηνυμάτων που ανταλλάσσονται, σε μια προσπάθεια να ελαχιστοποιηθεί ο συνολικός χρόνος επικοινωνίας.

2.2 Αξιοποίηση Σύγχρονων Μέσων Αποθήκευσης

Η αποδοτική επίλυση συστημάτων γραμμικών εξισώσεων με αραιούς πίνακες είναι θεμελιώδους σημασίας για τις υπολογιστικές επιστήμες, αφού τέτοιου είδους συστήματα απαντώνται κατά την επίλυση προβλημάτων του πραγματικού κόσμου (real world problems). Πιο συγκεκριμένα, στα προβλήματα πολλαπλών πεδίων (multidomain/multiphysics problems) αραιοί πίνακες προκύπτουν κατά την διακριτοποίηση μερικών διαφορικών εξισώσεων (ανεξάρτητα της μεθόδου διακριτοποίησης). Έτσι, τα λογισμικά για την προσομοίωση προβλημάτων πολλαπλών μοντέλων φυσικής (π.χ. COMSOL, ANSYS, Code_Aster, FEniCS) ενσωματώνουν και μεθόδους για την επίλυση αραιών συστημάτων.

Για την αντιμετώπιση των αυξημένων απαιτήσεων σε μνήμη στις άμεσες μεθόδους, έχουν προταθεί αλγόριθμοι που χρησιμοποιούν δευτερεύουσα μνήμη (αλγόριθμοι out-of-core). Οι περισσότερες βιβλιοθήκες λογισμικού για την επίλυση αραιών συστημάτων με άμεσες μεθόδους (π.χ. INTEL MKL PARDISO, MUMPS, PASTIX, TAUCS) περιλαμβάνουν και τέτοιους αλγορίθμους.

Τα τελευταία χρόνια οι μνήμες flash χρησιμοποιούνται ως αποθηκευτικό μέσο σε ενσωματωμένα συστήματα, κινητά τηλέφωνα, φορητούς υπολογιστές και εξυπηρετητές, έχοντας σαφή πλεονεκτήματα έναντι των παραδοσιακών μαγνητικών δίσκων. Οι εξαιρετικές επιδόσεις τους μας έδωσαν κίνητρο να ερευνήσουμε την δυνατότητα αξιοποίησής τους από αλγορίθμους που χρησιμοποιούν εξωτερική μνήμη για την επίλυση αραιών γραμμικών συστημάτων.

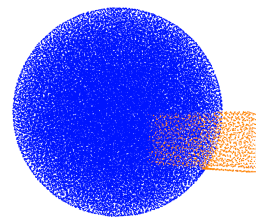
Μελετήσαμε την επίλυση αραιών συστημάτων με τα πακέτα: MUMPS [1] [2], την έκδοση του PARDISO [6] [7] που περιλαμβάνεται στην INTEL MKL και το WSMP [3] της IBM. Το PARDISO και το MUMPS υποστηρίζουν την επίλυση συμμετρικών και μη συμμετρικών συστημάτων, ενώ το WSMP υποστηρίζει μόνο συμμετρικά συστήματα. Πραγματοποιήσαμε πειράματα σε μαγνητικό σκληρό δίσκο και σε οδηγό στερεάς κατάστασης (flash SSD) και τα αποτελέσματα παρουσιάζονται στην Παράγραφο 3.2.

3 Πειραματικά Αποτελέσματα

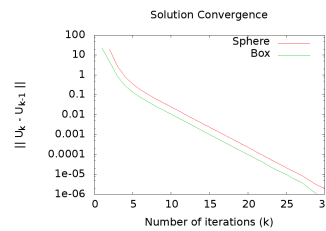
3.1 Επέκταση του ΠΕΠ για την Υποστήριξη της Κλάσης Προβλημάτων Ενδιαφέροντος του Έργου

3.1.1 Μέθοδος Schwarz για Προβλήματα με Επικαλυπτώμενα Υποχωρία

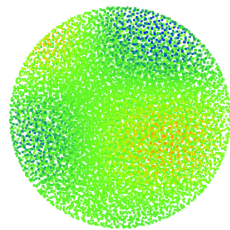
Χρήση σε 3D πρόβλημα-μοντέλο Ας θεωρήσουμε ένα μοντέλο 3-διάστατο πρόβλημα με 2 υποχωρία: μία σφαίρα και ένα παραλληλεπίπεδο, τα οποία επικαλύπτονται μερικώς όπως φαίνεται στην Εικόνα 2(α'). Οι Εικόνες 2(γ') και 2(δ') απεικονίζουν τη λύση για τα δύο υποχωρία με χρήση του επαναληπτικού επιλυτή



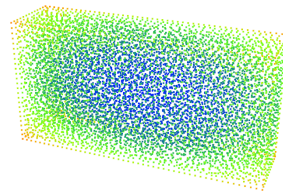
(α') Τοπολογία υποχωρίων



(β') Ρυθμός σύγκλισης για τα δύο υποπροβλήματα



(γ') Η λύση για το υποχωρίο σφαίρα



(δ') Η λύση για το υποχωρίο παραλληλεπίπεδο

Εικόνα 2: Ένα μοντέλο 3-διάστατο πρόβλημα με μερικώς επικαλυπτόμενα υποχωρία

Schwarz που υλοποιήθηκε στο FEniCS. Στην Εικόνα 2(β') παρουσιάζεται ο ρυθμός σύγκλισης για τα 2 υποπροβλήματα, έως την επίτευξη της επιθυμητής από το χρήστη ακρίβειας.

Κώδικας 1: Ορισμός του παραλληλεπίπεδου υποχωρίου αξιοποιώντας ένα αρχείο-σκελετό.

```

1 # user defined methods
2 def OverlappingWithOther(): pass
3 def userDefinedUFL(): pass
4 def userDefinedBoundaryCondition(): pass
5
6 # skeleton example
7 def ExtBC(x,on_boundary):
8     return on_boundary and not OverlappingWithOther()
9
10 def ExtIface(x,on_boundary):
11     return on_boundary and OverlappingWithOther()
12
13 class Problem(ConfigCommonProblem):
14     # Override the API methods init(), neighbours()
15     # and boundaries()
16     def init(self,*args,**kwargs):

```

```

17         self.domain_name = 'box'
18         mesh = Mesh(Box(-2,-1,-.5,2,1,.5),128)
19         self.V = FunctionSpace(mesh,'Lagrange',1)
20         self.a, self.L = userDefinedUFL(V)
21
22     # return a dictionary with interfaces for each neighbour
23     def neighbours(self):
24         interface = {}
25         interface[self.domain_name] = ExtIface
26         return interface
27
28     # return a list with all the external boundaries
29     def boundaries(self):
30         bc = DirichletBC(self.V, userDefinedBoundaryCondition(), ExtBC)
31         return [ bc ]

```

Ο Κώδικας 1 αντιστοιχεί στον ορισμό του παραλληλεπίπεδου υποχωρίου (box3D_1.py), ξεκινώντας από έναν κοινό (ανεξάρτητο του υποχωρίου) σκελετό. Παρόμοιος είναι και ο ορισμός του υποχωρίου που αντιστοιχεί στη σφαίρα (sphere3D_1.py).

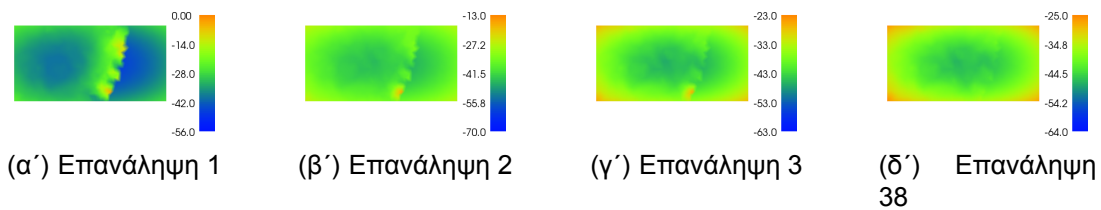
Κώδικας 2: Επίλυση 3D προβλήματος με 2 μερικώς αλληλεπικαλυπτόμενα υποχωρία.

```

1 from dolfin import *
2 import solverconfig
3 import solver
4
5 client = hmc.LocalClient()
6
7 import sphere3D_1 as sphere
8 import box3D_1 as box
9
10 sp = sphere.Problem(client=client)
11 bp = box.Problem(client=client)
12 subdomains=[ sp, bp ]
13
14 config = solverconfig.Config3D()
15 solver.solve(subdomains=subdomains, config=config)

```

Ο Κώδικας 2 οδηγεί τον επαναληπτικό επιλυτή, δεδομένων των ορισμών των υποχωρίων. Η απαιτούμενες αλλαγές από το χρήστη στους 2 σκελετούς απεικονίζονται με έντονους χαρακτήρες. Τα υποχωρία μπορούν να διαθέτουν – και διαθέτουν στο συγκεκριμένο παράδειγμα – διαφορετικά πλέγματα, διακριτοποιήσεις και PDEs.



Εικόνα 3: Στιγμιότυπα οριζόντιας τομής του παραλληλεπιπέδου σε διαφορετικές επαναλήψεις του επιλυτή

Στην Εικόνα 3 απεικονίζεται οριζόντια τομή του παραλληλεπιπέδου σε διαφορετικές επαναλήψεις (1, 2, 3 και 38) του επαναληπτικού επιλυτή. Μπορεί εύκολα κανείς να παρατηρήσει την αρχική έντονη ασυνέχεια στο τόξο τομής με τη σφαίρα, η οποία εξομαλύνεται στις μετέπειτα επαναλήψεις.

3.1.2 Υβριδική Αιτιοκρατική/Στοχαστική Μέθοδος για Προβλήματα MDMP

Ο κώδικας 3 επιδεικνύει τη χρήση της αιτιοκρατικής / στοχαστικής μεθόδου που υλοποιήθηκε στο FEniCS για τη λύση μιας PDE σε ένα εσωτερικό, τετράγωνο υποχωρίο ενός μεγαλύτερου χωρίου. Πριν τη λύση στο υποχωρίο εκτιμώνται, με στοχαστικό τρόπο, οι τιμές στη διεπαφή του εσωτερικού υποχωρίου. Οι αλλαγές που εισάγονται από τις επεκτάσεις μας διακρίνονται, και πάλι, με χρήση έντονων χαρακτήρων.

Κώδικας 3: Παράδειγμα χρήσης της μεθόδου montecarlo().

```

1 from dolfin import *
2 import hybridmc as hmc # the platform's Python module
3
4 def onbc(x,on_boundary):
5     return on_boundary
6
7 def mc_test_2D(Omega, Subdomain):
8     x, y = variable(Expression("x[0]")),
9           variable(Expression("x[1]"))
10    expr = (x)*(x-1)*(y)*(y-1)
11
12    mesh = Mesh(SubDomain,128)
13    V = FunctionSpace(mesh, 'Lagrange', 1)
14
15    u, v = TrialFunction(V), TestFunction(V)
16    f = -Laplacian(expr,x,y)
17    a = inner(grad(u), grad(v))*dx
18    L = f*v*dx

```

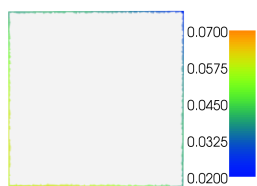
```

19
20 # get expressions as strings
21 f_expr, q_expr = hmc.tools.cppcode(expr,x,y),
22                   hmc.tools.cppcode(f,x,y)
23 mcbc, est = client.montecarlo(V, onbc, OpenCL=True, Omega=Omega,
24                               f=f_expr, q=q_expr)
25 sol_mc = Function(V)
26 solve(a==L, sol_mc, [ mcbc ])
27
28 if __name__ == '__main__':
29     Omega2D = [ 1., 1. ]
30     SubDomain2D = Rectangle(.4, .8, .4, .8)
31     client = hmc.LocalClient()
32     mc_test_2D(Omega2D, Subdomain2D)

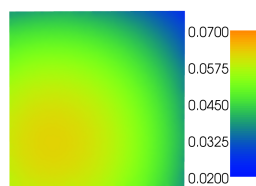
```

Το αντικείμενο `client` στη γραμμή 29 αναδεικνύει τη δυνατότητα τοπικής ή απομακρυσμένης υλοποίησης της μεθόδου.

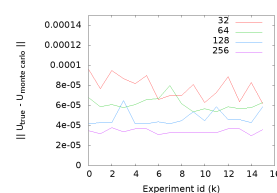
Η Εικόνα 4 περιλαμβάνει την εκτίμηση της στοχαστικής μεθόδου για τις τιμές στη διεπαφή του εσωτερικού υποχωρίου, τη λύση που επιτεύχθηκε από την υβριδική αιτιοκρατική / στοχαστική μέθοδο για το πρόβλημα-μοντέλο που περιγράφεται από τον Κώδικα 3, καθώς και το απόλυτο σφάλμα σε σχέση με έναν πλήρως αιτιοκρατικό επιλυτή για ένα σύνολο 16 πειραμάτων.



(α') Στοχαστική εκτίμηση τιμών στη διεπαφή του εσωτερικού υποχωρίου



(β') Λύση στο εσωτερικό υποχωρίο από τον υβριδικό επιλυτή



(γ') Σφάλμα λύσης σε σχέση με πλήρως αιτιοκρατικό επιλυτή. Διαφορετικές γραμμές αντιστοιχούν σε διαφορετικές πυκνότητες του υπολογιστικού πλέγματος

Εικόνα 4: Στοχαστική εκτίμηση, υβριδική λύση και σφάλμα λύσης σε σχέση με αιτιοκρατικό επιλυτή

3.1.3 Μέθοδος Geometric Contraction (GEO)

Ακολουθιακή υλοποίηση της GEO Σε αυτή την παράγραφο παρέχονται ενδεικτικά στιγματίσματα κώδικα από την υλοποίηση της μεθόδου GEO στο FEniCS.

Οι ανανεωμένες τιμές σε ένα σημείο της διεπαφής υπολογίζονται ως εξής:

$$u^{(k+1)}(x) = u^{(k)}(x) - \rho \left(\frac{\partial u_L^{(k)}(x)}{\partial n} - \frac{\partial u_R^{(k)}(x)}{\partial n} \right), k = 1, 2, \dots \quad (1)$$

που k είναι η επανάληψη, u είναι η υπολογισμένη λύση στο σημείο της διεπαφής x , $\frac{\partial u_L^{(k)}(x)}{\partial n} - \frac{\partial u_R^{(k)}(x)}{\partial n}$ είναι οι τιμές των προς τα έξω κανονικών παραγώγων στα δύο γειτονικά υποπεδία και ρ είναι η παράμετρος χαλάρωσης που χρησιμοποιείται για να επιταχύνει τη σύγκλιση. Μια νέα επανάληψη ξεκινάει με το που περαστούν οι χαλαρωμένες τιμές των διεπαφών πίσω στα υπο-πεδία.

Πιο αναλυτικά, τμήματα του FEniCS προγράμματος που υλοποιεί τα παραπάνω παρουσιάζεται στη συνέχεια. Αρχικά γίνεται η δημιουργία των υπο-πεδίων. Ενδεικτικά παρουσιάζουμε τη δημιουργία του αριστερού υπο-πεδίου για το ομοιόμορφο πρόβλημα και για την περίπτωση, όπου $h = 0.5$.

```
1 meshdown = RectangleMesh(0, 0, 0.3333, 2, 7, 40, 'left')
2 Vdown = FunctionSpace(meshdown, 'Lagrange', 1)
```

Στη συνέχεια ορίζουμε τις οριακές συνθήκες και τις συνθήκες των διεπαφών. Ενδεικτικά, η πρώτη (αριστερή) διεπαφή:

```
1 int1_expr = Expression('4135.5*x[1]-6077.6')
2 class Interface1(SubDomain):
3     def inside(self, x, on_boundary):
4         return on_boundary and abs(x[0] - 0.3333) < tol
5             and abs(x[1] - 2) > tol
6             and abs(x[1] - 1.5) > tol
7 Gamma_4_left = DirichletBC(Vdown, int1_expr, Interface1())
```

Ακολουθεί ο ορισμός του παραμετρικού προβλήματος. Πάλι για το αριστερό υπο-πεδίο του ομοιόμορφου προβλήματος:

```
1 u = TrialFunction(Vdown)
2 v = TestFunction(Vdown)
3 f = interpolate(f_expr, Vdown)
4 a = (inner(nabla_grad(u), nabla_grad(v)) + g2*inner(u, v))*dx
5 L = f*v*dx
```

Και στη συνέχεια ο υπολογισμός της λύσης:

```
1 u = Function(Vdown)
2 solve(a == L, u, bcsgdown)
```

Και της παραγώγου:

```
1 V_g = VectorFunctionSpace(meshdown, 'Lagrange', 1)
2 v = TestFunction(V_g)
3 w = TrialFunction(V_g)
```

```

4 a = inner(w, v)*dx
5 L = inner(grad(u), v)*dx
6 grad_u = Function(V_g)
7 solve(a == L, grad_u)

```

Παρουσιάζεται ο τρόπος με τον οποίο λαμβάνουμε τις τιμές της λύσης και της παραγώγου στα σημεία των διεπαφών:

```

1 vertex_to_dof_map = Vdown.dofmap().vertex_to_dof_map(meshdown)
2 dof_to_vertex_map = Vdown.dofmap().dof_to_vertex_map(meshdown)
3 m = u.vector()
4 dofs_at_vertices = m[dof_to_vertex_map]
5 coor = meshdown.coordinates()
6
7 grad_u_x, grad_u_y = grad_u.split(deepcopy=True)
8
9 for vdown in vertices(meshdown):
10     ldown = coor[vdown.index()]
11     ldown[0], ldown[1],
12     dofs_at_vertices[vdown.index()],
13     dofs_at_vertices_g_x_down[vdown.index()],
14     dofs_at_vertices_g_y_down[vdown.index()]]

```

Και ακολουθεί ο τρόπος που τις επαναθέτουμε, αφού τις επεξεργαστούμε:

```

1 downbound=(dofs_at_vertices[vdown.index()]+
2             dofs_at_vertices_up[vup.index()])/2+
3             omega*(-dofs_at_vertices_g_x_down[vdown.index()]+
4                  dofs_at_vertices_g_x_up[vup.index()])
5
6 upbound=(dofs_at_vertices[vdown.index()]+
7           dofs_at_vertices_up[vup.index()])/2+
8           omega*(-dofs_at_vertices_g_x_down[vdown.index()]+
9                  dofs_at_vertices_g_x_up[vup.index()])
10
11 u.vector().set_local(dofs_at_vertices.array()[vertex_to_dof_map])
12 downbound=Function(Vdown)
13 downbound.assign(u)

```

Παράλληλη υλοποίηση της GEO Ενδεικτικά τμήματα του κώδικα της παράλληλης υλοποίησης της GEO, δίνονται παρακάτω.

Ο client αρχικοποιεί την επικοινωνία:

```

1 def __init__(self):
2     self.credentials = pika.PlainCredentials('test', 'test')
3     self.connection = pika.BlockingConnection(

```

```

4         pika . ConnectionParameters ( '150.140.139.89' , 5672 ,
5                                         ' / ' ,
6                                         self . credentials ))
7     self . channel = self . connection . channel ()
8     result = self . channel . queue_declare ( exclusive = True )
9     self . callback_queue = result . method . queue
10    self . channel . basic_consume ( self . on_response , no_ack = True ,
11                                    queue = self . callback_queue )
12    self . connection2 = pika . BlockingConnection (
13        pika . ConnectionParameters ( '150.140.139.89' ,
14                                       5672 ,
15                                       ' / ' ,
16                                       self . credentials ))
17    self . channel2 = self . connection2 . channel ()
18    result2 = self . channel2 . queue_declare ( exclusive = True )
19    self . callback_queue2 = result2 . method . queue
20    self . channel2 . basic_consume ( self . on_response ,
21                                    no_ack = True ,
22                                    queue = self . callback_queue2 )

```

Περιμένει τις απαντήσεις από τους servers:

```

1 def on_response ( self , ch , method , props , body ) :
2     # print "on_response result ch=%s guess=%s" % ( ch , self . channel )
3     if ch == self . channel :
4         self . response = body
5     else :
6         self . response2 = body

```

Αφού πρώτα τους στείλει τις τιμές στις διεπαφές που υπολόγισε επιλύοντας το πρόβλημα στην εκάστοτε επανάληψη:

```

1 self . response = None
2 self . corr_id = str ( uuid . uuid4 () )
3 self . channel . basic_publish (
4     exchange = ' ' ,
5     routing_key = 'hello ' ,
6     properties = pika . BasicProperties (
7         reply_to = self . callback_queue ,
8         correlation_id = self . corr_id ) ,
9     body = inter2 )

```

Αντίστοιχα ο κάθε server επιλύει το πρόβλημά του, λαμβάνει από τον client τη λύση και την παράγωγο που υπολόγισε στη διεπαφή και υπολογίζει τις καινούργιες τιμές στη διεπαφή:

```

1 response = ( interfaceup + interfaced ) / 2 +
2             omega * ( - gradientd + gradientup )

```

Το αποτέλεσμα αυτό το χρησιμοποιεί για την επόμενη επανάληψή του. Το στέλνει, όμως και στον client για να προχωρήσει και αυτός με την επόμενη επανάληψη του.

```
1 ch.basic_publish(exchange='',
2   routing_key=props.reply_to,
3   properties=pika.BasicProperties(correlation_id =
4   props.correlation_id),
5   body=json.dumps(response.tolist()))
6 ch.basic_ack(delivery_tag = method.delivery_tag)
```

3.2 Αξιοποίηση Σύγχρονων Μέσων Αποθήκευσης

Τα πειράματα πραγματοποιήθηκαν σε δύο σταθμούς εργασίας DELL Precision T3500 με 8GB of DDR3 RAM και έναν τετρα-πύρηνο Intel Xeon W3550 3.06GHz το καθένα. Ο πρώτος σταθμός εργασίας διαθέτει έναν οδηγό SSD Intel 520 240GB, SATA-III, και έναν οδηγό OCZ Revodrive 350 PCIe 480GB ως επιπλέον αποθηκευτικό μέσο. Ο δεύτερος σταθμός εργασίας διαθέτει έναν σκληρό Seagate Barracuda 7200rpm 1TB. Και οι δύο σταθμοί χρησιμοποιούν Centos 6.5 με kernel 2.6.32-431.el6.x86_64 για λειτουργικό σύστημα και έχουν διαμορφωθεί ώστε να χρησιμοποιούν μέχρι 8GB swap.

Οι αραιοί πίνακες που χρησιμοποιήθηκαν στα πειράματα προέρχονται από τις συλλογές, του Πανεπιστημίου της Φλόριντα¹ και του ερευνητικού προγράμματος GRID-TLSE². Οι ιδιότητές των αραιών πινάκων συνοψίζονται στον Πίνακα 1.

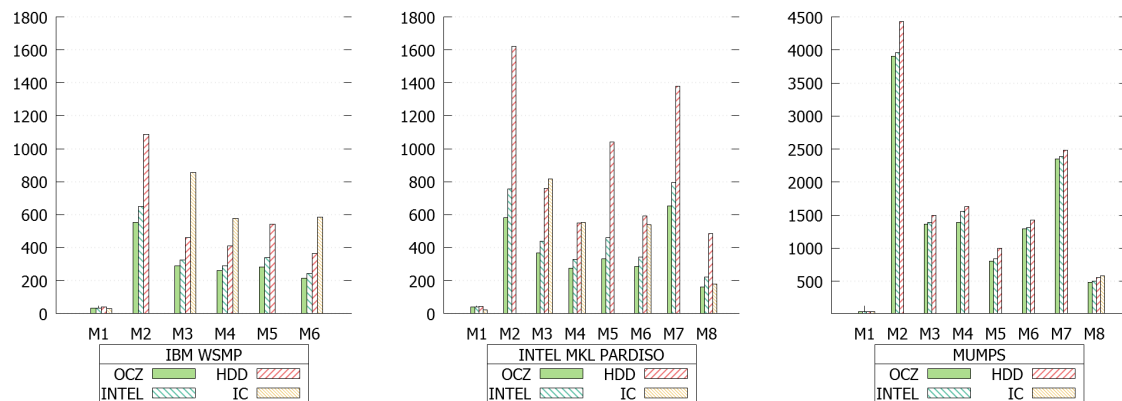
MATRIX		TYPE	$N(A)$	$NNZ(A)$	DESCRIPTION
INLINE_1	M1	RSPD	5.04E+05	1,87E+07	Structural
ASTER_PERF011A	M2	RSI	8,54E+05	7,11E+07	Structural engineering
AUDIkw_1	M3	RSI	9,44E+05	3,93E+07	Automotive model
NICE20MC	M4	RSI	7,16E+05	2,81E+07	Earthquake analysis
FLAN_1565	M5	RSPD	1,56E+06	1,14E+08	Structural
STOCF_1465	M6	RSPD	1,47E+06	2,10E+07	Fluid dynamics
ATMOSMODL	M7	RUI	1,49E+06	1,03E+07	Atmospheric modeling
ASTER_PERF002C	M8	RSI	1,0E+6	3,79E+07	Structural engineering

Πίνακας 1: Αραιοί Πίνακες

Για την επίλυση των συστημάτων στην κύρια μνήμη (in-core) χρησιμοποιήθηκε ο σταθμός εργασίας με τους δίσκους στερεάς κατάστασης. Στην Εικόνα 5 παρουσιάζονται οι χρόνοι επίλυσης (wall-clock time) για τα τρία πακέτα λογισμικού.

¹<http://www.cise.ufl.edu/research/sparse/matrices/>

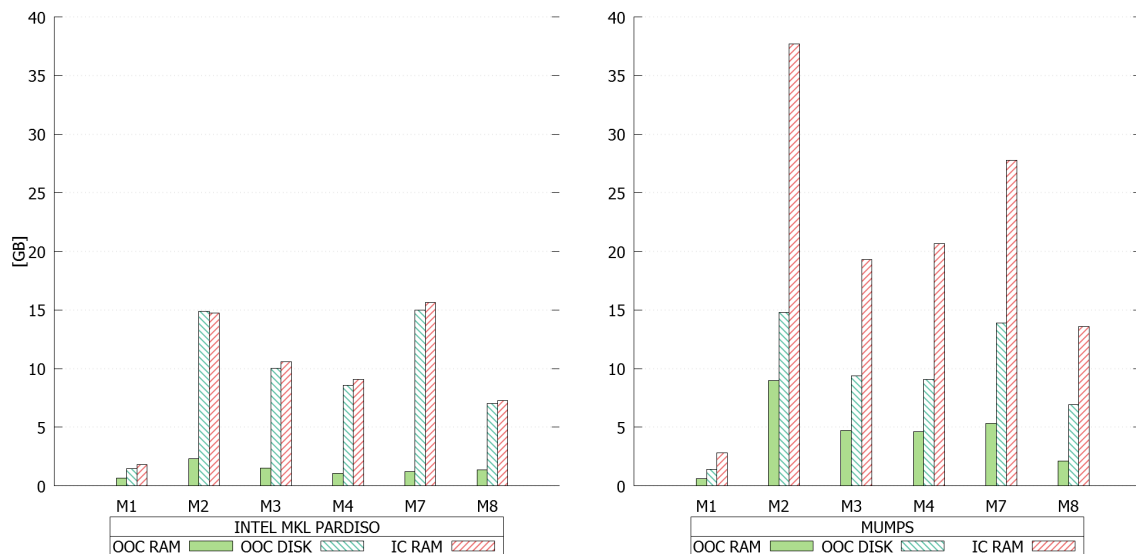
²<http://gridtlse.enseiht.fr/>



Εικόνα 5: Χρόνοι επίλυσης στη RAM σε HDD και flash SSD

Σε όλες σχεδόν τις περιπτώσεις η χρήση της flash οδηγεί σε σημαντική μείωση του χρόνου επίλυσης των συστημάτων σε σχέση με τον μαγνητικό δίσκο. Στο PARDISO, η επίλυση στην flash είναι δύο με τρεις φορές γρηγορότερη απ' ότι στον σκληρό δίσκο, στο WSMP ως και δύο φορές ταχύτερη, ενώ μικρότερη είναι η διαφορά στο MUMPS, αλλά παραμένει σημαντική. Σε όλες τις περιπτώσεις η επιτάχυνση είναι μικρότερη για το σύστημα M1, καθώς το μικρό του μέγεθός του ελαχιστοποιεί την πρόσβαση στην εξωτερική μνήμη. Η επίλυση στην κύρια μνήμη είναι γρηγορότερη εφόσον το μέγεθος την μνήμης που απαιτείται δεν υπερβαίνει την διαθέσιμη RAM, διαφορετικά χρησιμοποιείται εικονική μνήμη (swap) και η επίλυση καθυστερεί σημαντικά (π.χ. περίπτωση M3 & M4 στο WSMP και PARDISO). Στην περίπτωση δε, που η απαιτούμενη μνήμη υπερβαίνει το άθροισμα κύριας και εικονικής μνήμης δεν είναι δυνατή η επίλυση του συστήματος, για παράδειγμα μόνο τα συστήματα M1 και M8 κατέστη δυνατό να επιλυθούν με το MUMPS στην κύρια μνήμη. Οι απαιτήσεις σε κύρια και δευτερεύουσα μνήμη για το PARDISO και το MUMPS παρουσιάζονται στην Εικόνα 6, για το WSMP δεν ήταν δυνατό να έχουμε άμεσα συγκρίσιμες πληροφορίες.

Συνοψίζοντας, τα αποτελέσματα δείχνουν ότι η επίλυση μεγάλων αραιών συστημάτων γραμμικών εξισώσεων μπορεί να ωφεληθεί σημαντικά από την αξιοποίηση νέων τεχνολογιών, όπως οι μνήμες flash. Ωστόσο, είναι αναγκαία η περαιτέρω μελέτη των αλγορίθμων εξωτερικής μνήμης με σκοπό την βελτιστοποίησή τους γι' αυτό το νέο μέσο. Επιπλέον, τα πειράματα δείχνουν ότι η αποδοτική επίλυση ενός αραιού συστήματος απαιτεί την προσεκτική επιλογή του κατάλληλου αλγορίθμου (in-core/out-of-core) ανάλογα με το διαθέσιμο υλικό (hardware).



Εικόνα 6: Απαιτήσεις σε κύρια και δευτερεύουσα μνήμη

4 Παραδοτέα

Τα παραδοτέα της παρούσας δράσης, σύμφωνα και με το τεχνικό δελτίο του έργου, είναι τα ακόλουθα:

Ετήσια τεχνική έκθεση περιγραφής αποτελεσμάτων: το παρόν κείμενο.

Πρότυπο λογισμικό για Περιβάλλον Επίλυσης Προβλημάτων το οποίο περιλαμβάνει τα ακόλουθα επιμέρους στοιχεία:

- Προσθήκες στην πλατφόρμα FEniCS για την υποστήριξη προβλημάτων MDMP (μέθοδοι Schwarz, υβριδική αιτιοκρατική/στοχαστική, GEO, ROB).
- Προσθήκες στην πλατφόρμα FEniCS για την υποστήριξη εναλλακτικών αρχιτεκτονικών (επέκταση Whale).

Πλέον των παραδοτέων που προέβλεπε το τεχνικό δελτίο, στα πλαίσια της δράσης παρήχθη και η ακόλουθη δημοσίευση:

- Athanasios Fevgas, Panagiota Tsompanopoulou, Panayiotis Bozanis, "Exploring the Performance of Out-of-Core Linear Algebra Algorithms in Flash based Storage", 6th International Conference on Numerical Analysis (NumAn 2014), 02-05 Sep 2014, Chania, Crete, Greece.

	ΚΕΟ 1	ΚΕΟ 2	ΚΕΟ 3
Υλοποίηση Μεθόδου GEO		x	x
Υλοποίηση Μεθόδου ROB		x	x
Υλοποίηση Μεθόδου Schwarz	x	x	
Υλοποίηση Μεθόδου Hybrid		x	x
Υποστήριξη Flash Storage		x	

Πίνακας 2: Συνεργασίες στα πλαίσια της Δράσης 4.1.

5 Συνεργασίες

Η Δράση 4.1 είναι κεντρική στο έργο MATENVMED καθώς καλείται να ενοποιήσει και να συγκεράσει τις ερευνητικές προσπάθειες που αναπτύσσονται στις υπόλοιπες δράσεις του έργου. Σε αυτά τα πλαίσια, ενθάρρυνε τη συνεργασία μεταξύ και των 3 ερευνητικών ομάδων. Πιο συγκεκριμένα, στον Πίνακα 2 συνοψίζονται οι βασικές δραστηριότητες που αναπτύχθηκαν στα πλαίσια της δράσης και αναφέρονται οι ομάδες που συμμετείχαν.

6 Μελλοντικές Δράσεις

Στα πλαίσια της παρούσας δράσης κατά το 2014 επεκτάθηκε το FEniCS με μεθόδους απαραίτητες για την υποστήριξη των εφαρμογών στις οποίες στοχεύει το έργο MATENVMED. Επίσης, εξετάστηκε η αξιοποίηση από το FEniCS μοντέρνων, γρήγορων συσκευών αποθήκευσης.

Βασικές επιδιώξεις για το επόμενο έτος είναι:

- Η υποστήριξη και η αξιοποίηση υποδομών και υπηρεσιών cloud από το προτεινόμενο περιβάλλον επίλυσης προβλημάτων.
- Η αξιοποίηση του περιβάλλοντος για την αντιμετώπιση των κεντρικών εφαρμογών του έργου.

Αναφορές

- [1] Patrick R Amestoy, Iain S Duff, Jean-Yves L'Excellent, and Jacko Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.
- [2] Patrick R Amestoy, Abdou Guermouche, Jean-Yves L'Excellent, and Stephane Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel computing*, 32(2):136–156, 2006.

- [3] Haim Avron and Arpan Gupta. Managing data-movement for effective shared-memory parallelization of out-of-core sparse solvers. In *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*, pages 1–11. IEEE, 2012.
- [4] David M. Beazley. Swig: An easy to use tool for integrating scripting languages with c and c++. In *Proceedings of the 4th Conference on USENIX Tcl/Tk Workshop, 1996 - Volume 4*, TCLTK'96, pages 15–15, Berkeley, CA, USA, 1996. USENIX Association.
- [5] X. Cai. Overlapping domain decomposition methods. In HansPetter Langtangen and Aslak Tveito, editors, *Advanced Topics in Computational Partial Differential Equations*, volume 33 of *Lecture Notes in Computational Science and Engineering*, pages 57–95. Springer Berlin Heidelberg, 2003.
- [6] Olaf Schenk. Scalable parallel sparse lu factorization methods on shared memory multiprocessors. Technical report, ETH. Zurich, 2000.
- [7] Olaf Schenk and Klaus Gartner. Solving unsymmetric sparse systems of linear equations with pardiso. *Future Generation Computer Systems*, 20(3):475–487, 2004.
- [8] John E. Stone, David Gohara, and Guochun Shi. Opencl: A parallel programming standard for heterogeneous computing systems. *IEEE Des. Test*, 12(3):66–73, May 2010.
- [9] Panagiota Tsompanopoulou and Emmanouil Vavalis. "analysis of an interface relaxation method for composite elliptic differential equations". *Journal of Computational and Applied Mathematics*, 226(2):370–387, 2009.
- [10] Manolis Vavalis. Implementing hybrid PDE solvers, 08 2014.