

Τελική Τεχνική Έκθεση

Έτος 2015



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	4
2 Μεθοδολογία	6
2.1 Επιλογή βασικού Περιβάλλοντος Επίλυσης Προβλημάτων (ΠΕΠ)	6
2.1.1 Η βιβλιοθήκη Dolfin	7
2.1.2 Πεπερασμένα Στοιχεία και Πλέγματα	8
2.1.3 Η Unified Form Language (UFL)	8
2.1.4 Αξιοποίηση Λογισμικού Αιχμής στο Επίπεδο της Γραμμικής Άλγεβρας	10
2.1.5 Εγκατάσταση FEniCS σε Cluster & Cloud	10
2.2 Επέκταση του ΠΕΠ για την Υποστήριξη της Κλάσης Προβλημάτων Ενδιαφέροντος του Έργου	11
2.2.1 Μέθοδος Schwarz για Προβλήματα με Επικαλυπτώμενα Υποχωρία	11
2.2.2 Υβριδική Αιτιοκρατική/Στοχαστική Μέθοδος για Προβλήματα MDMP	12
2.2.3 Μέθοδος Geometric Contraction (GEO)	13
2.2.4 Μέθοδος ROB	14
2.3 Επέκταση του ΠΕΠ για την Αξιοποίηση Σύγχρονων Υπολογιστικών Συστημάτων (Ετερογενών, Υποδομής Cloud και Σύγχρονων Μέσων Αποθήκευσης)	15
2.3.1 Αξιοποίηση Ετερογενών Συστημάτων και Σύγχρονων Μέσων Αποθήκευσης	15
2.4 Χρήση Σύγχρονου Υλικού για την Επίλυση Μεγάλων Αραιών Γραμμικών Συστημάτων	17
2.4.1 Τεχνολογίες Δευτερεύουσας Μνήμης και Αλγόριθμοι Out-of-Core	17
2.4.2 Σύγχρονες Πλατφόρμες Υλικού και Αραιά Γραμμικά Συστημάτα	17
2.4.3 Αλγόριθμοι Εξωτερικής Μνήμης για Μνήμες Τύπου Flash	18
2.4.4 Αξιοποίηση Υποδομής Cloud - Web Services	18
2.4.5 Αξιοποίηση Υποδομής Cloud - Πλατφόρμα IRaaS	19
3 Πειραματικά Αποτελέσματα	25
3.1 Επέκταση του ΠΕΠ για την Υποστήριξη της Κλάσης Προβλημάτων Ενδιαφέροντος του Έργου	25
3.1.1 Μέθοδος Schwarz για Προβλήματα με Επικαλυπτώμενα Υποχωρία	25

3.1.2	Υβριδική Αιτιοκρατική/Στοχαστική Μέθοδος για Προβλήματα MDMP	30
3.1.3	Μέθοδος Geometric Contraction (GEO)	32
3.2	Χρήση Σύγχρονων Τεχνολογιών Υλικού για την Επίλυση Μεγάλων Αραιών Γραμμικών Συστημάτων	35
3.2.1	Τεχνολογίες Δευτερεύουσας Μνήμης και Αλγόριθμοι Out-of-Core	35
3.2.2	Σύγχρονες Πλατφόρμες Υλικού και Αραιά Γραμμικά Συστήματα	37
3.2.3	Αλγόριθμοι Εξωτερικής Μνήμης για Μνήμες Τύπου Flash	39
3.3	Επέκταση του ΠΕΠ για την Αξιοποίηση Υποδομής Cloud	42
3.3.1	Αξιοποίηση Υποδομής Cloud - Web Services	42
3.3.2	Αξιοποίηση Υποδομής Cloud - Πλατφόρμα IRaaS	44
4	Παραδοτέα	49
5	Συνεργασίες	51
6	Μελλοντικές Δράσεις	51

1 Σκοπός

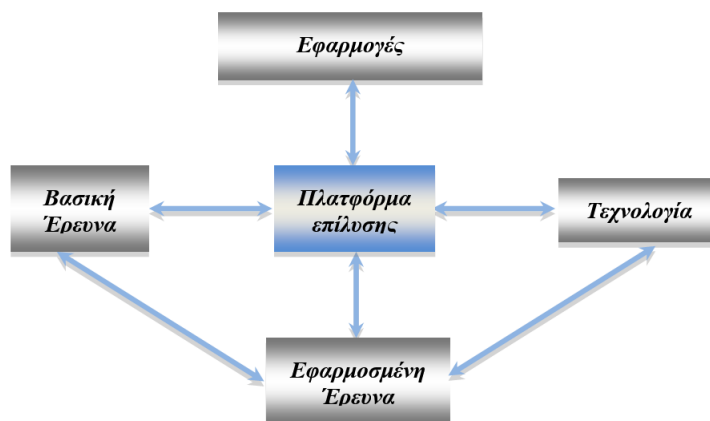
Τα τελευταία χρόνια αποτελεί κοινή διαπίστωση των εμπλεκομένων με την σχεδίαση και ανάπτυξη λογισμικού για ευρείας κλίμακας επιστημονικούς υπολογισμούς η αναγκαιότητα για την ανάπτυξη:

- Ενός λειτουργικού περιβάλλοντος επίλυσης προβλημάτων (Problem Solving Environment - PSE) το οποίο, εστιαζόμενο στα ιδιαίτερα χαρακτηριστικά της κλάσης των υπό μελέτη προβλημάτων, διευκολύνει την αποδοτική χρήση (και επανάχρηση) των λογισμικών μονάδων που είτε προϋπάρχουν είτε αναπτύσσονται.
- Μιας καινοτόμου πρακτικής ανάπτυξης λογισμικού η οποία θα μας επιτρέψει να εκμεταλλευτούμε τις δυνατότητες των σύγχρονων υπολογιστικών μηχανών.

Η διαπίστωση της αναγκαιότητας αυτής, η οποία αποκτά ιδιαίτερη σημασία για σύνθετα προβλήματα όπως αυτά με τα οποία ασχολείται το MATENVMED, αποτέλεσε την κινητήρια δύναμη για αρκετές προηγούμενες ερευνητικές προσπάθειες (όπως για παράδειγμα οι [13, 15, 18, 25, 37, 17]). Όμως, ελάχιστες από τις προσπάθειες αυτές πρότειναν λύσεις και πρακτικές με τον απαραίτητο βαθμό καινοτομίας για να ανταποκριθούν στις απαιτήσεις των προβλημάτων μας.

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 (Problem Solving Environments, PSEs) 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.

Τα σύγχρονα PSEs αποτελούνται από πολλές και διαφορετικές υπομονάδες λογισμικού (modules). Ο ορισμός, η συσχέτιση και επικοινωνία μεταξύ τους πραγματοποιούνται σε υψηλό επίπεδο. Τα σύγχρονα PSEs οφείλουν να στοχεύουν στην αποδοτική τους εκτέλεση, με όσο το δυνατό περισσότερο διάφανο στον τελικό χρήστη τρόπο, σε συστήματα τελευταίας τεχνολογίας (για παράδειγμα ετερογενή συστήματα με GPUs και FPGAs), όπως επίσης και Clusters ή ακόμα και στο Cloud. Επίσης οφείλουν να εκμεταλλεύονται οποιαδήποτε επιπλέον δυνατότητα παρέχεται από το υλικό, όπως για παράδειγμα γρήγορα μέσα μόνιμης



Εικόνα 1: Συνολική αρχιτεκτονική MATENVMED

αποθήκευσης (flash storage). Έχουν πολλές επιπλέον ιδιότητες σε σχέση με τα παλαιότερα PSEs όπως ευκαμψία, αποδοτικότητα, συνέπεια, ασφάλεια, δυναμικότητα στη δημιουργία νέων υπομονάδων και στην χρήση των πλέον αποδοτικών αλλά και ισχυρών υπολογιστικών πόρων, τόσο τοπικά όσο και απομακρυσμένα.

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, καλύπτοντας τις δύο προαναφερθείσες αναγκαιότητες. Όπως φαίνεται στην Εικόνα 1, η πλατφόρμα επίλυσης προβλημάτων αποτελεί κεντρικό σημείο του έργου, καθώς καλείται να ενοποιήσει και να συγκεράσει τις ερευνητικές προσπάθειες που αναπτύσσονται στις υπόλοιπες δράσεις του έργου.

Πιο συγκεκριμένα, οι σημαντικότεροι στόχοι της παρούσας δράσης είναι οι ακόλουθοι:

- Επιλογή βασικού Περιβάλλοντος Επίλυσης Προβλημάτων (ΠΕΠ).
- Επέκταση του ΠΕΠ για την υποστήριξη της κλάσης προβλημάτων ενδιαφέροντος του έργου.
- Επέκταση του ΠΕΠ για την αξιοποίηση σύγχρονων υπολογιστικών συστημάτων και πιο συγκεκριμένα ετερογενών συστημάτων, υποδομής cloud, αλλά και 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 Επιλογή βασικού Περιβάλλοντος Επίλυσης Προβλημάτων (ΠΕΠ)

Βασική επιλογή της ομάδας του έργου αποτέλεσε η απόφαση να εξαντληθεί κάθε περιθώριο αξιολόγησης και αξιοποίησης υπαρχόντων περιβαλλόντων επίλυσης προβλημάτων (ΠΕΠ). Με τον τρόπο αυτό, θα ήταν εφικτό να επικεντρωθεί η προσπάθεια της ομάδας στο ερευνητικό αντικείμενο του έργου, δηλαδή την υποστήριξη εφαρμογών MDMP (Multi-Domain, Multi-Physics), έναντι της εξαρχής υλοποίησης ενός νέου ΠΕΠ. Βασικά κριτήρια κατά τη φάση αξιολόγησης των υπαρχόντων ΠΕΠ ήταν τα ακόλουθα:

- Υποστήριξη επίλυσης διαφορικών εξισώσεων.
- Ανοιχτού κώδικα, ώστε να είναι δυνατή η ανάπτυξη της απαιτούμενης υποστήριξης για τις εφαρμογές και τις πλατφόρμες ενδιαφέροντος του έργου. Με άδεια χρήσης που δε θα απαγορεύει τις απαιτούμενες από το έργο δραστηριότητες.
- Ευρεία και ενεργή κοινότητα ανάπτυξης και υποστήριξης.
- Δυνατότητα χρήσης τεχνολογίας (βιβλιοθηκών) αιχμής στο επίπεδο της γραμμικής άλγεβρας.
- Απλή διεπαφή χρήστη.
- Δυνατότητα χρήσης ως ανεξάρτητο τμήμα λογισμικού στα πλαίσια μεγαλύτερων εφαρμογών.

Το πακέτο FEniCS [21] βρέθηκε να ικανοποιεί κατά τον καλύτερο τρόπο όλα τα παραπάνω κριτήρια. Αποτελεί συλλογή ελεύθερων εργαλείων λογισμικού εξειδικευμένων στην αυτόματη και αποδοτική επίλυση διαφορικών εξισώσεων. Η παράγραφος αυτή προσφέρει μια σύντομη επισκόπηση των τμημάτων του FEniCS που έχουν ενδιαφέρον στα πλαίσια του έργου. Λεπτομερέστερη περιγραφή μπορεί να βρεθεί στο βιβλίο του FEniCS [21] το οποίο είναι διαθέσιμο ελεύθερα στο βασικό ιστότοπο¹ του FEniCS.

¹<http://fenicsproject.org/book/index.html#book>

Τα σημαντικότερα χαρακτηριστικά του FEniCS, τα οποία αυτοματοποιούν τις φάσεις δημιουργίας και επίλυσης των συστημάτων που απαιτούνται για την επίλυση διαφορικών εξισώσεων, είναι τα ακόλουθα:

Dolfin [22]: Η βασική διεπαφή χρήστη. Αποκρύπτει τις λεπτομέρειες υλοποίησης των επιμέρους στοιχείων του FEniCS και προσφέρει ένα πλήρες API στον χρήστη, επιτρέποντάς του να αξιοποιήσει όλες τις υπηρεσίες της πλατφόρμας.

UFL [3]: Η Unified Form Language είναι η σημειογραφία έκφρασης των PDEs που προσφέρει η διεπαφή χρήστη. Η σημειογραφία αυτή βρίσκεται πολύ κοντά στη μαθηματική σημειογραφία (περισσότερες λεπτομέρειες για τη UFL θα συζητηθούν στην παράγραφο 2.1.3).

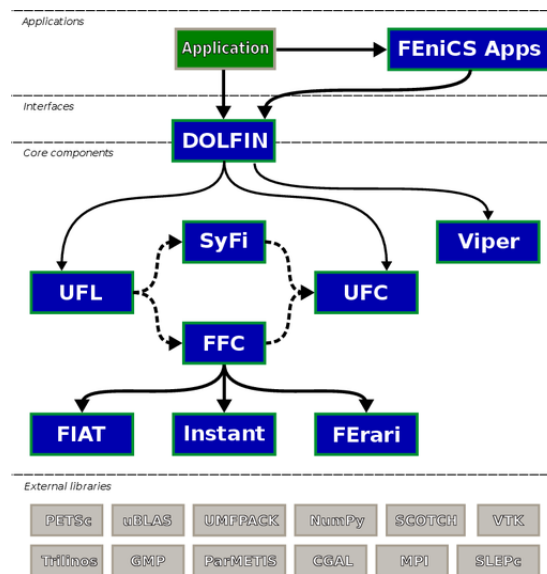
Πλήθος χώρων πεπερασμένων στοιχείων ώστε ο χρήστης να επιλέξει το κατάλληλο για το εκάστοτε πρόβλημα, τόσο για 2D, όσο και για 3D χωρία.

Πλήθος πακέτων γραμμικής άλγεβρας που υποστηρίζουν διαφορετικές οικογένειες επιλυτών. Τα περισσότερα είναι παραμετροποιήσιμα από το χρήστη.

Επιπλέον, το FEniCS αποτελεί ένα πλήρες και καλά ελεγμένο (τόσο ατομικά όσο και σε επίπεδο διαλειτουργικότητας) σύνολο εργαλείων που διευκολύνει την ανάπτυξη νέων μεθόδων. Η Εικόνα 2 απεικονίζει τη δομή του FEniCS. Η ανάπτυξη στα πλαίσια του MATENVMED επικεντρώνεται κυρίως στη διεπαφή Dolfin του FEniCS και την ικανότητα υποστήριξης νέων βιβλιοθηκών στα κατώτερα επίπεδα με διάφανο τρόπο.

2.1.1 Η βιβλιοθήκη Dolfin

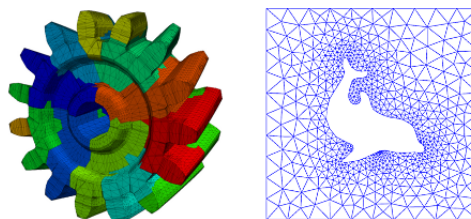
Το Dolfin [22] είναι βιβλιοθήκη C++/Python που λειτουργεί ως η βασική διεπαφή χρήστη του FEniCS. Υλοποιεί μεγάλο μέρος της λειτουργικότητας του FEniCS, συμπεριλαμβανομένων δομών δεδομένων και αλγορίθμων για την παραγωγή πλεγμάτων (meshes) και τη συγκρότηση συστημάτων πεπερασμένων στοιχείων. Τελικά παρέχει ένα ΠΕΠ για μοντέλα βασισμένα σε PDEs. Προκειμένου να παρέχει μια απλή και συνεπή διεπαφή χρήστη, το Dolfin λειτουργεί ως περιτύλιγμα (wrapper) της λειτουργικότητας άλλων τμημάτων του FEniCS καθώς και εξωτερικού λογισμικού και είναι υπεύθυνο για τη σωστή επικοινωνία μεταξύ τους.



Εικόνα 2: Η δομή του FEniCS [21, σελ. 172]

2.1.2 Πεπερασμένα Στοιχεία και Πλέγματα

Το FEniCS προσφέρει εκτεταμένη βιβλιοθήκη πεπερασμένων στοιχείων. Επιπλέον παρέχει πλήρως κατανομημένα πλέγματα σε 1 (διαστήματα), 2 (τρίγωνα) και 3 (τετράεδρα) διαστάσεις. Τα πλέγματα είναι δυνατό να γίνονται λεπτομερέστερα με προσαρμοστικό τρόπο. Επιπλέον, προσφέρεται υποστήριξη για παράλληλο υπολογισμό μέσω διαίρεσης του πλέγματος σε υποπλέγματα. Ένα παράδειγμα φαίνεται στην Εικόνα 3.



Εικόνα 3: Πλέγματα 3D & 2D [21, σελ. 214, 205]

2.1.3 Η Unified Form Language (UFL)

Η UFL [3] είναι ένα από τα κεντρικά συστατικά στοιχεία του FEniCS. Πρόκειται για γλώσσα πεδίου για την έκφραση διακριτοποιήσεων πεπερασμένων

στοιχείων, την έκφραση μη γραμμικών PDEs και την αυτόματη διαφορίση εκφράσεων και φορμών. Πιο συγκεκριμένα, η γλώσσα ορίζει μια ευέλικτη διεπαφή χρήστη για τη μοντελοποίηση χώρων πεπερασμένων στοιχείων και την έκφραση ασθενών διατυπώσεων (weak formulations) σε σημειογραφία που προσεγγίζει τη μαθηματική. Μπορεί να χειριστεί περίπλοκα προβλήματα με εύκολο, κομψό και αποδοτικό τρόπο.

Οι ασθενείς διατυπώσεις αποτελούν σημαντικό εργαλείο για την ανάλυση εξισώσεων. Επιτρέπουν τη μεταφορά αρχών και μεθόδων γραμμικής άλγεβρας στην επίλυση προβλημάτων σε άλλα πεδία, όπως στις μερικές διαφορικές εξισώσεις. Σε μία ασθενή διατύπωση η εξίσωση δεν απαιτείται πλέον να ικανοποιείται σε κάθε σημείο, αλλά αντίθετα έχει ασθενείς λύσεις ως προς διανύσματα ή συναρτήσεις ελέγχου [32, p. 24]. Αυτή η προσέγγιση είναι, για παράδειγμα, ισοδύναμη με τη διατύπωση ενός προβλήματος ώστε να απαιτεί λύση στη μορφή μιας κατανομής.

Η φιλική προς το χρήστη σημειογραφία και η υποστήριξη για ταχεία ανάπτυξη αποτελούν κεντρικές αξίες της σχεδίασης της UFL. Η χρήση σημειογραφίας κοντά στη μαθηματική επιτρέπει την εύκολη έκφραση ιδεών, μειώνοντας σημαντικά την πιθανότητα εισαγωγής σφαλμάτων στον κώδικα.

Παρακάτω ακολουθεί παράδειγμα έκφρασης της εξίσωσης Poisson σε UFL [21, p. 3]:

$$\underbrace{\int_{\Omega} \nabla u \cdot \nabla v \, dx}_{a(u,v)} = \underbrace{\int_{\Omega} f v \, dx}_{L(v)} \quad \forall v \in V.$$

Ο Κώδικας 1 αντιστοιχεί στην έκφραση της εξίσωσης στη σημειογραφία του FEniCS:

Κώδικας 1: Ορισμός PDE σε σημειογραφία FEniCS UFL

```
1 u = TrialFunction(V)
2 v = TestFunction(V)
3
4 a = dot(grad(u), grad(v))*dx
5 L = f*v*dx
```

Ο Κώδικας 2 αντιστοιχεί στις εντολές που απαιτούνται για την επίλυση του προβλήματος:

Κώδικας 2: Επίλυση PDE σε FEniCS UFL

```
1 u = Function(V)
2 solve(a == L, u, bc)
```

2.1.4 Αξιοποίηση Λογισμικού Αιχμής στο Επίπεδο της Γραμμικής Άλγεβρας

Το FEniCS ενσωματώνει και παρέχει ενιαίο τρόπο αξιοποίησης πλήθους βιβλιοθηκών γραμμικής άλγεβρας υψηλής επίδοσης, μέσω ενός κοινού για όλες περιτυλίγματος (wrapper). Ενσωματώνεται υποστήριξη για τις βιβλιοθήκες PETSc [9], Trilinos/Epetra [14], uBLAS [1] και MTL4 [2]. Ορισμένες από τις βιβλιοθήκες μπορούν εγγενώς να εκμεταλλευτούν παράλληλα συστήματα (PETSc, Epetra).

Ο χρήστης μπορεί εύκολα να εναλλαχθεί μεταξύ βιβλιοθηκών γραμμικής άλγεβρας στο χαμηλότερο επίπεδο, αλλάζοντας απλά την τιμή μιας παραμέτρου στον κώδικά του στο επίπεδο του Dolphin.

2.1.5 Εγκατάσταση FEniCS σε Cluster & Cloud

Αφού ολοκληρώθηκε η εγκατάσταση της πλατφόρμας FEniCS, έγινε δοκιμαστική λειτουργία και έτρεξαν ενδεικτικά προβλήματα, που έχει έτοιμα η πλατφόρμα, αφενός να γίνει εξοικείωση με τη χρήση της και αφετέρου να εξεταστούν οι δυνατότητες και ελλείψεις της. Πραγματοποιήθηκε η εγκατάσταση εξολοκλήρου της πλατφόρμας σε cluster περιβάλλον (στο Τμήμα Τηλεπικοινωνιακών Συστημάτων και Δικτύων του ΤΕΙ Δ. Ελλάδας), και σε μικρής κλίμακας υποδομή cloud (στο Εργαστήριο Αναγνώρισης Προτύπων του Τμήματος Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής του Πανεπιστημίου Πατρών).

Το cluster, στο οποίο εγκαταστάθηκε η FEniCS, αποτελείται από 14 blades, από τα οποία χρησιμοποιήθηκαν τα 2. Τα blades μεταξύ τους επικοινωνούν με infiniband δίκτυο. Το κάθε blade έχει 4 πυρήνες (Intel(R) Xeon(R) CPU E5530 @ 2.40GHz), 6GB RAM, και 256GB σκληρό δίσκο. Το λειτουργικό το οποίο τρέχει πάνω στο cluster είναι το SL5 (Scientific Linux 5) το οποίο είναι ουσιαστικά κλώνος του RH6. Η εγκατάσταση έγινε σε VM του Ubuntu Server και από εκεί και πέρα συνεχίστηκε η παραμετροποίηση και εγκατάσταση της FEniCS.

Ακολούθησε η εγκατάσταση της FEniCS στην cloud υποδομή του Εργαστηρίου Αναγνώρισης Προτύπων, του Τμήματος Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής του Πανεπιστημίου Πατρών το οποίο αποτελείται από Intel(R) Xeon(R) CPU E3-1220@3.10GHz με 4 cores και 16GB RAM. Το λειτουργικό το οποίο τρέχει στους κόμβους του cloud είναι το Ubuntu Server Edition. Η διαδικασία η οποία και χρησιμοποιήθηκε για την δημιουργία των cloud VM αναφέρεται στις εργασίες [19, 31].

2.2 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 [10].

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 διαφορετικών 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.

2.2.1 Μέθοδος Schwarz για Προβλήματα με Επικαλυπτόμενα Υποχωρία

Υλοποιήθηκε η επαναληπτική μέθοδος additive Schwarz [11, 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 κατανεμημένη 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 εκτέ-

λεση 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.

2.2.2 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 [40] 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 [33] 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.

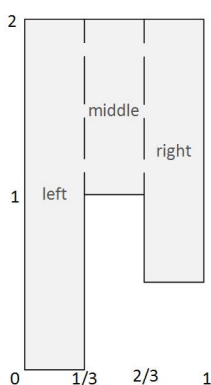
Στην παράγραφο 3.1.2 παρουσιάζεται η χρήση της υποδομής υποστήριξης υβριδικής αιτιοκρατικής/στοχαστικής μεθόδου στο FEniCS για την επίλυση υποπροβλήματος PDE με αρχική στοχαστική εκτίμηση των τιμών στη διεπαφή.

2.2.3 Μέθοδος Geometric Contraction (GEO)

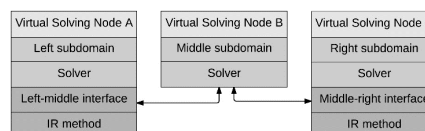
Ακολουθιακή Υλοποίηση της GEO Η μέθοδος geometric (GEO) contraction [39] υλοποιείται σαν ένα πρόγραμμα FEniCS γραμμένο στη γλώσσα UML. Αξιοποιείται η υποστήριξη του Dolphin για την εισαγωγή κλάσεων χρήσιμων για τη δημιουργία των υπο-πεδίων του προβλήματος και τη δημιουργία πλεγμάτων (τριγωνικά στοιχεία) 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 παραλληλίσιμη. Στην παράλληλη υλοποίησή της κάθε κόμβος (π.χ. εικονική μηχανή σε περιβάλλον cloud) 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 4. 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 4: Αρχιτεκτονική κατανεμημένης επίλυσης για τη μέθοδο GEO σε πρόβλημα-μοντέλο με 3 υποχωρία

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

2.2.4 Μέθοδος ROB

Ακολουθιακή Υλοποίηση της ROB Η μέθοδος ROB [39] 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 παράλληλη υλοποίηση. 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.

2.3 Επέκταση του ΠΕΠ για την Αξιοποίηση Σύγχρονων Υπολογιστικών Συστημάτων (Ετερογενών, Υποδομής Cloud και Σύγχρονων Μέσων Αποθήκευσης)

2.3.1 Αξιοποίηση Ετερογενών Συστημάτων και Σύγχρονων Μέσων Αποθήκευσης

Στόχος του έργου MATENVMED είναι η επέκταση του FEniCS για την αξιοποίηση επιλυτών υλοποιημένων σε επιταχυντές, επιλυτών που εκμεταλλεύονται μέσα αποθήκευσης flash, καθώς και για την υποστήριξη επιλυτών εξειδικευμένων σε συγκεκριμένες κατηγορίες προβλημάτων. Επίσης, θα πρέπει να υποστηρίζονται αντίστοιχες υλοποιήσεις των αλγόριθμων χαλάρωσης στις διεπαφές (για προβλήματα MDMP) και μάλιστα με φυσικό – για την κοινότητα χρηστών του FEniCS – τρόπο. Για το σκοπό αυτό, η εσωτερική διεπαφή προγραμματισμού του FEniCS έχει επεκταθεί με γενικό και αφαιρετικό τρόπο ως προς τον τελικό χρήστη.

Συγκεκριμένα, η σχεδίαση ορίζει μία νέα πρότυπη υλοποίηση (Whale) της υψηλού επιπέδου διεπαφής γραμμικής άλγεβρας, η οποία έχει τη δυνατότητα να συνδυάζει διαφορετικές υποστηριζόμενες βιβλιοθήκες γραμμικής άλγεβρας (ενδεχομένως με μετατροπή της μορφής δεδομένων από την μία στην άλλη), καθώς επίσης να ενσωματώνει νέους πειραματικούς επιλυτές στο ήδη υπάρχον μαθηματικό περιβάλλον.

Η συγκεκριμένη προσέγγιση προσφέρει λύση στην ανάγκη για επαλήθευση νέων μεθόδων και επιλυτών επάνω σε μια πλήρη, αποδοτική και παραμετροποιήσιμη πλατφόρμα όπως η πλατφόρμα FEniCS. Με αυτόν τον τρόπο μειώνεται ο χρόνος επαλήθευσης μιας νέας μεθόδου, γνωρίζοντας ότι τα υπόλοιπα συστατικά στοιχεία λειτουργούν και έχουν επαληθευθεί ανεξάρτητα. Επίσης η συγκεκριμένη προσέγγιση προσφέρει τη δυνατότητα εκτέλεσης μεθόδων και επιλυτών σε διαφορετικές μονάδες επεξεργασίας και σε επιταχυντές σε ένα ετερογενές σύστημα, εκμεταλλευόμενη το στοιχείο της παράλληλης επεξεργασίας όπου αυτό είναι δυνατόν, μειώνοντας επιπλέον τον χρόνο υπολογισμού για πολύ μεγάλα προβλήματα. Αντίστοιχα, επιτρέπει την εκτέλεση επιλυτών που αξιοποιούν τυχόν υπάρχουσες γρήγορες μονάδες μόνιμης αποθήκευσης.

Η προσθήκη Whale, πέραν των πλεονεκτημάτων προς τον τελικό χρήστη,

επιτρέπει στον σχεδιαστή επιλυτών να επικεντρωθεί στην εσωτερική βελτίωση της εκάστοτε αναπτυσσόμενης βιβλιοθήκης γραμμικής άλγεβρας και τον ορισμό της διεπαφής με το υπόλοιπο περιβάλλον του FEniCS, δίχως να απαιτείται αλλαγή στα υπόλοιπα λειτουργικά μέρη που το απαρτίζουν.

Η προσθήκη *Whale* αναπτύχθηκε επάνω στον πηγαίο κώδικα της έκδοσης 1.6 του FEniCS. Η ενοποίηση γίνεται εισάγοντας τον κώδικα της προσθήκης με την τεχνική 'μπαλώματος' (patching) στον πηγαίο κώδικα.

Ο χρήστης ελέγχει την ενεργοποίηση και απενεργοποίηση της προσθήκης μέσω των επιλογών ρύθμισης που προσφέρονται. Οι ρυθμίσεις ακολουθούν τη μορφή των υπολοίπων επιμέρους συστατικών του FEniCS και είναι προσβάσιμες από το περιβάλλον *Dolphin* μέσω της διεπαφής. Η προσθήκη *Whale* μπορεί να χρησιμοποιηθεί και από τις δυο γλώσσες που υποστηρίζονται από το *Dolphin*, δηλαδή την C++ και την Python.

Οι κλάσεις της διεπαφής γραμμικής άλγεβρας του FEniCS (*GenericVector*, *GenericMatrix*, *GenericLUSolver*, *GenericKrylovSolver*, κα.) έχουν επεκταθεί για να υποστηρίξουν την προσθήκη *Whale*, η οποία υλοποιεί την συγκεκριμένη διεπαφή. Η υλοποίηση όπως είναι αναμενόμενο ακολουθεί το παράδειγμα των ήδη υποστηριζόμενων βιβλιοθηκών γραμμικής άλγεβρας (*PETSc*, *Trilinos* κα.).

Η προσθήκη *Whale* είναι απενεργοποιημένη από προεπιλογή και ο χρήστης πρέπει να την ενεργοποιήσει ρητά μέσω κατάλληλης επιλογής ρύθμισης. Για παράδειγμα, χρησιμοποιώντας τη διεπαφή Python, ο τελικός χρήστης μπορεί εύκολα να ενεργοποιήσει την προσθήκη *Whale* σε οποιοδήποτε υπάρχον αρχείο πηγαίου κώδικα, απλά προσθέτοντας την παρακάτω εντολή ενεργοποίησης (Κώδικας 3):

Κώδικας 3: Επιλογή ρύθμισης για την ενεργοποίηση της προσθήκης *Whale* μέσω της διεπαφής Python.

```
from dolfin import *

parameters["use_whale_backend"] = True

# rest of the code
```

Με την ενεργοποίηση, η προσθήκη χρησιμοποιεί την προσαρμοσμένη βιβλιοθήκη γραμμικής άλγεβρας που έχει ενσωματώσει ο σχεδιαστής, για την επίλυση των γραμμικών συστημάτων που προκύπτουν από τον ορισμό του προβλήματος. Εάν δεν υπάρχει διαθέσιμη μία τέτοια προσαρμοσμένη βιβλιοθήκη, η προσθήκη χρησιμοποιεί την προεπιλεγμένη βιβλιοθήκη του FEniCS, έτσι ώστε ο τελικός χρήστης να είναι σε θέση να χρησιμοποιήσει το περιβάλλον με τον συνηθισμένο τρόπο, παρακάμπτοντας εξ ολοκλήρου τον μηχανισμό πρόσβασης σε προσαρμοσμένες βιβλιοθήκες γραμμικής άλγεβρας που προσφέρεται.

2.4 Χρήση Σύγχρονου Υλικού για την Επίλυση Μεγάλων Αραιών Γραμμικών Συστημάτων

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.

2.4.1 Τεχνολογίες Δευτερεύουσας Μνήμης και Αλγόριθμοι Out-of-Core

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 αλγόριθμους.

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

Μελετήσαμε την επίλυση αραιών συστημάτων με τα πακέτα: MUMPS [4] [5], την έκδοση του PARDISO [29] [30] που περιλαμβάνεται στην INTEL MKL και το WSMP [8] 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 3.2.1.

2.4.2 Σύγχρονες Πλατφόρμες Υλικού και Αραιά Γραμμικά Συστήματα

Η επίλυση αραιών συστημάτων μπορεί να πραγματοποιηθεί είτε με άμεσες είτε με επαναληπτικές μεθόδους. Οι άμεσες μέθοδοι εγγυώνται την επίλυση του συστήματος, εφόσον αυτό έχει λύση, αλλά δεν κλιμακώνουν εύκολα καθώς έχουν σημαντικές απαιτήσεις σε κύρια μνήμη. Από την άλλη, οι επαναληπτικές μέθοδοι

είναι ταχύτερες, δεν έχουν μεγάλες απαιτήσεις σε μνήμη και παραλληλοποιούνται εύκολα, αλλά δεν εγγυώνται πάντοτε σύγκλιση.

Η απόδοση κάθε μεθόδου επίλυσης είναι συνυφασμένη με την υποκείμενη υπολογιστική πλατφόρμα. Έτσι, η ανάπτυξη νέων αλγορίθμων, που εκμεταλλεύονται τα πλεονεκτήματα της διαθέσιμης κάθε φορά αρχιτεκτονικής υλικού, έχει σημαντικό ερευνητικό ενδιαφέρον. Με βάση τις παρατηρήσεις αυτές, πραγματοποιήθηκε μια συγκριτική μελέτη άμεσων και έμμεσων μεθόδων για την επίλυση αραιών γραμμικών συστημάτων σε διαφορετικές, σύγχρονες, πλατφόρμες υλικού. Σκοπό της μελέτης αποτέλεσε η ανίχνευση των ορίων και των δυνατοτήτων κάθε μεθόδου και τα αποτελέσματα παρουσιάζονται στην παράγραφο 3.2.2.

2.4.3 Αλγόριθμοι Εξωτερικής Μνήμης για Μνήμες Τύπου Flash

Οι αλγόριθμοι εξωτερικής μνήμης που αφορούν την διαχείριση δεδομένων σε μνήμες flash έχουν αποτελέσει αντικείμενο πρόσφατης έρευνας στο πεδίο των Βάσεων Δεδομένων και κυρίως στις δομές ευρετηρίου (B+tree, R-tree, κτλ). Η έρευνα έχει καταδείξει ότι απαιτείται η περαιτέρω εξέλιξή τους, ώστε να εκμεταλλεύονται αποδοτικότερα τις δυνατότητες της flash. Μερικές από τις τεχνικές που έχουν προταθεί είναι: α) η μετατροπή των «ακριβών» εγγραφών σε τυχαίες θέσεις (random writes), σε εγγραφές σε συνεχόμενες θέσεις (sequential writes) με χρήση log records, β) η αξιοποίηση του «εσωτερικού» παραλληλισμού των δίσκων SSD, με την αύξηση του μεγέθους της εισόδου/εξόδου (I/O) και του αριθμού των εξερχόμενων αιτημάτων ανάγνωσης/εγγραφής (outstanding I/O requests).

Μελετήσαμε την υλοποίηση της μεθόδου Cholesky στην βιβλιοθήκη TAUCS [38, 28] και αναζητήσαμε τεχνικές για την αποδοτικότερη υλοποίησή της για δίσκους στερεάς κατάστασης (flash SSDs). Διερευνήσαμε την χρήση ενδιάμεσων buffers για την πραγματοποίηση μαζικών (batch) εγγραφών και αναγνώσεων των στοιχείων πλήρωσης και των βοηθητικών δομών (π.χ. δέντρο απαλοιφής), καθώς και τις δυνατότητες προ-ανάκτησής τους (pre-fetching) στα διάφορα στάδια της παραγοντοποίησης (factorization) και της επίλυσης (forward elimination, backward substitution). Τα αποτελέσματα της μελέτης παρουσιάζονται στην παράγραφο 3.2.3.

2.4.4 Αξιοποίηση Υποδομής Cloud - Web Services

Το έργο MATENVMED οραματίστηκε ένα ριζικά διαφορετικό τρόπο υλοποίησης, δημοσίευσης, διαμοίρασης, αναζήτησης και επαναχρησιμοποίησης πόρων σχετικών με τον επιστημονικό υπολογισμό στην καθημερινή πρακτική. Για το λόγο αυτό είναι αναγκαίο ένα ανοιχτό, ισορροπημένο και διαρκώς εξελισσόμενο οικοσύστημα υπηρεσιών ιστού (web services). Στα πλαίσια του έργου διερευνήσαμε την ιδέα της ανάπτυξης, προσφοράς και χρήσης στα πλαίσια της πλατ-

φόρμας του έργου υπολογιστικών πακέτων σχετικών με προβλήματα MDMP με διάφανο και αφηρημένο τρόπο, μέσω του παγκόσμιου ιστού. Για το σκοπό αυτό επεκτείνουμε την πλατφόρμα με ένα επίπεδο που αξιοποιεί web services και υπακούει σε πρότυπα βασισμένα στην XML [26]:

SOAP: (Simple Object Access Protocol) για την ανταλλαγή δομημένης πληροφορίας μεταξύ web services πάνω από το δίκτυο. Βασίζεται σε άλλα πρωτόκολλα επιπέδου εφαρμογής, όπως το HTTP για τη διαπραγμάτευση και τη μετάδοση μηνυμάτων.

WSDL: (Web Services Description Language) που χρησιμοποιείται για την περιγραφή της λειτουργικότητας που προσφέρει ένα web service, πως μπορεί να κληθεί, τις παραμέτρους που αναμένει και τις δομές δεδομένων που επιστρέφει.

ebXML: (e-business XML) που επιτρέπει σε παρόχους web services να δημοσιεύσουν τις υπηρεσίες τους και σε καταναλωτές να υποβάλλουν ερωτήματα σχετικά με τη διαθεσιμότητα και την περιγραφή των services.

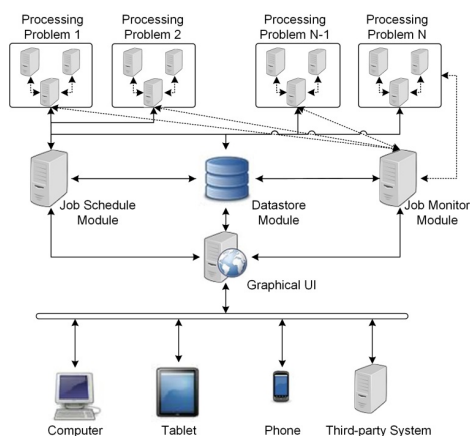
Ο βασικός τρόπος για την ανάπτυξη ενός επιτυχή MDMP με web services είναι η ακόλουθη. Αρχικά υλοποιούμε τα τμήματα λογισμικού που περιγράφηκαν στην παράγραφο 2.2, η οποιονδήποτε άλλο κώδικα όπως για παράδειγμα επιλυτές γραμμικής άλγεβρας, ως web services και τα δημοσιοποιούμε στον κατάλογο ebXML που διατηρούμε, χρησιμοποιώντας ένα από τα διαθέσιμα για το σκοπό αυτό περιβάλλοντα ανάπτυξης ή πλατφόρμες. Έχουμε υλοποιήσει web services χρησιμοποιώντας WSO2, .Net, Eclipse, ή Spyne [23], ακολουθώντας ένα συστηματικό τρόπο που οδηγεί σε μια διαδικασία ρουτίνας χωρίς ιδιαίτερες προκλήσεις. Ακολούθως, οποιοσδήποτε προγραμματιστής ή ακόμα και λογισμικό μπορεί να υποβάλλει ερωτήματα στο ebXML για συγκεκριμένα services, να λάβει τη λίστα των διαθεσίμων, να επιλέξει το καταλληλότερο και να συνδεθεί μαζί του αξιοποιώντας την πληροφορία στο αντίστοιχο αρχείο WSDL. Αυτή η διαδικασία είναι δυνατό να πραγματοποιηθεί ακόμα και κατά το χρόνο εκτέλεσης.

2.4.5 Αξιοποίηση Υποδομής Cloud - Πλατφόρμα IRaaS

Σε αυτή την παράγραφο περιγράφεται μία διαφορετική φιλοσοφία σε σχέση με την παράγραφο 3.3.1. Πιο συγκεκριμένα, περιγράφουμε τη σχεδίαση και υλοποίηση μιας προσαρμοσμένης αρχιτεκτονικής cloud για επίλυση προβλημάτων πολλαπλών πεδίων / πολλαπλών φυσικών που υλοποιεί τη μεθοδολογία χαλάρωσης στη διεπαφή. Ο χρήστης δίνει ένα πρόβλημα σαν είσοδο, επιλέγει την IR μέθοδο που ταιριάζει καλύτερα με το συγκεκριμένο πρόβλημα και έχει πρόσβαση στη λύση σε έναν κοντά στο βέλτιστο χρόνο εκτέλεσης, από οποιοδήποτε μέρος

και οποιαδήποτε συσκευή. Συγχρόνως, η εφαρμογή δεσμεύει τους ελάχιστους δυνατούς πόρους αυτόματα, στο παρασκήνιο, χωρίς παρέμβαση του χρήστη.

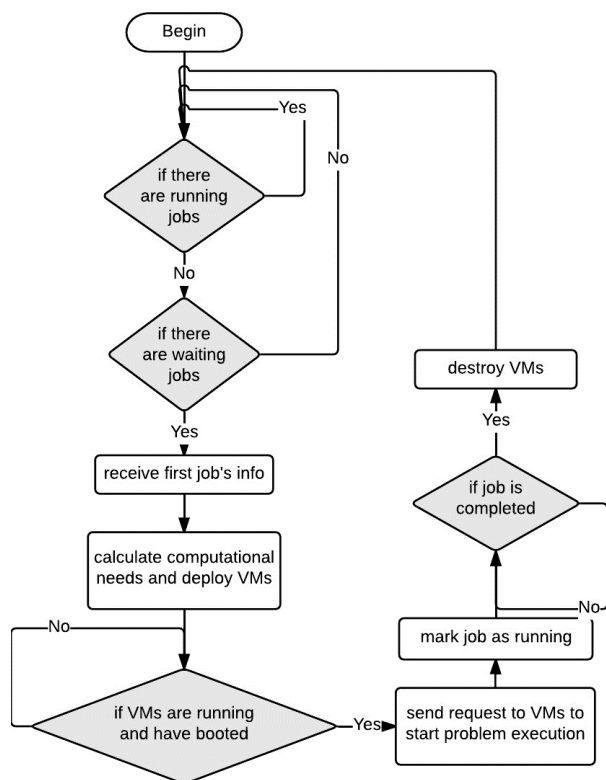
Για την επίτευξη των παραπάνω στόχων, εγκαταστάθηκε στις υποδομές του εργαστηρίου Αναγνώρισης Προτύπων μια συγκεκριμένη πλατφόρμα λογισμικού για την παροχή υπηρεσιών Cloud, το CloudStack. Το CloudStack αποτελεί ένα εργαλείο που ελέγχει τη δεξαμενή των υπολογιστικών πόρων καθώς επίσης διαχειρίζεται το δίκτυο και τον αποθηκευτικό χώρο. Η αρχιτεκτονική του συστήματος είναι βασισμένη στο μοντέλο IaaS και SaaS της τεχνολογίας Cloud και ταυτόχρονα στη διαχείριση της κατανομής των υπολογιστικών πόρων του Cloud έτσι ώστε να δημιουργηθούν ομάδες εικονικών μηχανών για παράλληλη και κατανεμημένη επεξεργασία. Υποστηρίζει ένα περιβάλλον για την επίλυση διαφόρων ειδών προβλημάτων και ανάλυσης δεδομένων, παρέχοντας μία απλή και φιλική προς τον χρήστη διεπαφή, αυτόματη διανομή του προβλήματος στους κατάλληλους υπολογιστικούς πόρους, τον απαραίτητο αποθηκευτικό χώρο για τα δεδομένα εισόδου και των αποτελεσμάτων ενός προβλήματος καθώς και την απαιτούμενη επικοινωνία μεταξύ των συστατικών στοιχείων του συστήματος.



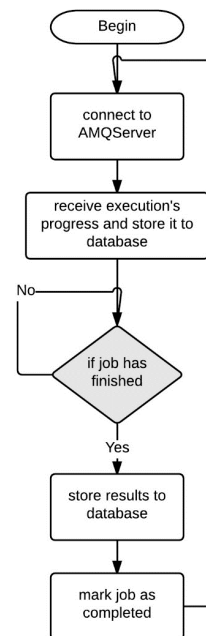
Εικόνα 5: Αρχιτεκτονική IRaaS

Το Σχήμα 5 παρουσιάζει μια επισκόπηση της αρχιτεκτονικής του συστήματος με όλα τα συστατικά στοιχεία του συστήματος και τις αλληλεπιδράσεις τους. Επίσης απεικονίζονται τρεις διαφορετικοί τύποι προβλημάτων έτσι ώστε να γίνει αντιληπτή η δυνατότητα του συστήματος να επιλύει διαφορετικού τύπου προβλήματα. Είναι σημαντικό επίσης να αναφερθεί ότι τόσο η επικοινωνία μεταξύ του Task Monitor με τις διάφορες εικονικές μηχανές κάθε προβλήματος, όσο και η επικοινωνία που έχουν οι εικονικές μηχανές κάθε προβλήματος μεταξύ τους γίνεται μέσω του Advanced Message Queuing Protocol (AMQP) για την ανταλλαγή των μηνυμάτων.

Τα σημαντικότερα συστατικά στοιχεία της αρχιτεκτονικής είναι τα εξής: Η γρα-



(α') Διάγραμμα ροής workflow manager



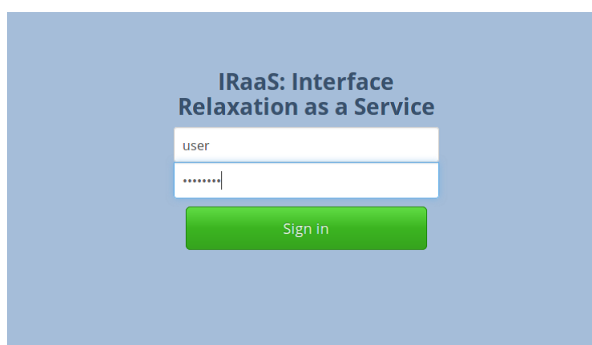
(β') Διάγραμμα ροής task monitor

Εικόνα 6: Διαγράμματα ροής workflow manager και task monitor.

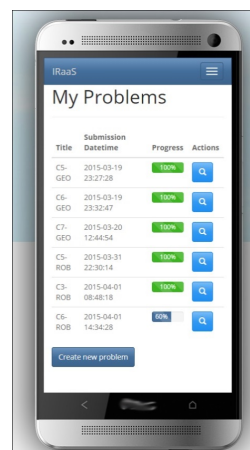
φική διεπαφή χρήστη (User Portal), όπου κάθε χρήστης έχει τη δυνατότητα εγγραφής ή σύνδεσης στο λογαριασμό του. Ο Task Monitor εμπεριέχει έναν εξυπηρετητή Advanced Message Queuing Protocol (AMQP) που χειρίζεται την επικοινωνία μεταξύ των οντοτήτων του συστήματος. Ο εξυπηρετητής AMQP είναι βασισμένος στο Message Queuing Protocol (AMQP) το οποίο συνδέει συστήματα και διαχειρίζεται την ανταλλαγή πληροφοριών και μηνυμάτων μεταξύ αυτών. Ο Workflow Manager είναι υπεύθυνος για την ενορχήστρωση και την εκτέλεση ολόκληρης της διαδικασίας στην υποδομή Cloud. Το Datastore ουσιαστικά είναι ένα αποθετήριο όπου αποθηκεύονται όλες οι πληροφορίες όπως τα δεδομένα εισόδου, τα αποτελέσματα και οι ενδιαμέσες πληροφορίες για το πρόβλημα ενός χρήστη. Σε αυτό έχουν πρόσβαση όλα τα στοιχεία του συστήματος και είναι ουσιαστικά η βάση δεδομένων του συστήματος. Τέλος, οι ομάδες εικονικών μηχανών είναι υπεύθυνες για την εκτέλεση της εκάστοτε διαδικασίας επίλυσης ενός προβλήματος.

Ο Workflow Manager παρέχει μια σειρά από λειτουργίες που φαίνονται αναλυτικά στο Σχήμα 6(α').

Για την απαραίτητη ανταλλαγή μηνυμάτων μεταξύ τόσο του Task Monitor με τις εικονικές μηχανές όσο και αυτών μεταξύ τους, χρησιμοποιήθηκε το RabbitMQ. Το RabbitMQ είναι ένας ελαφρύς, αξιόπιστος, επεκτάσιμος και φορητός μεσο-λαβητής μηνυμάτων. Ο Task Monitor παρέχει μια σειρά από λειτουργίες που φαίνονται αναλυτικά στο Σχήμα 6(β').



(α') Σελίδα sign in IRaaS



(β') Σελίδα IRaaS (από κινητό) με τα προβλήματα του χρήστη

Εικόνα 7: Σελίδες authentication & προβλημάτων IRaaS.

Στην Εικόνα 7(α') φαίνεται η οθόνη μέσω της οποίας εισάγεται στο σύστημα ο χρήστης. Αξίζει να σημειωθεί ότι το όνομα που δόθηκε στην υποδομή είναι: "IRaaS: Interface Relaxation as a Service" διότι παρέχεται ένα SaaS μοντέλο για την επίλυση προβλημάτων πολλαπλών πεδίων / πολλαπλών φυσικών με τη χρήση μεθόδων χαλάρωσης στις διεπαφές (Interface Relaxation Methods).

Μετά το authentication ο χρήστης μπορεί να δει μια λίστα που περιλαμβάνει όλα τα προβλήματα που έχει καταθέσει, τις λεπτομέρειες αυτών καθώς και να θέσει προς εκτέλεση ένα νέο πρόβλημα. Η σχετική οθόνη, όπως φαίνεται όταν χρησιμοποιείται από κινητό, απεικονίζεται στην Εικόνα 7(β'). Στην πρώτη στήλη της οθόνης ("Title") φαίνεται το όνομα του προβλήματος, στην δεύτερη στήλη ("Submission Datetime") η ημερομηνία και ώρα κατάθεσής του, στην τρίτη στήλη ("Progress") η πρόοδος του προβλήματος δηλαδή ποιο είναι το ποσοστό ολοκλήρωσής του και στην τέταρτη στήλη ("Actions") υπάρχει ένα κουμπί που μόλις πατηθεί ο χρήστης μεταβαίνει σε μία νέα οθόνη που υπάρχουν όλες οι πληροφορίες του προβλήματος καθώς και τα αρχεία αποτελεσμάτων αν έχει ολοκλη-

ρωθεί η εκτέλεσή του. Στο κάτω μέρος της οθόνης υπάρχει ένα κουμπί με όνομα “Create new problem” το οποίο, μόλις πατηθεί, μεταβιβάζει το χρήστη σε μία νέα οθόνη, που του παρέχει μια φόρμα ώστε να καταθέσει όλα τα απαραίτητα δεδομένα εισόδου του νέου προς εκτέλεση προβλήματος.

Domain	X	Y
A	7	40
B	7	10
C	7	20

Εικόνα 8: Σελίδα εισαγωγής νέου προβλήματος IRaaS

Στην Εικόνα 8 παρουσιάζεται η οθόνη της εφαρμογής που εμφανίζεται μόλις ο χρήστης πατήσει το κουμπί “Create new problem”. Για την ακρίβεια εμφανίζεται μια φόρμα στην οποία ο χρήστης πρέπει να συμπληρώσει τα στοιχεία του νέου προβλήματος που θέλει να εκτελέσει. Στη φόρμα αυτή δίνει ένα όνομα και μια περιγραφή του νέου προβλήματος, τις τιμές των τριών subdomains καθώς και διάφορες περαιτέρω πληροφορίες που είναι απαραίτητες για την επίλυση του προβλήματος. Επίσης στη φόρμα αυτή επιλέγει από μια drop-down λίστα την επιθυμητή μέθοδο χαλάρωσης με την οποία θέλει να γίνει η ανάλυση (GEO ή ROB) και μόλις συμπληρωθούν όλα τα κελιά, για να αποσταλούν στο σύστημα όλα τα δεδομένα που εισήγαγε και να ξεκινήσει η εκτέλεση, αρκεί να πατήσει το κουμπί “Start solution process” που βρίσκεται το κάτω μέρος της οθόνης. Μόλις αυτό το κουμπί πατηθεί, επιστρέφεται ένα μήνυμα επιτυχίας για την σωστή καταχώρηση των δεδομένων και ο χρήστης μεταβαίνει αυτόματα στην προηγούμενη οθόνη προκειμένου να παρακολουθήσει την πρόοδο του νέου προβλήματος. Αν

δεν έχει γίνει σωστή καταχώρηση των δεδομένων, τότε στην ίδια φόρμα υποδεικνύεται ποιο πεδίο περιέχει λανθασμένη τιμή ή είναι κενό.



Εικόνα 9: Σελίδα προβλήματος IRaaS

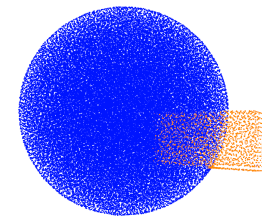
Εάν ένας χρήστης πατήσει το κουμπί “Action”, τότε ο χρήστης μεταβαίνει αυτόματα σε μία νέα οθόνη (Εικόνα 9) όπου υπάρχουν όλες οι πληροφορίες του προβλήματος καθώς και τα αρχεία αποτελεσμάτων αν έχει ολοκληρωθεί η εκτέλεσή του. Η συγκεκριμένη οθόνη περιέχει όλα τα δεδομένα εισόδου του προβλήματος, την κατάστασή του (αν έχει ξεκινήσει να εκτελείται ή ποιο είναι το ποσοστό που έχει πραγματοποιηθεί από τη συνολική διαδικασία επίλυσης ή αν έχει ολοκληρωθεί), και αν το πρόβλημα έχει ολοκληρωθεί, περιέχει επίσης την ημέρα και ώρα ολοκλήρωσης του και, πατώντας το κουμπί της στήλης “Download” κάτω δεξιά, τα αρχεία με τα τελικά αποτελέσματα του προβλήματος.

Τέλος, εάν ο χρήστης επιθυμεί να εξέλθει από την εφαρμογή, αρκεί να πατήσει το κουμπί με την ένδειξη “Logout” που υπάρχει στο βασικό μενού, στο πάνω μέρος της κάθε οθόνης της εφαρμογής.

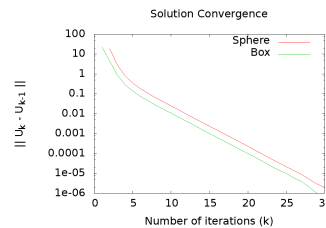
3 Πειραματικά Αποτελέσματα

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

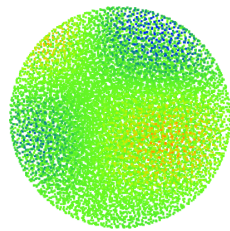
3.1.1 Μέθοδος Schwarz για Προβλήματα με Επικαλυπτόμενα Υποχωρία



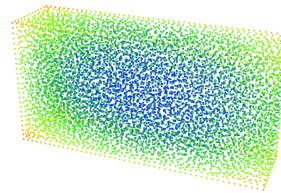
(α') Τοπολογία υποχωρίων



(β') Ρυθμός σύγκλισης για τα δύο υποπροβλήματα



(γ') Η λύση για το υποχωρίο σφαίρα



(δ') Η λύση για το υποχωρίο παραλληλεπίπεδο

Εικόνα 10: Ένα μοντέλο 3-διάστατο πρόβλημα με μερικώς επικαλυπτόμενα 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 10(α'). Οι Εικόνες 10(γ') και 10(δ') απεικονίζουν τη λύση για τα δύο υποχωρία με χρήση του επαναληπτικού επιλυτή Schwarz που υλοποιήθηκε στο FEniCS. Στην Εικόνα 10(β') παρουσιάζεται ο ρυθμός σύγκλισης για τα 2 υποπροβλήματα, έως την επίτευξη της επιθυμητής από το χρήστη ακρίβειας.

Κώδικας 4: 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 ]

```

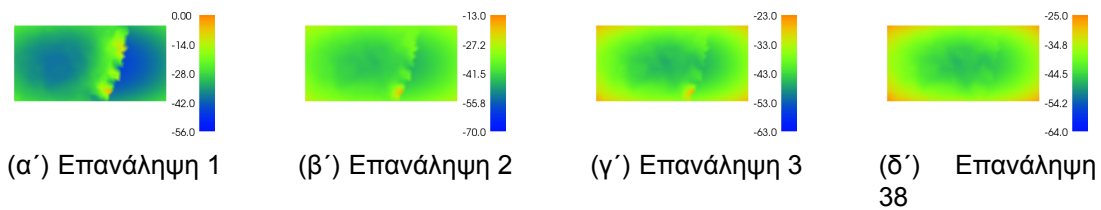
Ο Κώδικας 4 αντιστοιχεί στον ορισμό του παραλληλεπίπεδου υποχωρίου (box3D_1.py), ξεκινώντας από έναν κοινό (ανεξάρτητο του υποχωρίου) σκελετό. Παρόμοιος είναι και ο ορισμός του υποχωρίου που αντιστοιχεί στη σφαίρα (sphere3D_1.py).

Κώδικας 5: Επίλυση 3D προβλήματος με 2 μερικώς αλληλεπικαλυπτόμενα υποχωρία.

```

1 from dolfin import *
2 import solverconfig
3 import solver
4
5 # client = hmc.RemoteClient(wsdI_url)
6 # client.set_options(timeout=timeout_in_seconds)
7 client = hmc.LocalClient()
8
9 import sphere3D_1 as sphere

```



Εικόνα 11: Στιγμιότυπα οριζόντιας τομής του παραλληλεπιπέδου σε διαφορετικές επαναλήψεις του επιλυτή

```

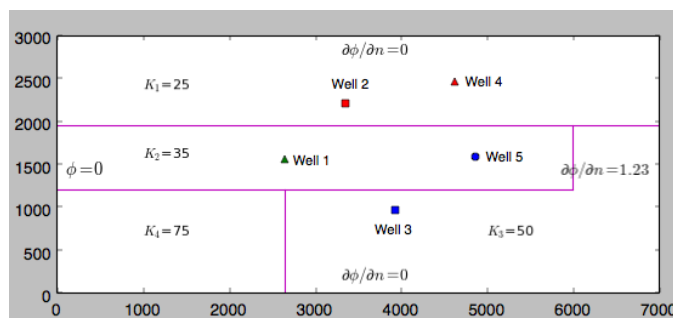
10 import box3D_1 as box
11
12 sp = sphere.Problem(client=client)
13 bp = box.Problem(client=client)
14 subdomains=[ sp, bp ]
15
16 config = solverconfig.Config3D()
17 solver.solve(subdomains=subdomains, config=config)

```

Ο Κώδικας 5 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. Ο αναγνώστης παρατηρήσει επίσης ότι η χρήση ενός απομακρυσμένου επιλυτή είναι εξαιρετικά απλή, αντικαθιστώντας τη γραμμή 7 με τις σχολιασμένες γραμμές 5-6.

Στην Εικόνα 11 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.

Χρήση σε 2D πρόβλημα περιβαλλοντικής μηχανικής Ας θεωρήσουμε την PDE η οποία περιγράφει τη σταθερή κατάσταση του προβλήματος υφαλμύρισης σε παράκτιους υδατοφορείς. Η εξίσωση πρέπει να λυθεί σε κάθε επαναληπτικό βήμα ενός στοχαστικού αλγορίθμου βελτιστοποίησης [36, 35, 34], ο οποίος αξιοποιείται ώστε να ελέγξει με βέλτιστο τρόπο την άντληση από όλες της ενεργές πηγές άντλησης (πηγάδια) ενός παράκτιου υδατοφορέα και να τις προστατεύσει από την υφαλμύριση. Η λύση του (δυναμικό ροής) χρησιμοποιείται ώστε να εντοπιστεί το όριο μεταξύ αλμυρού και γλυκού νερού. Ο υδατοφορέας που απεικονίζεται στην Εικόνα 12 προσωμοιώνει πραγματικό παράκτιο υδατοφορέα στο Βαθύ της Καλύμνου [24].



Εικόνα 12: Υδατοφορέας Καλύμνου.

Πιο συγκεκριμένα, ας θεωρήσουμε την ελλειπτική PDE

$$\frac{\partial}{\partial x} \left(K \frac{\partial \phi}{\partial x} \right) + \frac{\partial}{\partial y} \left(K \frac{\partial \phi}{\partial y} \right) + N - Q = 0, \quad (x, y) \in \mathcal{R}, \quad (1)$$

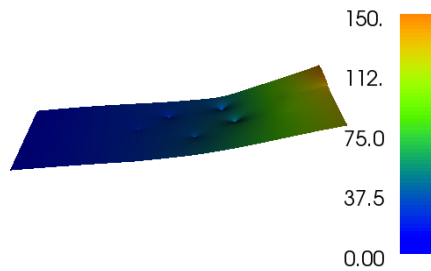
όπου ϕ (m^2) αντιστοιχεί στο δυναμικό ροής του Struck, N (m/day) δηλώνει τη συνολική ημερήσια επαναφόρτιση του υδατοφορέα, ομοιόμορφα κατανεμημένη στην επιφάνειά του, K (m/day) αντιστοιχεί στην υδραυλική διαπερατότητα και Q (m/day) στην συνολική ημερήσια αποφόρτιση του υδατοφορέα. Ο παραλληλόγραμμος υδατοφορέας $\mathcal{R}$ εκτείνεται σε επιφάνεια 7×3 Km, είναι ετερογενής ως προς την υδραυλική του διαπερατότητα και περιλαμβάνει M πηγάδια w_i ($i = 1, \dots, M$) τα οποία αντλούν με ρυθμούς Q_i (m^3/day). Στο πρόβλημά μας οι παραπάνω φυσικές παράμετροι τίθενται στις τιμές $N = 0.03$ $m/year$, $M = 5$ and $Q_1 = 252$ m^3/day , $Q_2 = 450$ m^3/day , $Q_3 = 749$ m^3/day , $Q_4 = 1045$ m^3/day and $Q_5 = 1270$ m^3/day , ενώ η υδραυλική διαπερατότητα K λαμβάνει τιμές $K_1 = 25$ m/day , $K_2 = 35$ m/day , $K_3 = 50$ m/day and $K_4 = 75$ m/day στις 4 υποπεριοχές του $\mathcal{R}$ (Εικόνα 12). Επιπλέον, ο συνολικός ρυθμός αποφόρτισης Q λαμβάνει την τιμή

$$Q = \sum_{i=1}^5 \tilde{Q}_i \delta(x - x_i, y - y_i)$$

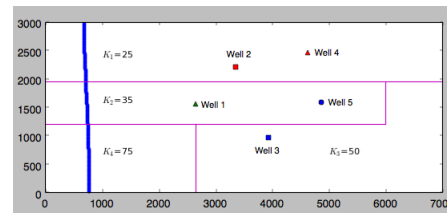
όπου $\tilde{Q}_i$ αντιστοιχεί στο ρυθμό άντλησης Q_i κανονικοποιημένο σε σχέση με κάποια στοιχειώδη επιφάνεια και $\delta(x - x_i, y - y_i)$ δηλώνει τη συνάρτηση Delta. Τέλος, θεωρούμε συνοριακή συνθήκη Dirichlet ($\phi(0, y) = 0$) στο αριστερό (παράκτιο) σύνορο, ενώ σε όλα τα άλλα σύνορα τίθενται συνοριακές συνθήκες Neumann όπως φαίνεται στην Εικόνα 12.

Το δυναμικό ροής που υπολογίσθηκε για το παραπάνω πρόβλημα με χρήση της υλοποίησης του αλγορίθμου Schwarz στο FEniCS απεικονίζεται στην Εικόνα 13(α'). Ακολούθως, καθορίστηκε αλγοριθμικά το όριο μεταξύ αλμυρού και γλυκού νερού όπως φαίνεται στην Εικόνα 13(β').

Το πρόβλημα χωρίζεται φυσιολογικά σε 4 υποπροβλήματα, λόγω του διαφορετικού τελεστή PDE (διαφορετικά Q και K στην εξίσωση PDE equation 1).



(α') Υπολογισθέν δυναμικό ροής.



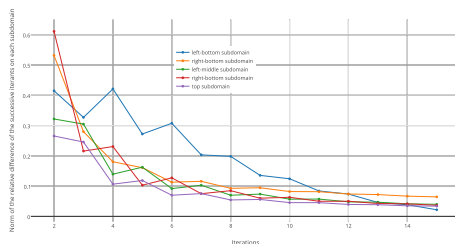
(β') Όριο μεταξύ γλυκού και αλμυρού νερού.

Εικόνα 13: Υπολογισθείσες λύσεις για τον υδατοφόρα Καλύμνου.

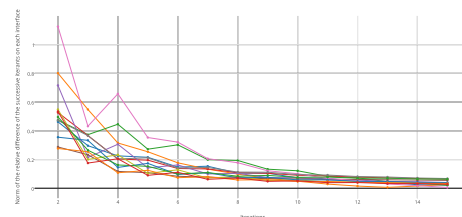
Τα υποπροβλήματα διακρίνονται στην Εικόνα 12. Χωρίζουμε επιπλέον το κάτω δεξιά χωρίο σε 2 υποχωρία για λόγους γεωμετρικής ευκολίας (ώστε να αντιμετωπίζουμε μόνο παραλληλόγραμμα υποχωρία). Έτσι, αντιμετωπίζουμε ένα σύνολο 5 υποπροβλημάτων.

Προκειμένου να είναι δυνατή η χρήση της μεθόδου Schwarz, θεωρούμε επικάλυψη του 2ου υποχωρίου (με υδραυλική διαπερατότητα K_2) με τα γειτονικά του. Η επιφάνειά του εκτείνεται οριζόντια κατά 400 μ μέσα στα γειτονικά υποχωρία. Επιπλέον, θεωρούμε επικαλυπτόμενη περιοχή του 4ου υποχωρίου μέσα στο 3ο βάθους 400 μ. Η ζώνη 1400 μ που περιλαμβάνει το πηγάδι 3 στο 2ο υποχωρίο επιλέγεται ως περιοχή επικάλυψης με το 3ο υποχωρίο. Ως αποτέλεσμα υπάρχουν 14 διεπαφές μεταξύ των 5 υποχωρίων.

Όπως φαίνεται καθαρά από τις Εικόνες 14(β') και 14(α') η μέθοδος συγκλίνει μάλλον γρήγορα, παρά την ύπαρξη 5 υποχωρίων και 14 διεπαφών μεταξύ τους.



(α') Ρυθμός σύγκλισης μεταξύ διαδοχικών επαναλήψεων για τα υποπροβλήματα.

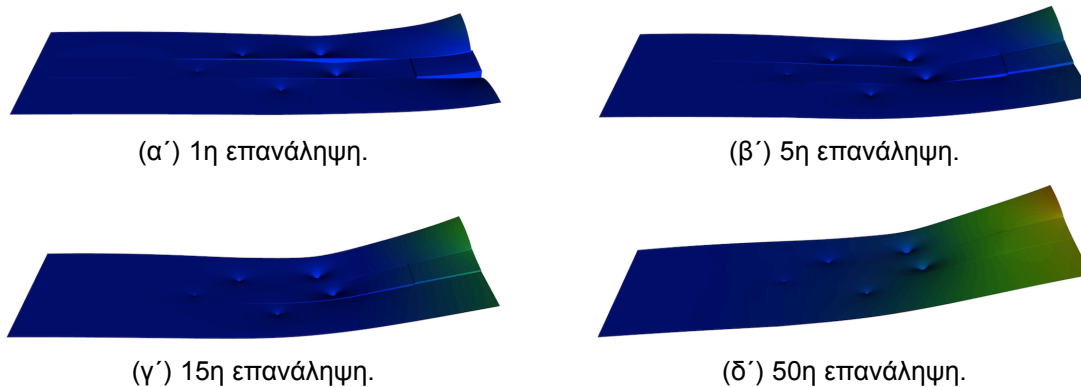


(β') Ρυθμός σύγκλισης μεταξύ διαδοχικών επαναλήψεων για τις 14 διεπαφές.

Εικόνα 14: Ρυθμός σύγκλισης μεταξύ διαδοχικών επαναλήψεων της μεθόδου.

Η φύση της σύγκλισης απεικονίζεται και στην Εικόνα 15. Οι όροι σφάλματος υψηλής συχνότητας εξαλείφονται σύντομα (κατά τη διάρκεια των πρώτων λίγων

επαναλήψεων) και η λύση εξομαλύνεται επιπλέον στις μετέπειτα επαναλήψεις.



Εικόνα 15: Η υπολογισθείσα συνολική λύση μετά από 1, 2, 15 και 50 επαναλήψεις.

3.1.2 Υβριδική Αιτιοκρατική/Στοχαστική Μέθοδος για Προβλήματα MDMP

Ο κώδικας 6 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 6: 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)

```

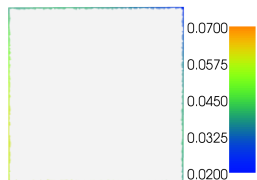
```

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)

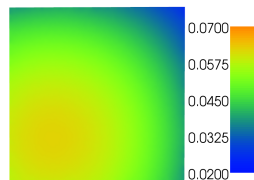
```

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

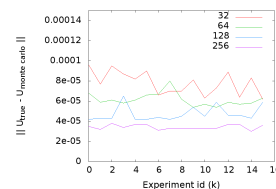
Η Εικόνα 16 περιλαμβάνει την εκτίμηση της στοχαστικής μεθόδου για τις τιμές στη διεπαφή του εσωτερικού υποχωρίου, τη λύση που επιτεύχθηκε από την υβριδική αιτιοκρατική / στοχαστική μέθοδο για το πρόβλημα-μοντέλο που περιγράφεται από τον Κώδικα 6, καθώς και το απόλυτο σφάλμα σε σχέση με έναν πλήρως αιτιοκρατικό 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 16: 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 στιγμιότυπα κώδικα από την υλοποίηση της μεθόδου GEO στο FEniCS.

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

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

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, bcsdown)
```

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 Χρήση Σύγχρονων Τεχνολογιών Υλικού για την Επίλυση Μεγάλων Αραιών Γραμμικών Συστημάτων

3.2.1 Τεχνολογίες Δευτερεύουσας Μνήμης και Αλγόριθμοι Out-of-Core

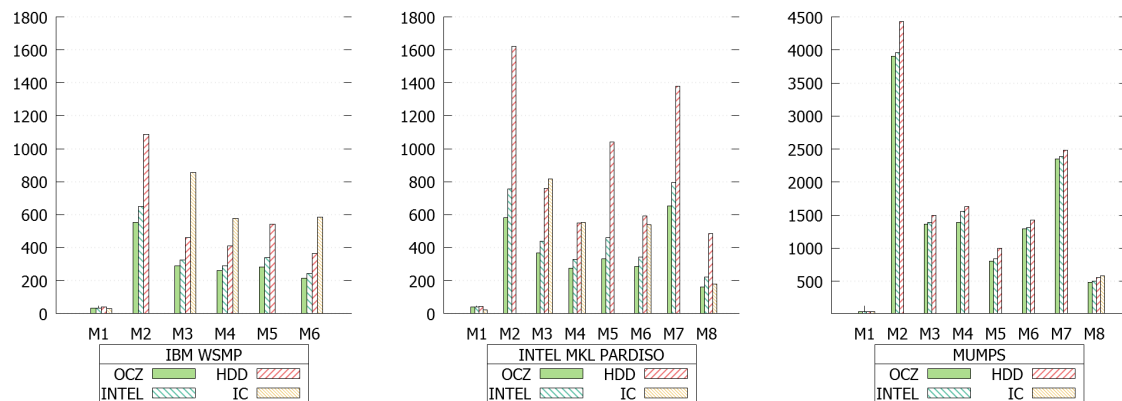
Τα πειράματα πραγματοποιήθηκαν σε δύο σταθμούς εργασίας DELL Precision T3500 με 8GB of DDR3 RAM και έναν τετρα-πύρηνο Intel Xeon W3550 3.06GHz το καθένα. Ο πρώτος σταθμός εργασίας διαθέτει έναν οδηγό SSD Intel 520 240GB, SATA-III, και έναν οδηγό OCZ Revodrive 350 PCIe 480GB ως επιπλέον αποθηκευτικό μέσο. 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 Φλόριντα² και του ερευνητικού προγράμματος GRID-TLSE³. Οι ιδιότητες 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

²<http://www.cise.ufl.edu/research/sparse/matrices/>

³<http://gridtlse.enseiht.fr/>



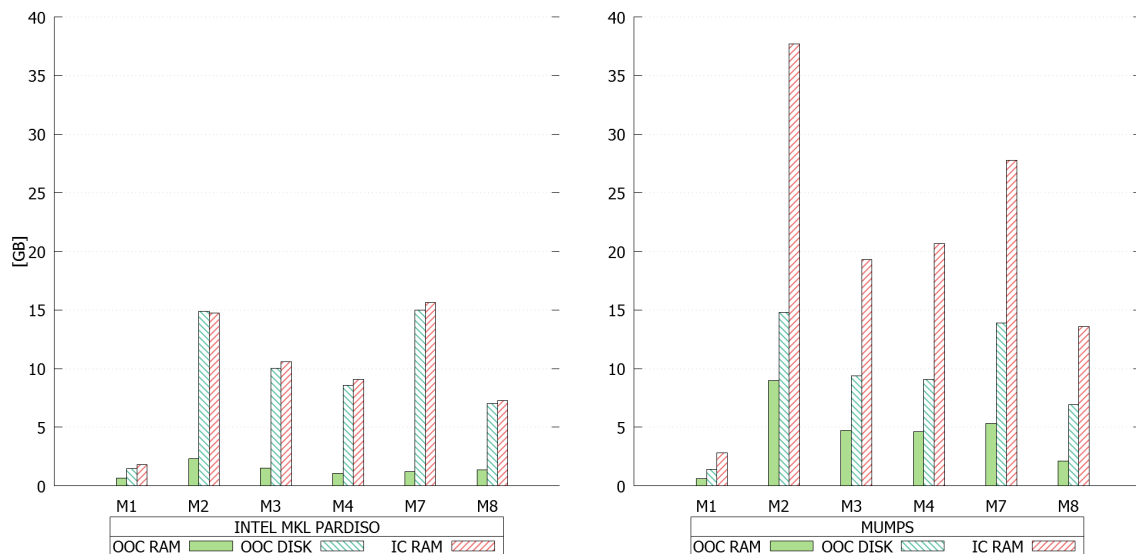
Εικόνα 17: Χρόνοι επίλυσης στη RAM σε HDD και flash SSD

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 17 παρουσιάζονται οι χρόνοι επίλυσης (wall-clock time) για τα τρία πακέτα λογισμικού.

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 WSMF 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 WSMF 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 18, για το WSMF 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



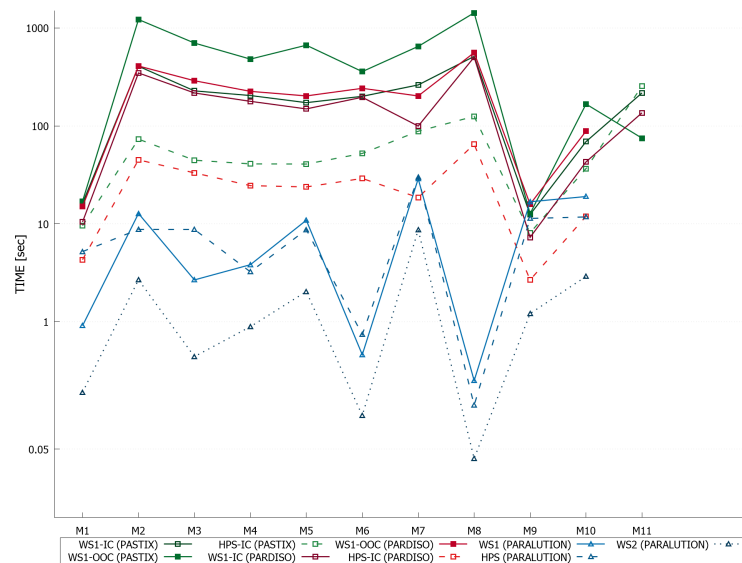
Εικόνα 18: Απαιτήσεις σε κύρια και δευτερεύουσα μνήμη

επίλυση ενός αραιού συστήματος απαιτεί την προσεκτική επιλογή του κατάλληλου αλγορίθμου (in-core/out-of-core) ανάλογα με το διαθέσιμο υλικό (hardware).

3.2.2 Σύγχρονες Πλατφόρμες Υλικού και Αραιά Γραμμικά Συστημάτα

Η οποιαδήποτε εξέλιξη στην τεχνολογία των Η/Υ μπορεί να επηρεάσει την επίλυση μεγάλων αραιών συστημάτων γραμμικών εξισώσεων. Εκτελέσαμε μια σειρά πειραμάτων, χρησιμοποιώντας τόσο άμεσες όσο και επαναληπτικές μεθόδους σε διαφορετικές πλατφόρμες σύγχρονου υλικού προκειμένου να ανιχνεύσουμε τα όρια και τις δυνατότητες κάθε συνδυασμού μεθόδου/υλικού. Τα πειράματα που αφορούν τις άμεσες μεθόδους τα πραγματοποιήσαμε με τα πακέτα: PASTIX [16] του INRIA και την έκδοση του PARDISO που περιλαμβάνεται στην INTEL MKL. Αντίστοιχα για τις επαναληπτικές μεθόδους χρησιμοποιήσαμε την βιβλιοθήκη PARALUTION [20].

Τα πειράματα πραγματοποιήθηκαν σε, α) ένα σταθμό εργασίας εξοπλισμένο με οδηγούς SSD (WS1), β) ένα σταθμό εργασίας με GPU (WS2) και γ) ένα εξυπηρετητή με δύο 18-core επεξεργαστές (HPS). Συγκεκριμένα ο σταθμός WS1 είναι ένα DELL Precision T3500 με 24GB DDR3 RAM, έναν 4-πύρηνο Intel Xeon W3550 3.06GHz. Διαθέτει έναν οδηγό SSD Intel 520 240GB SATA-III για το λειτουργικό σύστημα, έναν οδηγό OCZ Revodrive 350 PCIe 480GB και έναν SSD Intel 530 240GB ως επιπλέον αποθηκευτικά μέσα. Στα πειράματα χρησιμοποιήθηκε ο OCZ λόγω των καλύτερων επιδόσεών του. Ο δεύτερος σταθμός εργασίας είναι ένα DELL T5500 με 24GB DDR3 RAM, έναν επεξεργαστή 6-core Intel Xeon E5645 2,4GHz και μία Tesla C2075 GPU. Τέλος ο εξυπηρετητής είναι ένας HP



Εικόνα 19: Χρόνοι εκτέλεσης

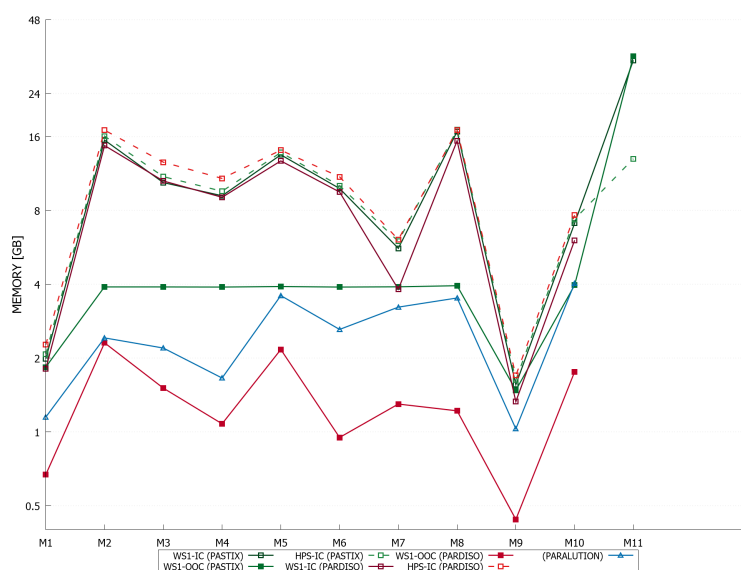
BL460c Gen 9 blade server με 64GB DDR4 RAM και 2 επεξεργαστές Intel Xeon CPUs E5-2695 v3 στα 2.30GHz με 14 cores ο καθένας.

ID	NAME	TYPE	$N(A)$	$NNZ(A)$
M1	Inline_1	R, S, PD	5,04E+05	1,87E+07
M2	Aster_perf-11a	R, S, I	8,54E+05	7,11E+07
M3	Audikw_1	R, S, I	9,44E+05	3,93E+07
M4	Nice20mc	R, S, I	7,16E+05	2,81E+07
M5	Flan_1565	R, S, PD	1,56E+06	1,14E+08
M6	StocF_1465	R, S, PD	1,47E+06	2,10E+07
M7	kkt_power	R, S	2,06E+06	1,28E+07
M8	Atmosmodl	R, U, I	1,49E+06	1,03E+07
M9	StepDHC_DIRK_1	R, U	4,10E+05	6,53E+06
M10	StepDHC_DIRK_2	R, U	1,64E+06	2,62E+07
M11	StepDHC_DIRK_3	R, U	6,55E+06	1,05E+08

Πίνακας 2: Αραιοί Πίνακες

Για τις ανάγκες των πειραμάτων χρησιμοποιήθηκαν συμμετρικοί και μη συμμετρικοί πίνακες. Οι αραιοί πίνακες M1-M8 προέρχονται από την συλλογή του Πανεπιστημίου της Φλόριντα, ενώ οι πίνακες M9-M11 προέρχονται από την διακριτοποίηση παραβολικών προβλημάτων πολλαπλών πεδίων και παρήχθησαν στα πλαίσια του έργου [6] [7]. Οι ιδιότητες των αραιών συστημάτων συνοψίζονται στον Πίνακα 2.

Σε κάθε πείραμα μετρήθηκαν οι χρόνοι εκτέλεσης και οι απαιτήσεις σε μνήμη. Τα πειράματα που αφορούν τις άμεσες μεθόδους εκτελέστηκαν στο σταθμό ερ-



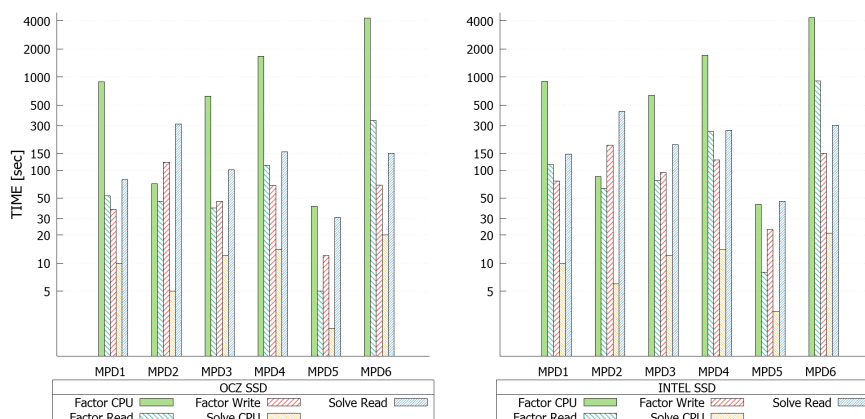
Εικόνα 20: Απαιτήσεις σε κύρια μνήμη

γασίας WS1 και στον εξυπηρετητή HPS. Δεν κατέστη δυνατή η επίλυση των περισσότερων συστημάτων με άμεσες μεθόδους στην GPU, εξαιτίας την περιορισμένης μνήμης της (μόνο τα συστήματα M1 και M9 επιλύθηκαν). Από την άλλη μεριά, οι επαναληπτικοί αλγόριθμοι εκτελέστηκαν και στις τρεις πλατφόρμες με επιτυχία. Η γραφική απεικόνιση των χρόνων εκτέλεσης παρουσιάζεται στην Εικόνα 19, αντίστοιχα οι απαιτήσεις σε μνήμη στην Εικόνα 20.

Τα αποτελέσματα δείχνουν ότι οι επαναληπτικές μέθοδοι υπερτερούν των άμεσων στις περισσότερες περιπτώσεις και έχουν την δυνατότητα να εκμεταλλεύονται αποδοτικότερα την επεξεργαστική ισχύ των σύγχρονων GPUs, αφού απαιτούν λιγότερη μνήμη. Σε ορισμένες περιπτώσεις στον εξυπηρετητή HPS οι άμεσες μέθοδοι δίνουν αποτελέσματα ταχύτερα από τις επαναληπτικές. Τέλος οι αλγόριθμοι που χρησιμοποιούν εξωτερική μνήμη σε συνδυασμό με νέες τεχνολογίες μνημών φαίνεται να αποτελούν ενδιαφέροντα εναλλακτική για τις άμεσες μεθόδους όταν η κύρια μνήμη δεν επαρκεί.

3.2.3 Αλγόριθμοι Εξωτερικής Μνήμης για Μνήμες Τύπου Flash

ID	NAME	$N(A)$	$NNZ(A)$	$NNZ(L)$	W_{FACT}	R_{FACT}	R_{SOLVE}
MPD1	audikw_1	9.44E+02	7.77E+07	1.25e+09	11,3GB	19,7GB	22,6GB
MPD2	ecology2	1.00E+03	4.996E+06	3.62e+07	9,05GB	3,1GB	18,1GB
MPD3	Flan_1565	1.14E+08	1.56E+06	1.48e+09	13,7GB	12,4GB	27,5GB
MPD4	Hook_1498	1.50E+06	5.94E+07	1.61e+09	15,8GB	45,7GB	31,6GB
MPD5	INLINE_1	5.04E+02	3.68E+07	1.61e+09	2,04GB	0,7GB	4,1GB
MPD6	Serena	1.39E+06	6.41E+07	2.74e+09	23,5GB	177,9GB	46,9GB



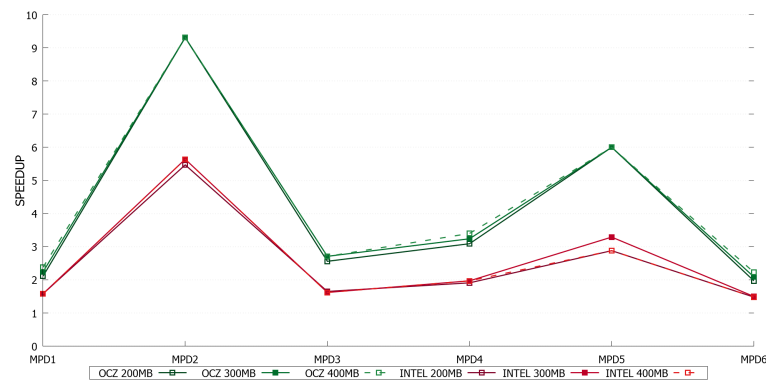
Εικόνα 21: Αρχικό κόστος

Πίνακας 3: Συμμετρικοί και θετικά ορισμένοι πίνακες

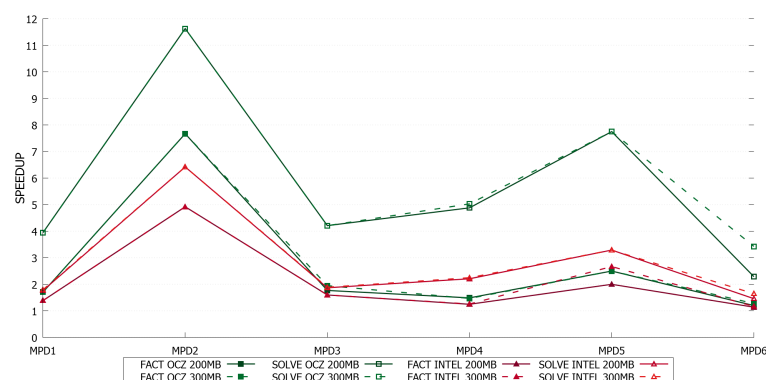
Βασιζόμενοι στα προηγούμενα συμπεράσματα μελετήσαμε τις δυνατότητες βελτίωσης της απόδοσης των αλγορίθμων εξωτερικής μνήμης στην flash και πιο συγκεκριμένα την υλοποίηση της μεθόδου Cholesky που περιλαμβάνεται στο TAUCS. Οι αραιοί πίνακες που χρησιμοποιήθηκαν στα πειράματα παρουσιάζονται στον Πίνακα 3. Τα πειράματα εκτελέστηκαν στον σταθμό εργασίας WS1, (που περιγράφηκε στην προηγούμενη παράγραφο), σε δύο διαφορετικούς οδηγούς SSD: έναν υψηλών επιδόσεων (OCZ Revodrive 350, PCI Express) και έναν μικρότερου κόστους και επιδόσεων (Intel 530 SSD, SATA III). Χρησιμοποιήθηκε DIRECT_IO για να μετρηθεί το πραγματικό κόστος της ανάγνωσης/εγγραφής (I/O) αποφεύγοντας την page cache του λειτουργικού συστήματος. Οι χρόνοι εκτέλεσης του αρχικού αλγορίθμου σε κάθε δίσκο, (χωρίς καμιά τροποποίηση), παρουσιάζονται στην Εικόνα 21.

Ο αλγόριθμος εξωτερικής μνήμης του TAUCS (out-of-core) αποθηκεύει τα στοιχεία του πίνακα της παραγοντοποίησης καθώς και βοηθητικές δομές σε ένα ή περισσότερα αρχεία στην δευτερεύουσα μνήμη. Η προσθήκη νέων δεδομένων γίνεται πάντα στο τέλος του αρχείου (append only) και δεν επιτρέπονται οι μεταβολές τους (in-place updates) μετά την εγγραφή. Η διαδικασία αυτή είναι αρκετά αποδοτική για την flash [27], ωστόσο παρατηρήσαμε ότι πραγματοποιείται ένας μεγάλος αριθμός των εγγραφών (writes), πολλές από τις οποίες έχουν μικρό μέγεθος. Προκειμένου να βελτιώσουμε την ταχύτητα εγγραφής στην flash, εισαγάγαμε έναν write buffer για την μαζική εκτέλεση εγγραφών (batch writes) και εκτελέσαμε πειράματα για διαφορετικά μεγέθη του buffer. Η επιτάχυνση (speedup) που επιτεύχθηκε για κάθε SSD παρουσιάζεται στην Εικόνα 22.

Η βελτίωση της απόδοσης της ανάκτησης (read) των αποθηκευμένων στοιχείων απαιτεί μεγαλύτερη παρέμβαση στον αλγόριθμο τόσο της παραγοντοποίησης



Εικόνα 22: Επιτάχυνση εγγραφής, με χρήση write buffer



Εικόνα 23: Επιτάχυνση ανάγνωσης, με χρήση read buffer και prefetching

σης (factorization) όσο και της επίλυσης (forward elimination & back substitution). Στο στάδιο της παραγοντοποίησης μειώσαμε τον αριθμό των λειτουργιών ανάγνωσης (read operations) των αποθηκευμένων βοηθητικών δομών (π.χ. supernode structure). Συγκεκριμένα χρησιμοποιήσαμε έναν read buffer και αξιοποιήσαμε το δέντρο απαλοιφής για να υλοποιήσουμε ένα μηχανισμό προ-ανάκτησης (prefetching) στα διάφορα στάδια της παραγοντοποίησης. Στο δε, στάδιο της επίλυσης χρησιμοποιήσαμε την ίδια μεθοδολογία όχι μόνο για την ανάκτηση των βοηθητικών δομών αλλά και για την ανάκτηση του πίνακα της παραγοντοποίησης. Η επιτάχυνση που μετρήσαμε σε κάθε περίπτωση, και για κάθε οδηγό, εμφανίζεται στην Εικόνα 23.

3.3 Επέκταση του ΠΕΠ για την Αξιοποίηση Υποδομής Cloud

3.3.1 Αξιοποίηση Υποδομής Cloud - Web Services

Σε αυτή την παράγραφο παρουσιάζουμε, ως παράδειγμα χρήσης web services για την αξιοποίηση υποδομής cloud, την υλοποίηση της υβριδικής αιτιοκρατικής / στοχαστικής μεθόδου για προβλήματα MDMP. Για την υλοποίηση έχουμε επιλέξει να αξιοποιήσουμε το Spyne [23], ένα από τα πολλά frameworks για την υποστήριξη cloud computing σε Python. Παρακάτω παρουσιάζουμε συνοπτικά τα κύρια βήματα.

Κώδικας 7: Ορισμός μιας απλής server function

```

1 from spyne import Application , rpc , ServiceBase , Integer ,
2     Double , Array
3 from spyne.protocol.soap import Soap11
4 import _hybridmc as core
5 import numpy as np
6
7 class MDMPService( ServiceBase ):
8     """1. convert the input Python lists to numpy arrays
9        2. call the core method and return output as Python list"""
10    @rpc( Array( Double ) , Integer , Array( Double ) , Integer ,
11         String , String , Boolean ,
12         _returns=Array( Double ))
13    def montecarlo( ctx , dims , dim , coords , nof_nodes , f , q , OpenCL ):
14        D = np.array( dims , dtype=np.float_ )
15        node_coord = np.array( coords , dtype=np.float_ )
16        if not OpenCL:
17            f = Expression( f )
18            q = Expression( q )
19        return core.montecarlo( D , dim , node_coord , nof_nodes , f , q )

```

Ο Κώδικας 7 δείχνει την υλοποίηση μιας απλής server function που λειτουργεί ως περιτύλιγμα (wrapper) της υλοποίησης της υβριδικής μεθόδου (montecarlo()). Η ενεργοποίηση του server μπορεί να γίνει όπως φαίνεται στον Κώδικα 8. Ο server, στη συγκεκριμένη περίπτωση, ενεργοποιείται στο τοπικό μηχάνημα στο port 8000.

Κώδικας 8: Ενεργοποίηση server.

```

1 from spyne import Application
2 from spyne.server.wsgi import WsgiApplication
3 from wsgiref.simple_server import make_server
4 from mdmp_service import MDMPService
5 import logging
6

```

```

7 app = Application([MDMPService], 'spyne.examples.hello.soap',
8                     in_protocol=Soap11(validator='lxml'),
9                     out_protocol=Soap11())
10 wsgi_app = WsgiApplication(app)
11 # logging code omitted
12 server = make_server('127.0.0.1', 8000, wsgi_app).serve_forever()

```

Η κλάση `RemoteClient` χρησιμοποιεί τον `Suds` [12], έναν ελαφρύ client Python για web services βασισμένα σε SOAP, ο οποίος διαβάζει και ερμηνεύει αρχεία WSDL κατά το χρόνο εκτέλεσης. Αφού ερμηνεύσει το WSDL, παράγει από αυτό μια περιγραφή προς το χρήστη για τον τρόπο με τον οποίο θα πρέπει να επικοινωνήσει με τον server.

Προκειμένου να προσφερθεί ένα συνεπές API μεταξύ τοπικών και απομακρυσμένων μεθόδων, εκτός από την κλάση `RemoteClient` η πλατφόρμα ορίζει και μία κλάση `LocalClient` με τις ίδιες ακριβώς μεθόδους (για την περίπτωση που η εκτέλεση γίνεται τοπικά). Όταν χρησιμοποιείται η κλάση `RemoteClient`, τα δεδομένα εισόδου στέλνονται στον απομακρυσμένο server, ο οποίος απαντά επιστρέφοντας τα αποτελέσματα. Ο Κώδικας 9 απεικονίζει τον βασικό ορισμό των κλάσεων `RemoteClient` και `LocalClient`.

Κώδικας 9: Βασικός ορισμός κλάσεων `RemoteClient` και `LocalClient`.

```

1 from suds.client import Client
2
3 class RemoteClient(Client):
4     def __init__(self, *args, **kwargs):
5         self.is_local = False
6         Client.__init__(self, *args, **kwargs)
7
8 class LocalClient():
9     def __init__(self, *args, **kwargs):
10        self.is_local = True

```

Τυχόν διαφορές υλοποίησης μεταξύ των τοπικών και των απομακρυσμένων κλήσεων δεν είναι ορατές στον τελικό χρήστη, ο οποίος και στις δύο περιπτώσεις χρησιμοποιεί τη λειτουργικότητα και λαμβάνει τα αποτελέσματα με τον παραδοσιακό τρόπο μιας κλήσης συνάρτησης, όπως φαίνεται στον Κώδικα 10.

Κώδικας 10: Χρήση των αντικειμένων `Client` (`Local` ή `Remote`) από τον κώδικα του τελικού χρήστη.

```

1 from dolfin import *
2 import hybridmc as hmc
3
4 if use_remote_client:
5     client = hmc.RemoteClient(wSDL_url)

```

```

6      client.set_options(timeout=90) # ...
7  else:
8      client = hmc.LocalClient()

```

3.3.2 Αξιοποίηση Υποδομής Cloud - Πλατφόρμα IRaaS

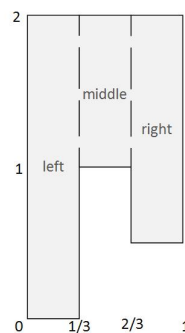
Για την αξιολόγηση της απόδοσης της προτεινόμενης εφαρμογής, χρησιμοποιήθηκε το παρακάτω ελλειπτικό πρόβλημα:

$$Lu \equiv -\nabla^2 u(x, y) + \gamma^2 u(x, y) = f(x, y) \quad (x, y) \in \Omega \quad (3)$$

$$u(x, y) = u^b(x, y) \quad (x, y) \in \partial\Omega \quad (4)$$

με $f(x, y)$ και $u(x, y)$ τέτοια ώστε η πραγματική λύση να είναι:
 $u(x, y) = e^{(y(x+4))} x(x-1)(x-0.7)y(y-0.5)$

Το πρόβλημα αποτελείται από τρία διαφορετικά subdomains, τα οποία απεικονίζονται στο Σχήμα 24.



Εικόνα 24: Πρόβλημα-μοντέλο με 3 υποχωρία

Τα σημεία διεπαφής είναι το $x_1 = 1/3$ και $x_2 = 2/3$ και η τιμή του γ είναι $\gamma^2 = 2$. Εξετάζονται επτά διαφορετικά μεγέθη πλέγματος βάσει της παραμέτρου διακριτοποίησης h , η οποία έχει θεωρηθεί ίση τόσο στην x όσο και στην κατεύθυνση y . Ο αριθμός των σημείων των διεπαφών σε κάθε περίπτωση ισούται με τον αριθμό των σημείων της y διάστασης του μεσαίου subdomain. Αυξάνεται, δηλαδή, από τα 6 έως τα 321 σημεία. Το αριστερό subdomain είναι το μεγαλύτερο πρόβλημα από τα τρία με σημαντική διαφορά από τα υπόλοιπα (τέσσερις φορές περίπου μεγαλύτερο από το μεσαίο) ενώ το δεξί subdomain είναι περίπου δύο φορές μεγαλύτερο από το μεσαίο. Οι διεπαφές έχουν όλες ίσο αριθμό σημείων και άρα απαιτούν ίσο φόρτο εργασίας για την επεξεργασία τους. Οι περιπτώσεις που θα μελετήσουμε και τα προκύπτοντα μεγέθη του πλέγματος για το αριστερό, το μεσαίο και το δεξί subdomain, φαίνονται στον Πίνακα 4.

Πίνακας 4: Εξεταζόμενες περιπτώσεις

Case	H	Left	Middle	Right
C1	0.1	4x21	4x6	4x11
C2	0.05	8x41	8x11	8x21
C3	0.025	14x81	14x21	14x41
C4	0.0125	28x161	28x41	28x81
C5	0.00625	55x321	55x81	55x161
C6	0.003125	108x641	108x161	108x321
C7	0.0015625	214x1281	214x321	214x641

Η εφαρμογή IRaaS εφαρμόστηκε και εγκαταστάθηκε σε μία μικρή ιδιωτική cloud υποδομή στις εγκαταστάσεις του εργαστηρίου Αναγνώρισης Προτύπων του τμήματος Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Πατρών. Η συγκεκριμένη υποδομή περιλαμβάνει έναν server με Intel(R) Xeon(R) CPU E3-1220@3.10GHz με 4 cores και 16GB RAM.

Οι συνδυασμοί των υπολογιστικών πόρων που χρησιμοποιήθηκαν για να παραχθεί ένας κανόνας με τον οποίο θα υπολογίζεται η κατάλληλη και αυτόματη κατανομή πόρων για κάθε διαφορετική περίπτωση του προβλήματος παρουσιάζεται στον Πίνακα 5.

Πίνακας 5: Χρησιμοποιηθέντες υπολογιστικοί πόροι

Resources	Left domain			Middle domain			Right domain		
	Cores	GHz	GB	Cores	GHz	GB	Cores	GHz	GB
R1	1	2	1	1	2	1	1	2	1
R2	1	2	2	1	2	2	1	2	2
R3	1	2	4	1	2	1	1	2	2
R4	1	2	4	1	1.5	2	1	2	2

Ο συνδυασμός R1 είναι η ελάχιστη παροχή πόρων όπου κάθε subdomain επιλύεται σε ένα εικονικό μηχάνημα που έχει επεξεργαστή με 1 core, των 2 GHz και 1 GB RAM. Στην περίπτωση R2 κάθε subdomain επιλύεται σε ένα εικονικό μηχάνημα που έχει επεξεργαστή με 1 core, των 2 GHz και 2 GB RAM. Η περίπτωση R3 είναι πιο πολύπλοκη από τις υπόλοιπες περιπτώσεις αφού το αριστερό subdomain επιλύεται σε ένα εικονικό μηχάνημα που έχει επεξεργαστή με 1 core, των 2 GHz και 4 GB RAM, το μεσαίο subdomain σε ένα εικονικό μηχάνημα που έχει επεξεργαστή με 1 core, των 2 GHz και 1 GB RAM και το δεξί subdomain επιλύεται σε ένα εικονικό μηχάνημα που έχει επεξεργαστή με 1 core, των 2 GHz και 2 GB RAM. Αυτή η περίπτωση μελετήθηκε με τέτοιο τρόπο ώστε να δούμε την ανταπόκριση του συστήματος όταν το μέγεθος της RAM είναι ανάλογο του μέγεθος του subdomain που επιλύει (μεγάλη RAM για την ει-

κονική μηχανή που επιλύει το αριστερό subdomain που είναι το μεγαλύτερο, λίγο μικρότερη RAM για το δεξί subdomain που είναι το αμέσως μικρότερο, και την πιο μικρή RAM στο μεσαίο subdomain που είναι και το μικρότερο από όλα). Η περίπτωση R4 είναι παρόμοια με την περίπτωση R3 μόνο που εδώ δόθηκε στο εικονικό μηχάνημα που επιλύει το μεσαίο subdomain επεξεργαστής με 1 core, των 1.5 GHz και 2 GB RAM.

Ο Πίνακας 6 παρουσιάζει τους τελικούς χρόνους εκτέλεσης της μεθόδου GEO για όλες τις πιθανές περιπτώσεις συνδυασμών ανάθεσης πόρων στις εικονικές μηχανές. Κάθε τιμή του πίνακα περιλαμβάνει τον χρόνο για την επίλυση των τριών subdomains μαζί με το χρόνο για τον υπολογισμό των διεπαφών και τον χρόνο που χρειάζεται για την απαραίτητη επικοινωνία των 15 επαναλήψεων.

Πίνακας 6: Χρόνοι εκτέλεσης

Resources/Case	R1	R2	R3	R4
C1	15.041	15.082	15.016	15.042
C2	15.411	15.457	15.320	15.345
C3	15.724	15.894	15.736	15.735
C4	19.169	21.442	19.234	19.334
C5	28.548	37.271	28.669	28.470
C6	71.416	81.252	70.972	71.499
C7	500.146	307.241	297.122	298.870

Στις πρώτες 5 περιπτώσεις παρατηρείται ξεκάθαρα ότι οι χρόνοι εκτέλεσης σε κάθε περίπτωση συνδυασμού ανάθεσης πόρων είναι σχεδόν ίσοι ανεξαρτήτως των υπολογιστικών πόρων που χρησιμοποιήθηκαν. Σαν αποτέλεσμα ο συνδυασμός R1, που γίνεται η ελάχιστη ανάθεση πόρων στις εικονικές μηχανές, φαίνεται να είναι επαρκής ούτως ώστε να γίνει όσο το δυνατόν μικρότερη χρήση υπολογιστικών πόρων. Από την άλλη, στην περίπτωση C6 η πολυπλοκότητα της επίλυσης του προβλήματος μεγαλώνει και παρόλο που οι χρόνοι για κάθε περίπτωση ανάθεσης πόρων είναι σχεδόν οι ίδιοι, επιλέγεται η περίπτωση R1 που παρέχει την μικρότερη ανάθεση πόρων. Για την περίπτωση C7 του προβλήματος, ο συνδυασμός ανάθεσης πόρων R3 φαίνεται να είναι ο καταλληλότερος αφού με τη χρήση του και γίνεται μικρή ανάθεση πόρων αλλά και παρέχονται μικροί χρόνοι εκτέλεσης σε σύγκριση με όλες τις υπόλοιπες περιπτώσεις συνδυασμών ανάθεσης πόρων. Οι χρόνοι του προβλήματος C7 για τις περιπτώσεις πόρων R3 και R4 είναι σχεδόν ίδιοι, όμως επιλέγεται ως καταλληλότερη η περίπτωση R3 αφού με αυτήν γίνεται μικρότερη ανάθεση πόρων.

Με βάση τις παραπάνω παρατηρήσεις, δημιουργήθηκε για αυτά τα προβλήματα ένας κανόνας για την αυτόματη ανάθεση υπολογιστικών πόρων στις εικονικές μηχανές ανάλογα με τα δεδομένα εισόδου (περίπτωση προβλήματος C1-C7):

Κώδικας 11: Αλγόριθμος ανάθεσης υπολογιστικών πόρων

```
1 if subdomain size <= 75000:  
2     virtual solving node RAM = 1GB  
3     virtual solving node processor = 1core , 2GHz  
4 else if 75000 < subdomain size <= 150000:  
5     virtual solving node RAM = 2GB  
6     virtual solving node processor = 1core , 2GHz  
7 else if subdomain size > 150000:  
8     virtual solving node RAM = 4GB  
9     virtual solving node processor = 1core , 2GHz
```

Όταν αυτός ο κανόνας χρησιμοποιηθεί από την ροή εργασιών, είτε η μέθοδος επίλυσης είναι η GEO είτε είναι η ROB, επιτυγχάνονται ταυτόχρονα τρεις διαφορετικοί στόχοι:

1. Ανατίθενται στις εικονικές μηχανές οι μικρότεροι δυνατοί (σε πλήθος) πόροι για την επίλυση του εκάστοτε προβλήματος,
2. Η επίλυση του προβλήματος πραγματοποιείται με το σχεδόν μικρότερο χρόνο εκτέλεσης και
3. Η ανάθεση των υπολογιστικών πόρων γίνεται στο παρασκήνιο χωρίς καμία παρέμβαση από το χρήστη.

Με τη χρήση λοιπόν του παραπάνω κανόνα, μέσω της εφαρμογής μπορεί να επιτευχθεί ο καλύτερος δυνατός συνδυασμός ανάθεσης πόρων και χρόνου εκτέλεσης. Η επίτευξη αυτού του στόχου, δηλαδή η ανάθεση του καλύτερου συνδυασμού υπολογιστικών πόρων για την επίλυση ενός προβλήματος εξαρτάται αποκλειστικά από τα δεδομένα εισόδου. Για αυτό το λόγο, η συγκεκριμένη εφαρμογή παραμετροποιήθηκε ούτως ώστε να υιοθετηθεί ο παραπάνω κανόνας.

Χρησιμοποιώντας το νέο κανόνα και εκμεταλλευόμενοι των πλεονεκτημάτων του, εκτελέστηκε για όλα τα προβλήματα εισόδου τόσο η μέθοδος GEO όσο και η μέθοδος ROB. Στον Πίνακα 7 φαίνονται οι αντίστοιχοι χρόνοι εκτέλεσης που βρέθηκαν με τη χρήση της προτεινόμενης εφαρμογής.

Στον Πίνακα 8 συγκρίνονται οι χρόνοι εκτέλεσης της IRaaS με τους αντίστοιχους χρόνους εκτέλεσης των προβλημάτων με την προηγούμενη υλοποίησή μας σε 3 διαφορετικές προκαθορισμένες εικονικές μηχανές που δεν βρίσκονταν πάνω σε κάποια cloud υποδομή. Τα εικονικά αυτά μηχανήματα ήταν τρία στο πλήθος και είχαν 4 cores και 2GB RAM το καθένα, και έτρεχαν πάνω σε έναν 2 x Intel(R) Xeon(R) CPU E5-2620, 2.00GHz server με εγκατεστημένο XenServer.

Παρατηρείται λοιπόν ότι η παρούσα εφαρμογή παρέχει μια σημαντική μείωση στο χρόνο εκτέλεσης (και για τις δυο μεθόδους) συγκριτικά με εκτελέσεις

Πίνακας 7: Χρόνοι εκτέλεσης με αυτόματη ανάθεση πόρων

Problem	Execution Times	
	ROB	GEO
P1	15.348	15.041
P2	15.373	15.411
P3	16.022	15.724
P4	19.223	19.169
P5	27.313	28.548
P6	65.386	71.416
P7	265.415	297.122

Πίνακας 8: Χρόνοι εκτέλεσης - Cluster / Cloud

Problem	Comparison Analysis			
	Virtual Implementation ROB	Proposed Cloud Implementation ROB	Virtual Implementation GEO	Proposed Cloud Implementation GEO
P1	16.004	15.348	15.495	15.041
P2	16.281	15.373	16.068	15.411
P3	17.392	16.022	18.140	15.724
P4	23.634	19.223	25.595	19.169
P5	45.456	27.313	51.808	19.169
P6	132.179	65.386	166.633	71.416
P7	1099.05	265.415	735.731	297.122

που πραγματοποιούνταν σε προκαθορισμένες εικονικές μηχανές που δεν άνηκαν στο cloud. Παρατηρείται επίσης ότι όσο μεγαλώνει το μέγεθος του προβλήματος, τόσο πιο σημαντική είναι η μείωση του χρόνου εκτέλεσης, γεγονός που καθιστά την εφαρμογή κατάλληλη για την ανάλυση μεγάλου όγκου δεδομένων.

Παράλληλα, η εφαρμογή παρέχει και σημαντική μείωση των υπολογιστικών πόρων που χρειάζονται για να γίνει η ίδια ακριβώς ανάλυση (βάσει του κανόνα που εφαρμόστηκε). Στον Πίνακα 9 παρουσιάζονται αναλυτικά οι υπολογιστικοί πόροι που χρησιμοποιήθηκαν για την επίλυση κάθε περίπτωσης δεδομένων εισόδου του προβλήματος από την υλοποίηση με τις προκαθορισμένες εικονικές μηχανές και από την εφαρμογή IRaaS και για τις δυο μεθόδους (GEO και ROB).

Παρατηρείται ότι με βάση τον κανόνα αυτόματης ανάθεσης πόρων, για την περίπτωση C7 η εικονική μηχανή 1 χρειάζεται 4 GB RAM και η εικονική μηχανή 3 χρειάζεται 2 GB RAM. Αυτό συμβαίνει διότι στη C7 μεγαλώνει σημαντικά το μέγεθος του προβλήματος. Οι προκαθορισμένες εικονικές μηχανές, ανεξάρτητα από το μέγεθος του προβλήματος, χρειάζονται για όλες τις περιπτώσεις τον ίδιο αριθμό υπολογιστικών πόρων για κάθε εικονική μηχανή. Η βελτίωση και ουσιαστικά η μείωση της χρήσης υπολογιστικών πόρων (ανάλογα με το μέγεθος του προβλήματος) από την εφαρμογή IRaaS έχει ως αποτέλεσμα και την μείωση

Πίνακας 9: Σύγκριση υπολογιστικών πόρων για GEO / ROB

ΠΕΡΙΠΤΩΣΗ ΠΡΟΒΛΗΜΑΤΟΣ	ΠΡΟΚΑΘΟΡΙΣΜΕΝΕΣ ΕΙΚΟΝΙΚΕΣ ΜΗΧΑΝΕΣ (και για τις τρεις)	ΕΙΚΟΝΙΚΕΣ ΜΗΧΑΝΕΣ IRaaS		
		ΕΙΚΟΝΙΚΗ ΜΗΧΑΝΗ 1	ΕΙΚΟΝΙΚΗ ΜΗΧΑΝΗ 2	ΕΙΚΟΝΙΚΗ ΜΗΧΑΝΗ 3
C1	4cores-2GHz/ 2 GB RAM	1core-2GHz/ 1 GB RAM	1core-2GHz/ 1 GB RAM	1core-2GHz/ 1 GB RAM
C2	4cores-2GHz/ 2 GB RAM	1core-2GHz/ 1 GB RAM	1core-2GHz/ 1 GB RAM	1core-2GHz/ 1 GB RAM
C3	4cores-2GHz/ 2 GB RAM	1core-2GHz/ 1 GB RAM	1core-2GHz/ 1 GB RAM	1core-2GHz/ 1 GB RAM
C4	4cores-2GHz/ 2 GB RAM	1core-2GHz/ 1 GB RAM	1core-2GHz/ 1 GB RAM	1core-2GHz/ 1 GB RAM
C5	4cores-2GHz/ 2 GB RAM	1core-2GHz/ 1 GB RAM	1core-2GHz/ 1 GB RAM	1core-2GHz/ 1 GB RAM
C6	4cores-2GHz/ 2 GB RAM	1core-2GHz/ 1 GB RAM	1core-2GHz/ 1 GB RAM	1core-2GHz/ 1 GB RAM
C7	4cores-2GHz/ 2 GB RAM	1core-2GHz/ 4 GB RAM	1core-2GHz/ 1 GB RAM	1core-2GHz/ 2 GB RAM

του συνολικού κόστους για την εκτέλεση κάθε είδους προβλήματος. Το κόστος υπολογίζεται συναρτήσει των πόρων που δεσμεύει η κάθε εικονική μηχανή αλλά και του απαιτούμενου χρόνου που δεσμεύεται η εικονική μηχανή και είναι σε λειτουργία (μέχρι να πραγματοποιήσει μια ανάλυση). Οπότε λαμβάνοντας υπόψη το μέγεθος των εικονικών μηχανών που χρειάζονται για κάθε περίπτωση επίλυσης προβλήματος, τόσο για τις προκαθορισμένες εικονικές μηχανές όσο και για τις εικονικές μηχανές της εφαρμογής IRaaS, και του χρόνου που χρειάζεται κάθε ξεχωριστή περίπτωση για να επιλύσει το πρόβλημα, έχουμε και μείωση κόστους με χρήση της εφαρμογής IRaaS.

Καταλήγουμε λοιπόν στο συμπέρασμα ότι η εφαρμογή παρέχει καλύτερους χρόνους εκτέλεσης, μικρότερη δέσμευση υπολογιστικών πόρων και κατά συνέπεια μικρότερο κόστος για την επίλυση κάθε περίπτωσης του ίδιου προβλήματος συγκριτικά με παλαιότερη υλοποίησή μας όπου οι εικονικές μηχανές είχαν προκαθορισμένους υπολογιστικούς πόρους και δεν βρίσκονταν εγκατεστημένες σε κάποια cloud υποδομή.

4 Παραδοτέα

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

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).
- Προσθήκες στην πλατφόρμα FEniCS για την αξιοποίηση cloud με χρήση web services.
- Προσθήκες στην πλατφόρμα FEniCS για την αξιοποίηση προσαρμοσμένων (custom) μηχανισμών cloud.
- Η προσαρμοσμένη (custom) πλατφόρμα cloud IRaaS.

Πλέον των παραδοτέων που προέβλεπε το τεχνικό δελτίο, στα πλαίσια της δράσης παρήχθησαν και οι ακόλουθες δημοσιεύσεις:

- 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.
- Aigli Korfiati, Niki Sfika, Konstantis Daloukas, Christos Alexakos, Panagiota Tsompanopoulou and Spiros Likothanassis, "IRaaS: A Cloud Implementation of an Interface Relaxation Method for the Solution of PDEs", 2015 International Conference of Parallel and Distributed Computing (ICPDC'15), part of World Congress on Engineering 2015 (WCE 2015). (Certificate of Merit).
- Athanasios Fevgas, Konstantis Daloukas, Panagiota Tsompanopoulou, Panayiotis Bozanis, "Efficient Solution of Large Sparse Linear Systems in Modern Hardware", 6th International Conference on Information, Intelligence, Systems and Applications (IISA2015), 06-08 Jul 2015, Corfu, Greece.
- Niki Sfika, Aigli Korfiati, Christos Alexakos, Spiros Likothanassis, Konstantis Daloukas and Panagiota Tsompanopoulou, "Dynamic Cloud Resources Allocation on Multidomain/Multiphysics Problems", The 3rd International Conference on Future Internet of Things and Cloud (FiCloud 2015), 24-26 August 2015, Rome, Italy.
- Christos Antonopoulos, Manolis Maroudas and Manolis Vavalis, "On PDE problem solving environments for multidomain multiphysics problems", Journal of Mathematics and Statistical Science. Accepted 10/2015, in press.

- Athanasios Fevgas, Konstantis Daloukas, Panagiota Tsompanopoulou, and Panayiotis Bozanis, "A Study of Sparse Matrix Methods on New Hardware: Advances and Challenges", International Journal of Monitoring and Surveillance Technologies Research (IJMSTR) (to appear).

Επίσης, έχει υποβληθεί προς κρίση η ακόλουθη εργασία:

- N. Vilanakis, E. Maroudas, E. Mathioudakis, Ch. Antonopoulos, Y. Saridakis and M. Vavalis, "Schwarz Splitting for the Steady State Problem of Saltwater Intrusion in Coastal Aquifers", 2015 International Conference on Mathematical Methods & Computational Techniques in Science & Engineering (MMCTSE).

Τέλος, είναι υπό συγγραφή 2 επιστημονικά άρθρα για υποβολή σε επιστημονικά περιοδικά. Το πρώτο αφορά την παράλληλη υλοποίηση 2 μεθόδων της Interface Relaxation (GEO, ROB) στην Πλατφόρμα FEniCS σε περιβάλλον Cloud. Το δεύτερο αφορά την παράλληλη υλοποίηση της μεθόδου ALOPEX στην Πλατφόρμα FEniCS και την εφαρμογή της στο πρόβλημα Περιβαλλοντικής Μηχανικής (πρόβλημα υφαλμύρισης).

5 Συνεργασίες

Όπως αναφέρθηκε στην εισαγωγική παράγραφο, 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 10 συνοψίζονται οι βασικές δραστηριότητες που αναπτύχθηκαν στα πλαίσια της δράσης και αναφέρονται οι ομάδες που συμμετείχαν.

6 Μελλοντικές Δράσεις

Στα πλαίσια της παρούσας δράσης επελέχθη καταρχήν το FEniCS ως κατάλληλο περιβάλλον επίλυσης διαφορικών εξισώσεων. Ακολούθως επεκτάθηκε με μέθοδοι απαραίτητες για την υποστήριξη των εφαρμογών στις οποίες στοχεύει το έργο MATENVMED. Επίσης, αναπτύχθηκε υποδομή ώστε να είναι δυνατή η αξιοποίηση από το FEniCS μοντέρνων αρχιτεκτονικών (GPUs, FPGAs, γρήγορες συσκευές αποθήκευσης και υποδομές cloud). Ειδικά για την περίπτωση της αξιοποίησης υποδομών cloud εξετάστηκαν δύο διαφορετικές προσεγγίσεις: η παροχή τους με τη μορφή προτυποποιημένων web services, και η προσέγγιση προσαρμοσμένης υλοποίησης περιβάλλοντος cloud (IRaaS).



	ΚΕΟ 1	ΚΕΟ 2	ΚΕΟ 3
Προδιαγραφές ΠΕΠ	x	x	x
Απαιτήσεις για υποστήριξη προβλημάτων MDMP	x	x	x
Υλοποίηση Μεθόδου GEO		x	x
Υλοποίηση Μεθόδου ROB		x	x
Υλοποίηση Μεθόδου Schwarz	x	x	
Υλοποίηση Μεθόδου Hybrid		x	x
Υποδομή Υποστήριξης GPUs/FPGAs/Flash Storage		x	
Υποδομή Υποστήριξης Cloud (Web Services)		x	
Υποδομή Υποστήριξης Cloud (IRaaS)			x
Εφαρμογή σε Προβλήματα Υφαλμύρισης	x	x	x

Πίνακας 10: Συνεργασίες στα πλαίσια της Δράσης 4.1.

Βασική επιδίωξη στο άμεσο μέλλον είναι η προτυποποίηση της υποδομής που αναπτύχθηκε και η προσφορά της στην κοινότητα του FEniCS ώστε να είναι ευρέως διαθέσιμη και να αυξηθεί η βάση χρηστών της. Επιπλέον, ιδιαίτερο ενδιαφέρον έχει η υλοποίηση επιπλέον υπηρεσιών για την υποστήριξη εφαρμογών MDMP και η βελτιστοποίηση επίδοσης. Τέλος, στόχος είναι η αξιοποίηση της πλατφόρμας που αναπτύχθηκε και για άλλες πραγματικές, κρίσιμες εφαρμογές παρόμοιες με αυτές που αποτέλεσαν αντικείμενο του έργου.

Αναφορές

- [1] Boost basic linear algebra library, 2014.
- [2] Overview of mtl4, 2014.
- [3] Martin S. Alnæs. *UFL: a Finite Element Form Language*, chapter 17. Springer, 2012.
- [4] 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.
- [5] 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.
- [6] John E Athanasakis, MG Papadomanolaki, EP Papadopoulou, and YG Saridakis. Discontinuous hermite collocation and diagonally implicit rk3 for a brain tumour invasion model. In *Proceedings of the World Congress on Engineering*, volume 1, 2013.

- [7] John E Athanasakis, EP Papadopoulou, and YG Saridakis. Tensor product discontinuous hermite collocation for linear & nonlinear parabolic multidomain problems in 2+1 dimensions. In *Work in progress, personal communication*, 2015.
- [8] 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.
- [9] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.
- [10] 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.
- [11] 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.
- [12] Fedorahosted.org. Suds is a lightweight soap python client for consuming web services., 2014. [Online; accessed 23-October-2014].
- [13] N Goedel, T Warburton, and M Clemens. Gpu accelerated discontinuous galerkin fem for electromagnetic radio frequency problems. In *2009 IEEE Antennas and Propagation Society International Symposium*, 2009.
- [14] Michael A. Heroux, Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, Richard B. Lehoucq, Kevin R. Long, Roger P. Pawlowski, Eric T. Phipps, Andrew G. Salinger, Heidi K. Thornquist, Ray S. Tuminaro, James M. Willenbring, Alan Williams, and Kendall S. Stanley. An overview of the trilinos project. *ACM Trans. Math. Softw.*, 31(3):397–423, 2005.
- [15] Xin-Ming Huang and Jing Ma. An fpga-based accelerator for multiphysics modeling. In *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms, ERSA'04, June 21-24, 2004, Las Vegas, Nevada, USA*, pages 209–212, 2004.

- [16] INRIA. Pastix, 2015. <http://pastix.gforge.inria.fr/>.
- [17] Volodymyr V Kindratenko, Jeremy J Enos, Guochun Shi, Michael T Showerman, Galen W Arnold, John E Stone, James C Phillips, and Wenmei Hwu. Gpu clusters for high-performance computing. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–8. IEEE, 2009.
- [18] Sándor Kocsárdi, Zoltán Nagy, Árpád Csík, and Péter Szolgay. Simulation of 2d inviscid, adiabatic, compressible flows on emulated digital cnn-um. *Int. J. Circuit Theory Appl.*, 37(4):569–585, May 2009.
- [19] Aigli Korfiati, Niki Sfika, Konstantis Daloukas, Christos Alexakos, Panagiota Tsompanopoulou, and Spiros Likothanassis. "iraas: A cloud implementation of an interface relaxation method for the solution of pdes". In *Proceedings of 2015 International Conference of Parallel and Distributed Computing (ICPDC'15), part of World Congress on Engineering 2015 (WCE 2015) - Certificate of Merit*, 2015.
- [20] PARALUTION Labs. Paralution, 2015. <http://www.paralution.com/>.
- [21] Anders Logg, Kent-Andre Mardal, Garth N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.
- [22] Anders Logg and Garth N. Wells. Dofin: Automated finite element computing. *ACM Transactions on Mathematical Software*, 37(2), 2010.
- [23] Arskom Ltd. spyne - rpc that doesn't break your back., 2014. [Online; accessed 23-October-2014].
- [24] A. Mantoglou, M. Papantoniou, and P. Giannouloupoulos. Management of coastal aquifers based on nonlinear optimization and evolutionary algorithms. *Journal of Hydrology*, 297:209–228, 2004.
- [25] R. Nunez, J. Gonzalez, and J. Camberos. Large-scale numerical solution of partial differential equations with reconfigurable computing, 2007 2007.
- [26] N. Papazoglou. *Web Services: Principles and Technology*. Pearson Prentice Hall, 2008.
- [27] Hongchan Roh, Sungho Kim, Daewook Lee, and Sanghyun Park. As b-tree: A study of an efficient b+-tree for ssds. *Journal of Information Science and Engineering*, 30(1):85–106, 2014.

- [28] Vladimir Rotkin and Sivan Toledo. The design and implementation of a new out-of-core sparse cholesky factorization method. *ACM Transactions on Mathematical Software (TOMS)*, 30(1):19–46, 2004.
- [29] Olaf Schenk. Scalable parallel sparse lu factorization methods on shared memory multiprocessors. Technical report, ETH. Zurich, 2000.
- [30] Olaf Schenk and Klaus Gartner. Solving unsymmetric sparse systems of linear equations with pardiso. *Future Generation Computer Systems*, 20(3):475–487, 2004.
- [31] Niki Sfika, Aigli Korfiati, Christos Alexakos, Spiros Likothanassis, Konstantis Daloukas, and Panagiota Tsompanopoulou. "dynamic cloud resources allocation on multidomain/multiphysics problems". In *Proceedings of the 3rd International Conference on Future Internet of Things and Cloud (FiCloud 2015)*, Rome, Italy, Aug 2015.
- [32] Richard A Shapiro. *Adaptive finite element solution algorithm for the Euler equations*, volume 32. Vieweg+ Teubner Verlag, 2013.
- [33] 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.
- [34] P. Stratis, Z. Dokou, G. Karatzas, E. Papadopoulou, and Y. Saridakis. Stochastic optimization and numerical simulation for pumping management of the hersonissos freshwater coastal aquifer in crete. In *Procs of 2015 Int. Conf. on Water Resources, Hydraulics and Hydrology, Zakynthos, Greece*, pages 329–334.
- [35] P. Stratis, G. Karatzas, E. Papadopoulou, M. Zakynthinaki, and Y. Saridakis. Stochastic optimization for an analytical method of saltwater intrusion on coastal aquifers. submitted to PLOSone, 2015.
- [36] P. Stratis, Y. Saridakis, M. Zakynthinaki, and E. Papadopoulou. Alopex stochastic optimization for pumping management in fresh water coastal aquifers. In *Journal of Physics: Conference Series*, volume 490. 2014.
- [37] Julien C Thibault. *Implementation of a Cartesian grid incompressible Navier-Stokes solver on multi-GPU desktop platforms using CUDA*. PhD thesis, Boise State University, 2009.
- [38] Sivan Toledo, Doron Chen, and Vladimir Rotkin. Taucs: A library of sparse linear solvers. Technical report, School of Computer Science, Tel-Aviv University, 2003.

- [39] 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.
- [40] Manolis Vavalis. Implementing hybrid PDE solvers, 08 2014.