ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Υλοποίηση και Ανάλυση Τεχνικών Μέτρησης Ενεργά Χρησιμοποιούμενης Μνήμης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Άννας Γαλανού

Επιβλέπων: Γεώργιος Γκούμας
Επίκουρος Καθηγητής

Αθήνα, Αύγουστος 2019

Εθνικο Μετσοβιο Πολυτεχνειο
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Υλοποίηση και Ανάλυση Τεχνικών Μέτρησης Ενεργά Χρησιμοποιούμενης Μνήμης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Άννας Γαλανού

Επιβλέπων: Γεώργιος Γκούμας
Επίκουρος Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 1η Αυγούστου 2019.

.....	.....	.....
Γεώργιος Γκούμας	Νεκτάριος Κοζύρης	Νικόλαος Παπασπύρου
Επ. Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π

Αθήνα, Αύγουστος 2019.

.....

Άννα Γαλανού
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η μνήμη αποτελεί ένα από τα βασικότερα και πλέον κοστοβόρα στοιχεία που δομούν ένα σύστημα υπολογιστών και συνεπώς οι σύγχρονες τεχνικές διαχείρισής της τείνουν να ακολουθούν το μοτίβο του overcommiting για την καλύτερη αξιοποίησή της. Αυτόν το σκοπό εξυπηρετεί και η εικονικοποίηση με την οποία τα σύγχρονα datacenters καταφέρνουν και παρέχουν ελαστική διαχείριση πόρων στους χρήστες μέσω των εικονικών μηχανών. Συχνό φαινόμενο στα σύγχρονα συστήματα του cloud αποτελεί η σπατάλη μνήμης, η οποία αν και εικονικά φαίνεται να είναι απεριόριστη στην ουσία είναι πεπερασμένη και δεν υπάρχει περιθώριο για αναποτελεσματική χρήση της. Συνεπώς τίθεται το ζήτημα της ακριβής μέτρησης μνήμης που πραγματικά χρειάζεται κάθε εικονική μηχανή, που στην ουσία αποτελεί το working set της.

Στην παρούσα διπλωματική εργασία, μελετάμε λεπτομερώς κάποιες από τις τεχνικές που έχουν προταθεί στην βιβλιογραφία για τη μέτρηση του working set των εφαρμογών. Αρχικά, εστιάζουμε στην λεγόμενη idle page tracking τεχνική. Η συγκεκριμένη τεχνική βασίζεται στο idle page tracking API που προστέθηκε στο σύστημα Linux από την έκδοση πυρήνα 4.3 και μετά, και επιτρέπει στον χρήστη να εντοπίσει ποιες και πόσες σελίδες μνήμης έχουν προσπελαστεί πρόσφατα από τη διεργασία ή είναι αδρανείς (idle). Επιπλέον, υλοποιήσαμε δύο τεχνικές που βασίζονται σε δειγματοληψία και hardware performance counters και που σκοπό έχουν την μείωση του αντίκτυπου της μέτρησης του working set στην επίδοση των εφαρμογών. Στη συνέχεια, επικεντρωθήκαμε στην επέκταση του πυρήνα Linux, εγκαθιστώντας μια νέα διεπαφή για τη λήψη μετρήσεων του working set των εφαρμογών χρησιμοποιώντας τη λειτουργία του idle page tracking και μεθόδους δειγματοληψίας για την καλύτερη απόδοση του μηχανισμού. Εκμεταλλευόμενοι τη συγκεκριμένη επέκταση, τη συνδυάζουμε με τις μελετούμενες τεχνικές μέτρησης του working set. Αξιολογήσαμε τις τεχνικές σε περιβάλλον Linux εκτελώντας πειράματα με τα μετροπρογράμματα SPEC CPU2017. Επιπλέον, εκτελέσαμε κάποια ενδεικτικά πειράματα και σε εικονικό περιβάλλον Linux χρησιμοποιώντας το QEMU/KVM ως hypervisor. Τα πειραματικά αποτελέσματα μας δείχνουν πως οι προταθείσες τεχνικές είναι κατάλληλες για το σκοπό μας, καθώς μειώνουν το κόστος παρακολούθησης της μνήμης χωρίς απώλεια ακρίβειας στις μετρήσεις.

Λέξεις-Κλειδιά: διαχείριση μνήμης, βελτιστοποίηση κατανάλωσης ενέργειας, εικονικοποίηση, ενεργά χρησιμοποιούμενη μνήμη, idle page tracking, μετρητές απόδοσης υλικού

Abstract

Memory makes up a big part of the total cost of equipping a data center and as a result, data-center operators try to make the best use of memory they can, generally overcommitting it significantly. To this end, numerous datacenters are relying on virtualization, as it provides flexible resource management means such as virtual machine (VM) checkpoint/restart, migration and consolidation. However, one of the main hindrances to server consolidation is physical memory. In nowadays cloud, memory is generally statically allocated to VMs and wasted if not used. Techniques (such as ballooning) were introduced for dynamically reclaiming memory from VMs, such that only the needed memory is provisioned to each VM. However, the challenge is to precisely monitor the needed memory, i.e., the working set of each VM.

In this diploma thesis, we thoroughly review some of the main techniques that were proposed for monitoring the working set of processes, among which is idle page tracking. This is a feature that allows us to track which memory pages are being accessed by a workload and which are idle. Additionally, we have implemented the main techniques in a native Linux 4.9.110 environment and we have evaluated their efficiency using the SPEC CPU2017 benchmarks. We also ran some experiments in virtual Linux system 4.9.110 using QEMU/KVM as hypervisor.

We also propose an extension to the Linux kernel 4.9.110 which leverages idle page tracking and sampling methods in order to provide an interface for taking measurements of the working set size metric. We exploit this extension to optimize the proposed techniques and then evaluate their performance using the same SPEC CPU2017 benchmarks. Finally, we prove that the techniques we implemented are suitable for monitoring the working set of processes, since they manage to lower the overhead of memory tracking without a significant loss of accuracy.

Keywords: memory management, energy consumption optimization, virtualization, working set, idle page tracking, hardware performance counters

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο Εργαστήριο Υπολογιστικών Συστημάτων της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου το ακαδημαϊκό έτος 2018-2019.

Αρχικά, θα ήθελα να ευχαριστήσω τον Επιβλέποντα Καθηγητή μου κ.Γεώργιο Γκούμα για τη δυνατότητα που μου έδωσε να αναλάβω τη συγκεκριμένη εργασία. Κατά τη διάρκεια της μελέτης αυτής, συνεργάστηκα άψογα με το Μεταδιδακτορικό Ερευνητή κ.Βασίλη Καρακώστα και του οφείλω ένα μεγάλο ευχαριστώ για την καθοδήγηση και υποστήριξή του καθ' όλη τη διάρκεια της συνεργασίας μας αλλά και για τις πολύτιμες συμβουλές του σχετικά με τα επόμενα ακαδημαϊκά μου βήματα. Ευχαριστώ πολύ επίσης το Μεταδιδακτορικό Ερευνητή κ.Κωνσταντίνο Νίκα και την Υποψήφιο Διδάκτορα κ.Χλόη Αλβέρτη για τη βοήθειά τους.

Ολοκληρώνοντας πλέον ένα σημαντικό κεφάλαιο της ζωής μου, θα ήθελα να εκφράσω τις ευχαριστίες μου στους καθηγητές μου και κυρίως στους κ.Νεκτάριο Κοζύρη και κ.Γεώργιο Γκούμα, των οποίων οι διαλέξεις μού μετέδωσαν μέρος του πηγαίου ενδιαφέροντός τους για το αντικείμενο των υπολογιστικών συστημάτων και στάθηκαν αφορμή για να εκπονήσω τη διπλωματική μου όπως και να συνεχίσω τις ακαδημαϊκές μου σπουδές στον τομέα αυτό.

Το υπέροχο ταξίδι όλων αυτών των χρόνων δε θα ήταν το ίδιο χωρίς την αγαπημένη μου φίλη, Νάνσυ, που δε σταμάτησε ποτέ να με στηρίζει και να είναι πλάι μου στις χαρούμενες αλλά και δυσάρεστες στιγμές και της χρωστάω ένα μεγάλο ευχαριστώ για την πολύτιμη φιλία της. Επίσης, θα ήθελα να ευχαριστήσω τους φίλους μου για όλες τις εμπειρίες που ζήσαμε μαζί αυτά τα χρόνια.

Τέλος, το μεγαλύτερο ευχαριστώ το αξίζει η οικογένεια μου για την απεριόριστη στήριξη, εμπιστοσύνη και πίστη τους σε μένα. Ιδιαίτερα θα ήθελα να εκφράσω την ευγνωμοσύνη μου στη μητέρα μου, για την αμέριστη φροντίδα της και την κατανόηση που μου έδειξε όλα αυτά τα χρόνια.

Άννα Γαλανού
Αθήνα, 1η Αυγούστου 2019

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Στο κεφάλαιο αυτό θα αναλυθούν τα κίνητρα που οδήγησαν στη πραγματοποίηση της μελέτης αυτής καθώς και τα διάφορα προβλήματα που συναντήθηκαν κατά την εκπόνησή της. Τέλος, αναφέρονται συνοπτικά όλα τα θέματα που θα αναλυθούν στη συνέχεια και γίνεται μία μικρής έκτασης παρουσίαση των συνεισφορών της μελέτης αυτής.

1.1 Κίνητρα

Στα σύγχρονα συστήματα υπολογιστών η μνήμη αποτελεί ένα από τα σημαντικότερα και πιο κοστοβόρα στοιχεία. Για την αποδοτικότερη εκτέλεση των εφαρμογών είναι σύνηθες μοτίβο να γίνεται overcommiting της μνήμης κατά μεγάλο βαθμό, τακτική που αν και αυξάνει τη συνολική χρήση της επιφέρει ένα σημαντικό πρόβλημα. Τα συστήματα πρέπει να διαχειριστούν τη λεγόμενη πίεση μνήμης (memory pressure) και ως εκ τούτου αναγκάζονται να ανακαλέσουν σελίδες από δευτερεύουσες μονάδες αποθήκευσης. Η απευθείας ανάκληση (direct reclaim), όπου η διεργασία που δεσμεύει τη μνήμη πρέπει να απελευθερώσει μνήμη που χρησιμοποιείται από άλλες διεργασίες, επιφέρει πολλές καθυστερήσεις και περιορίζει την απομόνωση μεταξύ χρηστών.

Overcommiting γίνεται και κατά την εικονικοποίηση με την οποία τα σύγχρονα datacenters καταφέρνουν και παρέχουν ελαστική διαχείριση πόρων στους χρήστες μέσω των εικονικών μηχανών. Συχνό φαινόμενο στα σύγχρονα συστήματα του cloud αποτελεί η σπατάλη μνήμης, η οποία αν και εικονικά φαίνεται να είναι απεριόριστη στην ουσία είναι πεπερασμένη, και δεν υπάρχει περιθώριο για αναποτελεσματική χρήση της. Μια ενδεχόμενη λύση στα προκαλούμενα προβλήματα θα ήταν η δέσμευση, για κάθε εικονικό μηχάνημα, μνήμης με μέγεθος ανάλογο των αναγκών τους για το εκάστοτε χρονικό διάστημα, ώστε να αποφευχθεί η πίεση μνήμης. Στην ουσία τίθεται το ζήτημα ακριβούς μέτρησης της ενεργά χρησιμοποιούμενης μνήμης καθενός από τα VMs , ή αλλιώς του working set τους.

1.2 Προσέγγιση και Συνεισφορά Μελέτης

Στα πλαίσια της παρούσας διπλωματικής εργασίας μελετάμε τις υπάρχουσες τεχνικές μέτρησης του working set εφαρμογών όπως και την επιβάρυνση που εισάγουν στην εκτέλεσή τους. Στις τεχνικές αυτές συγκαταλέγονται τόσο υπάρχοντα εργαλεία και λειτουργικότητες του συστήματος Linux όσο και μεθόδοι που έχουν προταθεί από άλλες μελέτες. Για την αξιολόγηση των τελευταίων κληθήκαμε να υλοποιήσουμε τις τεχνικές αυτές, όπως και να τις συνδυάσουμε με μηχανισμούς που προσφέρει το σύστημα, όπως είναι το idle page tracking. Για την αξιολόγηση της επιβάρυνσης των υλοποιηθέντων και υπάρχοντων μεθόδων μέτρησης χρησιμοποιούμε μια ομάδα από μετροπρογράμματα (benchmarks) που επιφέρουν διάφορα επίπεδα πίεσης μνήμης στις εφαρμογές. Επιπλέον, για να βελτιώσουμε την απόδοση των παραπάνω τεχνικών, υλοποιήσαμε μία επέκταση του κώδικα πυρήνα συστήματος Linux ώστε να ενσωματώνονται κάποιες από τις τεχνικές μέτρησης μεγέθους του working set και να εδραιωθεί ένα interface για την άμεση και εύχρηστη λήψη μετρήσεων.

1.3 Οργάνωση Μελέτης

Το Κεφάλαιο 2 προσφέρει επιπρόσθετες πληροφορίες όσον αφορά τη διαχείριση μνήμης στο σύστημα Linux, το μοντέλο της ενεργά χρησιμοποιούμενης μνήμης, την υλοποίησή του, τα εργαλεία μέτρησής της που παρέχονται από το Linux όπως και τις τεχνικές μέτρησης της.
Το Κεφάλαιο 3 παρουσιάζει τη μεθοδολογία που ακολουθήθηκε κατά την εκτέλεση των πειραμάτων, περιλαμβάνοντας την περιγραφή του πειραματικού περιβάλλοντος στο οποίο εκτελέστηκαν, των benchmarks που χρησιμοποίθηκαν και των μετρικών με τις οποίες αξιολογήθηκαν.
Το Κεφάλαιο 4 αφορά την ανάλυση των τεχνικών που υλοποιήθηκαν και χρησιμοποιήθηκαν κατά την εκτέλεση των πειραμάτων.
Το Κεφάλαιο 5 παρουσιάζει τα πειραματικά αποτελέσματα με τη βοήθεια των μετρικών αξιολόγησης που επιλέχθηκαν.
Το Κεφάλαιο 6 περιλαμβάνει εκτενή σχολιασμό των αποτελεσμάτων για την εξαγωγή χρήσιμων συμπερασμάτων.
Το Κεφάλαιο 7 παρουσιάζει περιπτώσεις χρήσης του μοντέλου της ενεργά χρησιμοποιούμενης μνήμης.
Το Κεφάλαιο 8 αφορά σχετικές μελέτες που έχουν γίνει με αντικείμενο το working set και τις τεχνικές μέτρησής του.
Το Κεφάλαιο 9 παραθέτει τα τελικά συμπεράσματα που προκύπτουν από τα αποτελέσματα των πειραμάτων που διεξήχθησαν.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

Στο κεφάλαιο αυτό θα αναλυθούν οι βασικές έννοιες που συνθέτουν τη διαχείριση μνήμης στο λειτουργικό σύστημα Linux, όπως και οι μηχανισμοί αρχιτεκτονικής υποστήριξης με τους οποίους αυτή υλοποιείται. Επιπλέον, θα παρουσιαστεί όλο το θεωρητικό υπόβαθρο σχετικά με την ενεργά χρησιμοποιούμενη μνήμη και τις υπάρχουσες τεχνικές μέτρησής της.

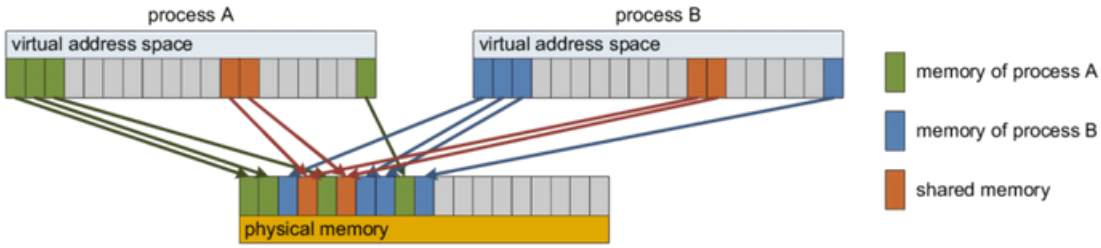
2.1 Διαχείριση μνήμης

Το υποσύστημα διαχείρισης μνήμης του Linux (Linux memory management subsystem) είναι υπεύθυνο για την υλοποίηση της εικονικής μνήμης (virtual memory), τη σελιδοποίηση (paging), τη δέσμευση μνήμης για δομές πυρήνα και προγραμμάτων χρήστη, την αντιστοίχιση αρχείων στο χώρο διεύθυνσεων των διεργασιών και για άλλες πολλές λειτουργίες. Είναι ένα πολύπλοκο σύστημα, το οποίο μπορεί να παραμετροποιηθεί με διάφορους τρόπους, αλλά κυρίως μέσω του `/proc` filesystem και κάνοντας χρήση της λειτουργικότητας sysctl.

2.1.1 Εικονική Μνήμη

Εικονική μνήμη είναι η τεχνική διαχείρισης μνήμης με την οποία αντιστοιχίζεται ο χώρος διευθύνσεων της φυσικής μνήμης που ανήκει σε συσκευές, όπως η RAM, PCI, GPU VRam κ.λ.π. σε έναν άλλο 'εικονικό' χώρο διευθύνσεων. Ως χώρος διευθύνσεων (address space) ορίζεται ένα σύνολο διακριτών διευθύνσεων που 'δείχνουν' σε κάποιο είδος μνήμης. Αυτή η τεχνική αποδίδει αρκετά πλεονεκτήματα, μερικά από τα οποία είναι:

- η απομόνωση μνήμης μεταξύ των ίδιων των διεργασιών αλλά και με τον πυρήνα,
- η ενσωμάτωση της μνήμης συσκευών στο χώρο διευθύνσεων μιας διεργασίας,



Σχήμα 2.1: Φυσική και Εικονική Μνήμη

- η δυνατότητα διαμοιρασμού μνήμης (shared memory),
- η δυνατότητα ορισμού δικαιωμάτων ανά περιοχή μνήμης,
- και η μετακίνηση σελίδων μνήμης (swap out) στο δίσκο ή στη μνήμη swap.

Η βασική ιδέα πίσω από την εικονική μνήμη είναι ότι κάθε πρόγραμμα έχει τον δικό του χώρο εικονικό διευθύνσεων. Γενικά υπάρχουν δύο είδη χώρου διευθύνσεων, οι φυσικές και οι εικονικές. Οι φυσικές διευθύνσεις είναι ουσιαστικά αυτές που χρησιμοποιούνται απευθείας από το hardware (DMA, περιφερειακά κ.λ.π.), ενώ οι εικονικές είναι οι διευθύνσεις που χρησιμοποιεί το software. Συγκεκριμένα για την αρχιτεκτονική που αναλύεται στην παρούσα εργασία (δηλαδή CISC – x86/AMD64), οι εικονικές διευθύνσεις είναι αυτές που χρησιμοποιούνται από τις εντολές assembly Load και Store. Οποιαδήποτε διεργασία τρέχει σε χώρο χρήστη (userspace) ″βλέπει″ εικονικές διευθύνσεις και όχι φυσικές. Το ίδιο ισχύει και για ένα μεγάλο μέρος του πυρήνα.

Η αντιστοίχηση φυσικών σε εικονικές διευθύνσεις γίνεται σε επίπεδο hardware. Με αυτόν τον τρόπο:

- δεν επηρεάζεται η επίδοση του συστήματος (performance overhead) όταν γίνεται προσπέλαση ήδη χαρτογραφημένης μνήμης,
- χρησιμοποιούνται οι ίδιες εντολές (assembly) τόσο για την προσπέλαση μνήμης όσο και συσκευών,
- και οι διεργασίες τόσο του userspace όσο και του πυρήνα χρησιμοποιούν εικονικές διευθύνσεις.

2.1.2 Σελιδοποίηση

Ο εικονικός χώρος διευθύνσεων διαιρείται σε μικρά κομμάτια που ονομάζονται σελίδες (pages). Κάθε σελίδα είναι ένα συνεχές εύρος διευθύνσεων. Οι σελίδες

της εικονικής μνήμης χαρτογραφούνται στην φυσική μνήμη, αλλά δεν χρειάζεται να βρίσκονται όλες οι σελίδες στη μνήμη για να εκτελεστεί ένα πρόγραμμα. Οι διευθύνσεις που δημιουργούνται από τα προγράμματα ονομάζονται εικονικές διευθύνσεις και συνθέτουν τον χώρο εικονικών διευθύνσεων. Σε υπολογιστές που δεν υποστηρίζουν εικονική μνήμη, οι διευθύνσεις που παράγουν τα προγράμματα τοποθετούνται απευθείας στο δίαυλο της μνήμης και προκαλούν την ανάγνωση ή την εγγραφή της λέξης στη φυσική μνήμη με την ίδια διεύθυνση. Σε υπολογιστές με εικονική μνήμη οι εικονικές διευθύνσεις δεν τοποθετούνται αμέσως στον δίαυλο αλλά τοποθετούνται στη Μονάδα Διαχείρισης Μνήμης (MMU, Memory Management Unit) η οποία αναλαμβάνει την χαρτογράφηση των εικονικών διευθύνσεων σε διευθύνσεις φυσικής μνήμης.

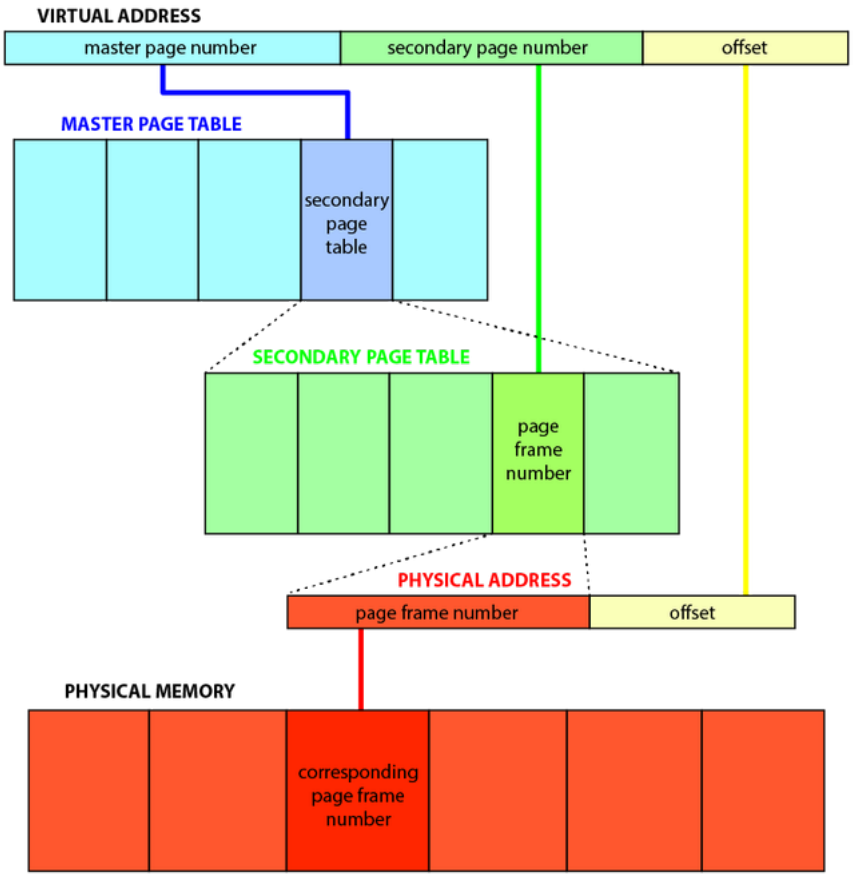
Σε αυτό το σημείο αξίζει να γίνει διάκριση μεταξύ εικονικής σελίδας (virtual page) και πλαισίου σελίδας (page frame). Virtual page είναι ένα συνεχές, σταθερού μήκους, μπλοκ πάνω στο χώρο διευθύνσεων της εικονικής μνήμης, το οποίο περιέχεται στον πίνακα σελίδων (page table) του συστήματος. Page frame είναι πάλι ένα συνεχές, σταθερού μήκους μπλοκ, αλλά πάνω στον φυσικό χώρο διευθύνσεων, δηλαδή πάνω στη φυσική μνήμη του συστήματος.

2.2 Μηχανισμοί Αρχιτεκτονικής Υποστήριξης

Η υλοποίηση της εικονικής μνήμης για να είναι ορθή και συνεπής απαιτεί την συνύπαρξη μηχανισμών τόσο λογισμικού όσο και υλικού. Το hardware που είναι υπεύθυνο για τη μνήμη του συστήματος συνοψίζεται κατά κύριο λόγο στα εξής τρία διακριτά μέρη.

MMU – Memory Management Unit Πρόκειται για το εξάρτημα που είναι υπεύθυνο για την υλοποίηση της εικονικής μνήμης. Συνήθως είναι ενσωματωμένο στον επεξεργαστή και τοποθετείται ανάμεσα στον πυρήνα του επεξεργαστή και την μνήμη. Χειρίζεται την προσπέλαση μνήμης των εντολών load/store με διαφανή τρόπο και αντιστοιχίζει τις προσπελάσεις που χρησιμοποιούν εικονικές διευθύνσεις σε διευθύνσεις φυσικής μνήμης RAM. Το ίδιο ισχύει και για τις συσκευές που έχουν αντιστοιχηθεί σε μνήμη (memory-mapped devices). Επιπλέον χειρίζεται τα δικαιώματα των χώρων μνήμης και δημιουργεί το Page Fault Exception στην περίπτωση λανθασμένης προσπέλασης (λόγω permissions ή αχαρτογράφητης διεύθυνσης).

Πίνακες Σελίδων Η λειτουργία της MMU υλοποιείται με την χρήση των πινάκων σελίδων (page tables). Σε μία απλή υλοποίηση, η χαρτογράφηση των εικονικών διευθύνσεων σε φυσικές διευθύνσεις μπορεί να συνοψιστεί ως εξής: η εισερχόμενη εικονική διεύθυνση διαιρείται σε έναν αριθμό εικονικής σελίδας (bit υψηλής τάξης) και μια σχετική διεύθυνση (bit χαμηλής τάξης). Ο αριθμός της εικονικής σελίδας



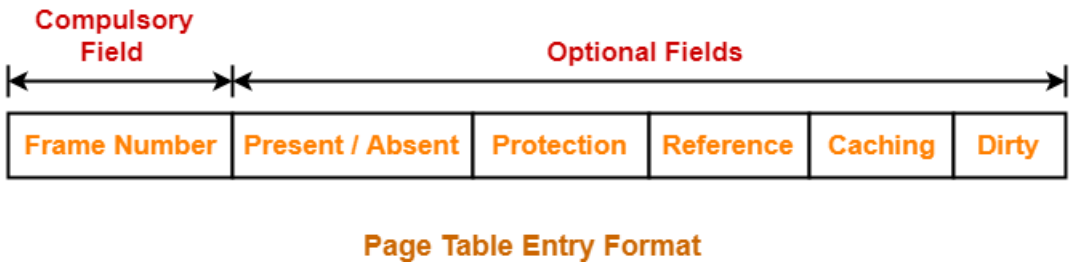
Σχήμα 2.2: Μετάφραση Εικονικών Διευθύνσεων σε Φυσικές

χρησιμοποιείται ως αριθμοδείκτης θέσης στον πίνακα σελίδων προκειμένου να εντοπίζεται η καταχώριση της συγκεκριμένης εικονικής σελίδας. Από την καταχώρηση του πίνακα σελίδων υπολογίζεται ο αριθμός του πλαισίου σελίδας (αν υπάρχει). Ο αριθμός πλαισίου σελίδας προσαρτάται στα αριστερά της σχετικής διεύθυνσης (πιο σημαντικά bit) αντικαθιστώντας τον αριθμό εικονικής σελίδας, ώστε να σχηματίσει μια φυσική διεύθυνση που μπορεί πλέον να σταλεί στη μνήμη. Από μαθηματική σκοπιά, ο πίνακας σελίδων είναι μια συνάρτηση, με όρισμα τον αριθμό εικονικής σελίδας και αποτέλεσμα τον αριθμό φυσικού πλαισίου.

Πιο αναλυτικά η ακριβής μορφή μίας καταχώρησης στον πίνακα σελίδων (page table entry) εξαρτάται από το υπολογιστικό σύστημα, ωστόσο το είδος των πληροφοριών που περιέχει σε γενικές γραμμές είναι το ίδιο. Το μέγεθος της καταχώρησης ποικίλει ανάλογα με τον υπολογιστή αλλά το πιο συνηθισμένο είναι τα 32bit. Το

σημαντικότερο πεδίο και αυτό που καταλαμβάνει και τον περισσότερο χώρο είναι ο αριθμός πλαισίου σελίδας (page frame number), εφόσον ο σκοπός της χαρτογράφησης είναι να ληφθεί αυτή η τιμή ως έξοδος. Το επόμενο από τα αριστερά bit είναι αυτό της παρουσίας/απουσίας (present bit), αν το συγκεκριμένο bit είναι 1 τότε η καταχώρηση είναι έγκυρη και μπορεί να χρησιμοποιηθεί. Αν είναι 0 τότε προκαλείται σφάλμα σελίδας καθώς η εικονική σελίδα στην οποία ανήκει αυτή η καταχώρηση δε βρίσκεται στη μνήμη την συγκεκριμένη χρονική στιγμή. Τα bits προστασίας (protection bits) δηλώνουν τα είδη πρόσβασης που επιτρέπονται. Συνήθως είναι 3 bits, καθένα από τα οποία επιτρέπει ή όχι (0 ή 1) την ανάγνωση, την εγγραφή και την εκτέλεση αντίστοιχα. Το bit αναφοράς (referenced bit) χρησιμοποιείται κατά την επιλογή από το λειτουργικό σύστημα της σελίδας που πρόκειται να αφαιρεθεί από τη μνήμη όταν συμβεί σφάλμα σελίδας καθώς παίρνει τιμή 1 κάθε φορά που γίνεται αναφορά στη συγκεκριμένη σελίδα, γεγονός που βοηθά στο να επιλέγονται σελίδες που δεν έχουν χρησιμοποιηθεί πρόσφατα. Το τελευταίο bit (caching enabled/disabled bit) επιτρέπει την απενεργοποίηση της κρυφής μνήμης για τη σελίδα αυτή, το οποίο είναι σημαντικό για σελίδες που χαρτογραφούνται σε καταχωρητές συσκευών και όχι στη μνήμη. Το bit τροποποίησης (modified bit) διατηρεί την πληροφορία για τη χρήση της σελίδας όπου χρησιμεύει κυρίως κατά την απόσυρση του πλαισίου από τη μνήμη και την εγγραφή του στο δίσκο.

Κάτι σημαντικό που πρέπει να διευκρινιστεί εδώ είναι ότι η διεύθυνση στο δίσκο



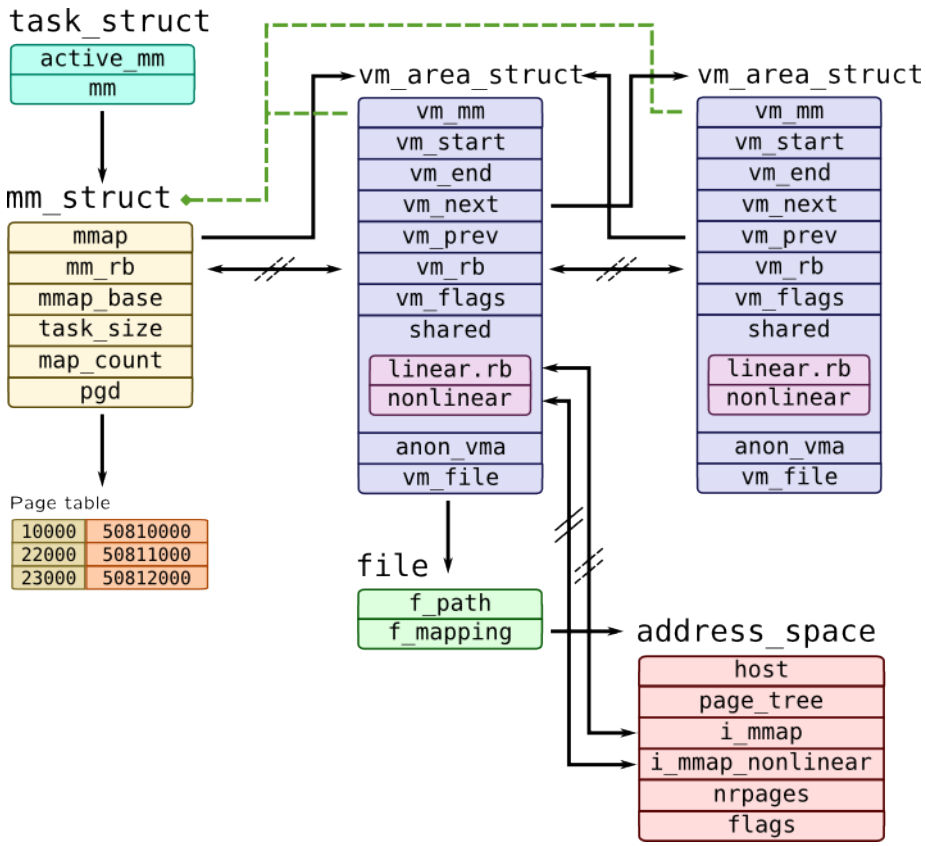
Σχήμα 2.3: Format του Page Table Entry

όπου αποθηκεύεται η σελίδα όταν δεν βρίσκεται στη μνήμη , δεν αποτελεί μέρος του πίνακα σελίδων καθώς ο πίνακας σελίδων διατηρεί μόνο πληροφορίες που χρειάζονται στο υλικό για τη μετάφραση μίας εικονικής διεύθυνσης σε φυσική, ενώ οι πληροφορίες που χρειάζονται στο λειτουργικό σύστημα για τη διαχείριση των σφαλμάτων σελίδας διατηρούνται σε πίνακες λογισμικού και είναι άχρηστες για το υλικό.

Ο πυρήνας σελίδων αποθηκεύει τις πληροφορίες των αντιστοιχίσεων αυτών στις δομές:

- struct_mm
- vm_area_struct

- task_struct



Σχήμα 2.4: Σχέσεις μεταξύ των Structs του Page Table

Memory Controller Το συγκεκριμένο εξάρτημα είναι υπεύθυνο για την άμεση επικοινωνία με τη μνήμη και κατ' επέκταση την περαιτέρω μετάφραση των διευθύνσεων (φυσικών) σε διευθύνσεις που έχουν νόημα για τις DRAM (bank, page, row, column bits)– ανάλογα φυσικά με το είδος της μνήμης. Θα πρέπει να είναι φυσικά συμβατό με τα πρωτόκολλα της DRAM και τις παραμέτρους timing (π.χ. DDR3/4/5 κ.λ.π.)

TLB – Translation Lookaside Buffer Προτού αναλύσουμε τη συγκεκριμένη μονάδα αξίζει να αναφερθεί πως σε οποιοδήποτε σύστημα σελιδοποίησης, υπάρχουν δύο σημαντικά ζητήματα που πρέπει να αντιμετωπιστούν:

- Η χαρτογράφηση από την εικονική διεύθυνση στη φυσική πρέπει να είναι γρήγορη.
- Αν ο χώρος των εικονικών διευθύνσεων είναι μεγάλος, ο πίνακας σελίδων μπορεί να είναι κι αυτός μεγάλος.

Η ανάγκη για ταχύτατη χαρτογράφηση μιας μεγάλης μνήμης περιορίζει τον τρόπο κατασκευής των Η/Υ. Ο πιο απλός σχεδιασμός εννοιολογικά, είναι να υπάρχει ένας μόνο πίνακας σελίδων ο οποίος να αποτελείται από έναν πίνακα από γρήγορους καταχωρητές υλικού και να χαρτογραφεί μια καταχώριση σε κάθε εικονική σελίδα, χρησιμοποιώντας ως αριθμοδείκτη θέσης τον αριθμό εικονικής σελίδας. Κατά την έναρξη μιας διεργασίας, το λειτουργικό σύστημα φορτώνει στους καταχωρητές τον πίνακα σελίδων της διεργασίας ο οποίος προέρχεται από ένα αντίγραφο που υπάρχει στην αρχική μνήμη. Κατά την εκτέλεση της διεργασίας ο πίνακας σελίδων δε χρειάζεται άλλες αναφορές στη μνήμη. Πλεονεκτήματα της παραπάνω μεθόδου είναι η απλότητα της και δεν απαιτείται αναφορά στη μνήμη κατά τη διάρκεια της χαρτογράφησης. Ως μειονέκτημα σημειώνεται πως είναι χρονοβόρα η παραπάνω μέθοδος αν ο πίνακας σελίδων είναι μεγάλος.

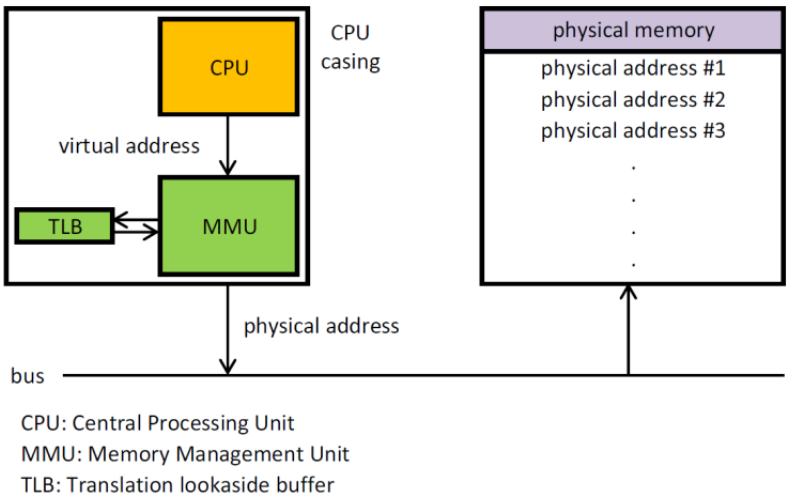
Το βασικό για τις περισσότερες τεχνικές επιτάχυνσης σελιδοποίησης είναι ότι ο πίνακας σελίδων βρίσκεται στη μνήμη. Ο σχεδιασμός αυτός είναι σημαντικός στην απόδοση. Οι σχεδιαστές των υπολογιστών γνωρίζουν το πρόβλημα αυτό και έχουν καταλήξει σε μια λύση. Η λύση βασίζεται στην παρατήρηση ότι τα περισσότερα προγράμματα συνήθως κάνουν πολλές αναφορές σε ελάχιστο αριθμό σελίδων και όχι το αντίστροφο. Επομένως μόνο ένα μικρό κλάσμα των καταχωρήσεων του πίνακα σελίδων διαβάζονται συχνά ενώ οι υπόλοιπες χρησιμοποιούνται ελάχιστα έως καθόλου. Η λύση ήταν να εξοπλιστούν οι υπολογιστές με μια μικρή συσκευή υλικού που χρησιμεύει στη χαρτογράφηση των εικονικών διευθύνσεων σε φυσικές χωρίς να χρησιμοποιεί τον πίνακα σελίδων. Η συσκευή ονομάζεται TLB (Translation Lookaside Buffer-Κρυφή μνήμη αναζήτησης μετάφρασης) ή συνειρμική ή συσχετιστική μνήμη (associative memory).

Όταν μια εικονική διεύθυνση μεταφέρεται στην MMU προκειμένου να μεταφραστεί, γίνεται έλεγχος αν ο αριθμός της εικονικής σελίδας υπάρχει στον TLB με παράλληλη σύγκριση όλων των καταχωρήσεων.

- Εφόσον υπάρχει και αφού η πρόσβαση δεν παραβιάζει τα bits προστασίας, το πλαίσιο σελίδας λαμβάνεται απευθείας από τον TLB χωρίς να χρησιμοποιηθεί ο πίνακας σελίδων.

- Αν η εικονική διεύθυνση δεν περιέχεται εντός του TLB τότε δημιουργείται ένα ″σφάλμα σελίδας″ (page fault) και θα διαδοθεί ένα interrupt στη CPU, η οποία θα παραδώσει τον έλεγχο στην αντίστοιχη interrupt routine που θα χειριστεί το εν λόγω σφάλμα.

Page Faults Το page fault είναι ένα exception σε επίπεδο επεξεργαστή, το οποίο δημιουργείται όταν έχουμε προσπέλαση ″μη ορθής″ εικονικής διεύθυνσης μνήμης. Μία διεύθυνση είναι invalid όχι μόνο όταν δεν υπάρχει στο χώρο διευθύνσεων, αλλά και στις παρακάτω περιπτώσεις:



Σχήμα 2.5: Αφαιρετικό διάγραμμα ΚΜΕ και μνήμης

- Η εικονική διεύθυνση δεν έχει αντιστοιχηθεί ακόμα στο χώρο διευθύνσεων της διεργασίας που προσπαθεί να την προσπελάσει.
- Η διεργασία δεν έχει δικαίωμα προσπέλασης της διεύθυνσης που ζητάει.
- Η διεύθυνση είναι έγκυρη αλλά έχει γίνει swap out. Το συγκεκριμένο fault είναι software fault (trap) που θα διαχειριστεί ο πυρήνας, σε αντίθεση με τα παραπάνω.

Σε περίπτωση page fault το λειτουργικό σύστημα θα πρέπει να:

- εντοπίσει τη θέση των ζητούμενων δεδομένων στο δίσκο,
- αποκτήσει ένα άδειο page frame στη RAM ώστε να αποθηκεύσει εκεί τα δεδομένα,
- φορτώσει τα ζητούμενα δεδομένα στο διαθέσιμο page frame,
- ανανεώσει τον πίνακα σελίδων ώστε να γίνεται αναφορά στο νέο page frame,
- επιστρέψει τον έλεγχο στο πρόγραμμα και να ξαναπροσπαθήσει με διαφανή τρόπο να εκτελέσει την εντολή που οδήγησε στο page fault.

Όταν όλα τα page frames χρησιμοποιούνται, τότε το λειτουργικό σύστημα θα πρέπει να διαλέξει ένα από αυτά για επαναχρησιμοποίηση ώστε να φορτώσει τη σελίδα που χρειάζεται το πρόγραμμα. Εάν το page frame που αποσύρθηκε ήταν δεσμευμένο δυναμικά από το πρόγραμμα για να διατηρεί δεδομένα ή εάν το πρόγραμμα το τροποποίησε αφότου το διάβασε από τη RAM, τότε θα πρέπει πρώτα να εγγραφεί στο δίσκο προτού απελευθερωθεί. Εάν το πρόγραμμα προσπαθήσει να προσπελάσει τη σελίδα που αποσύρθηκε, θα προκύψει και άλλο page fault ώστε η συγκεκριμένη σελίδα να

φορτωθεί πάλι στη RAM.

Η μέθοδος που χρησιμοποιεί το λειτουργικό σύστημα για να διαλέξει ποιο page frame να επιλέγεται, υλοποιείται από τον αλγόριθμο αντικατάστασης σελίδων (page replacement algorithm) και είναι πολύ σημαντική για την αποδοτικότητα του συστήματος. Το λειτουργικό σύστημα προβλέπει ποιο page frame είναι λιγότερο πιθανό να χρειαστεί σύντομα, κυρίως μέσω του αλγορίθμου LRU ή μέσω αλγορίθμων που βασίζονται στη μετρική της ενεργά χρησιμοποιούμενης μνήμης. Για να αυξήσουν την ανταποκρισιμότητα, τα συστήματα σελιδοποίησης μπορεί να προβλέψουν ποιες σελίδες θα χρειαστούν σύντομα και να τις φορτώσουν προληπτικά στη RAM προτού τις προσπελάσει το πρόγραμμα.

2.3 Ενεργά Χρησιμοποιούμενη Μνήμη (Working Set)

Ο όρος ενεργά χρησιμοποιούμενη μνήμη ή αλλιώς working set περιγράφει τη μνήμη που χρειάζεται μια διεργασία για να λειτουργήσει σε ένα δεδομένο χρονικό διάστημα. Ο Peter Denning [1] ορίζει working set $W(t, \tau)$ μιας διεργασίας τη χρονική στιγμή t ως το σύνολο πληροφοριών στις οποία αποκτά πρόσβαση η διεργασία κατά τη διάρκεια του χρονικού διαστήματος $(t - \tau, t)$. Τυπικά μονάδα μέτρησης της εν λόγω πληροφορίας θεωρούνται οι σελίδες μνήμης. Η συγκεκριμένη μετρική είναι στην ουσία εκτίμηση του συνόλου των σελίδων στις οποίες θα αποκτήσει πρόσβαση η διεργασία στο μέλλον (έστω τις επόμενες τ χρονικές στιγμές), και πιο συγκεκριμένα προτείνεται ως ένδειξη για τις σελίδες που πρέπει να διατηρηθούν στην κύρια μνήμη, ώστε να επιτραπεί η μεγαλύτερη δυνατή πρόοδος της εκτέλεσης της διεργασίας.

2.4 Σημασία

Η επιλογή των σελίδων που θα κρατηθούν στην κύρια μνήμη, σε αντίθεση με αυτές που θα μεταφερθούν (page out) στη βοηθητική, έχει μεγάλη σημασία. Αν κρατηθούν πολλές σελίδες μιας διεργασίας στην κύρια μνήμη, τότε λιγότερες διεργασίες θα είναι σε ετοιμότητα την εκάστοτε χρονική στιγμή. Εάν κρατηθούν, αντίθετα, λίγες σελίδες μιας διεργασίας, τότε η συχνότητα των page faults θα αυξηθεί και ο αριθμός των ενεργών διεργασιών που εκτελούνται την παρούσα χρονική στιγμή στο σύστημα θα πλησιάσει το μηδέν.

Το μοντέλο του working set αναφέρει πως μία διεργασία μπορεί να είναι στη RAM, αν και μόνο αν όλες οι σελίδες που χρησιμοποιεί την εκάστοτε χρονική στιγμή χωράνε εκεί. Εάν οι σελίδες που χρειάζεται αυξηθούν και δεν υπάρχει άλλος διαθέσιμος χώρος στη RAM, τότε επιστρατεύεται η δευτερεύουσα μνήμη στην οποία μεταφέρονται

οι σελίδες της διεργασίας (swap out) για να απελευθερωθεί χώρος για τις υπόλοιπες ενεργές διεργασίες.

Πολλές φορές ένα σύστημα με βαρύ υπολογιστικό φορτίο έχει τόσες πολλές διεργασίες σε λίστα αναμονής, που αν μπορούσαν όλες να διεκδικήσουν ένα μερίδιο της χρονοδρομολόγησης, τότε οι σελίδες που θα χρειάζονταν θα υπερβούσαν τη συνολική RAM, προκαλώντας το λεγόμενο thrashing στον υπολογιστή. Ο όρος αυτός περιγράφει την κατάσταση κατά την οποία ο υπολογιστής αδυνατεί να εκτελέσει το μεγαλύτερο ποσοστό των διεργασιών χρήστη λόγω υπερβολικού αριθμού page faults και συνεχόμενων φαινομένων paging, που εντέλει οδηγούν στην υπολειτουργία ή την κατάρρευσή του. Ως paging ορίζεται το πλάνο διαχείρισης μνήμης του υπολογιστή, κατά το οποίο αποθηκεύεται και ανακτάται πληροφορία από τη δευτερεύουσα μνήμη με σκοπό να χρησιμοποιηθεί από την κύρια.

Η μεταφορά κάποιων από τις ενεργές διεργασίες στη δευτερεύουσα μνήμη από την κύρια, συντελεί στη γρηγορότερη ολοκλήρωσή τους, ακόμα και σε σύγκριση με την περίπτωση που το σύστημα επιχειρούσε να τις εκτελέσει όλες μαζί. Επιπλέον, οι διεργασίες εκτελούνται γρηγορότερα απ΄ ότι θα ολοκληρώνονταν αν το σύστημα εκτελούσε μία από αυτές κάθε φορά, καθώς επιτρέπει στις άλλες διεργασίες να κάνουν πρόοδο σε περιόδους που η χρονοδρομολογημένη διεργασία αναμένει δεδομένα από το σκληρό δίσκο ή άλλες εξωτερικές μονάδες μνήμης.

Συνεπώς, η στρατηγική του working set αποτρέπει το thrashing ενώ επιτρέπει τον πολυγραμματισμό στο μέγιστο βαθμό που είναι δυνατόν, βελτιστοποιώντας έτσι τη χρήση της επεξεργαστικής δύναμης και την απόδοση του υπολογιστή.

2.5 Υλοποίηση Μοντέλου

Η κυριότερη δυσκολία στην υλοποίηση του μοντέλου working set είναι η εκτίμηση του μεγέθους του μέσα στο χρόνο. Το παράθυρο μέτρησης του working set είναι ένα ολισθούμενο παράθυρο. Σε κάθε πρόσβαση μνήμης, μία νέα πρόσβαση προστίθεται στο παράθυρο και μια παλιά φεύγει από αυτό. Μια σελίδα ανήκει στο working set όταν η πρόσβαση της γίνεται εντός του παραθύρου.

Το μοντέλο θα μπορούσε να υλοποιηθεί ως λίστα με δεδομένα τις τελευταίες k σελίδες στις οποίες απέκτησε πρόσβαση η διεργασία, αλλά κάτι τέτοιο θα προκαλούσε μεγάλη χρονική επιβάρυνση. Για την αποφυγή αυτού του κόστους η μέθοδος που συνήθως ακολουθείται είναι η διατήρηση της χρονικής στιγμής t κατά την οποία έγινε η πιο πρόσφατη πρόσβαση μνήμης και η εκτίμηση του working set ως το σύνολο των σελίδων, στις οποίες έγινε αναφορά εντός χρονικού διαστήματος $(t, t + \tau)$.

Το working set δεν είναι αλγόριθμος για την επιλογή σελίδων που θα μεταφερθούν στην κύρια μνήμη, αλλά τέτοιοι αλγόριθμοι μπορούν να σχεδιαστούν με βάση

αυτό ώστε να αφαιρούν σελίδες που δεν ανήκουν στο working set μιας διεργασίας. Χαρακτηριστικό παράδειγμα τέτοιου αλγόριθμου είναι η τροποποιημένη εκδοχή του λεγόμενου clock algorithm, WSClock, όπως τον όρισε ο Carr [2].

2.6 Παραλλαγές Μοντέλου

Το μοντέλο του working set μπορεί να εφαρμοστεί τόσο στον κώδικα όσο και στα δεδομένα. Η διαφοροποίηση είναι σημαντική, όταν ο κώδικας και τα δεδομένα είναι σε ξεχωριστά μέρη του σχετικού επιπέδου ιεραρχίας μνήμης, αφού αν δε χωράει κανένα από τα δύο στο συγκεκριμένο επίπεδο, τότε το σύστημα θα οδηγηθεί σε thrashing. Πέρα από τον κώδικα και τα δεδομένα, σε συστήματα με εικονική μνήμη, οι αντιστοιχίσεις των εικονικών διευθύνσεων σε φυσικές, των σελίδων του working set, πρέπει να υπάρχουν στον πίνακα TLB ώστε η διεργασία να εκτελεστεί αποδοτικά. Ο διαχωρισμός μεταξύ κώδικα και δεδομένων υπάρχει, γιατί μεταφέρονται στην cache κατά μικρά μπλοκ (cache lines), και όχι κατά σελίδες, αλλά η αναζήτηση των διευθύνσεων γίνεται σε επίπεδο σελίδας. Οπότε ακόμα και στην περίπτωση που τα working sets του κώδικα και των δεδομένων χωράνε στην cache, αν αυτά είναι χωρισμένα σε πολλές σελίδες, τότε το working set της εικονικής μνήμης δε θα χωράει στον TLB, προκαλώντας thrashing.

Ανάλογα μοντέλα του working set υπάρχουν για περιορισμένους άλλους πόρους, κυρίως διεργασίες. Αν ένα σύνολο διεργασιών απαιτεί τακτική αλληλεπίδραση μεταξύ πολλαπλών διεργασιών, τότε έχει ένα working set διεργασίας, το οποίο πρέπει να συνδρομολογηθεί ώστε να προχωρήσει αποδοτικά η εκτέλεση, όπως για παράδειγμα γίνεται στις παράλληλες εφαρμογές. Εάν οι εφαρμογές δεν δρομολογηθούν ταυτόχρονα, όπως π.χ θα γινόταν εάν υπήρχαν δύο διεργασίες και ένας πυρήνας να τις εκτελέσει, τότε οι διεργασίες μπορούν να επιτελέσουν μία αλληλεπίδραση ανά χρονική μονάδα.

Άλλα παραδείγματα αποτελούν οι χειριστές αρχείων και οι υποδοχές δικτύων. Για παράδειγμα η αντιγραφή ενός αρχείου σε ένα άλλο, γίνεται με τη χρήση δύο χειριστών, ενός για την είσοδο και ενός άλλου για την εξόδο, οπότε το working set του έχει μέγεθος 2. Εάν υπήρχε μόνο ένας διαθέσιμος χειριστής, τότε η αντιγραφή θα γινόταν αρχικά με τη λήψη του χειριστή για την είσοδο, το διάβασμα της εισόδου σε ένα προσωρινό buffer, την αποδέσμευση του χειριστή, τη λήψη του για την έξοδο, την εγγραφή των δεδομένων του buffer στην έξοδο, κ.ο.κ. Παρομοίως, ένας server μπορεί να χρειαστεί πολλές υποδοχές και αν αυτές είναι περιορισμένες τότε θα πρέπει συνεχώς να τις δεσμεύει και να τις αποδεσμεύει. Αν δε μπορούν να δεσμευτούν οι απαραίτητοι πόροι όσοι και το εκάστοτε working set, τότε το σύστημα θα οδηγηθεί σε thrashing.

2.7 Εργαλεία Μέτρησης Μνήμης στο Linux

Το λειτουργικό Linux δεν περιλαμβάνει κάποιο εργαλείο από το οποίο μπορεί να βγει απευθείας συμπέρασμα για το μέγεθος του working set μιας εφαρμογής. Αντίθετα, παρέχει άλλες μετρικές που αφορούν τη χρήση μνήμης είτε συγκεκριμένης διεργασίας είτε όλου του συστήματος και μπορούν να διαβαστούν από αρχεία όπως το `/proc/[pid]/smaps`, `/proc/meminfo` , αλλά και από εργαλεία όπως το vmstat, free, top. Τέτοιες μετρικές είναι η εικονική μνήμη, η λεγόμενη resident memory, που αναφέρεται στο σύνολο των σελίδων που έχουν αντιστοιχηθεί σε φυσικές και βρίσκονται στην κύρια μνήμη, και τέλος η αναλογική μνήμη ή αλλιώς proportional memory, που αφορά το μερίδιο της RAM που καταλαμβάνει η διεργασία για την ιδιωτική της μνήμη αλλά και για αυτή που μοιράζεται με άλλες διεργασίες.

Αυτές οι μετρικές είναι εύκολο να υπολογιστούν από τον πυρήνα, καθώς μέσω αυτού γίνεται η δέσμευση της εικονικής και φυσικής μνήμης. Αντίθετα, όταν η εφαρμογή τρέχει σε επίπεδο χρήστη και εκτελεί εντολές διαβάσματος και εγγραφής, ο πυρήνας δεν εμπλέκεται οπότε είναι δύσκολο να εξάγει κάποιο συμπέρασμα για το working set της.

2.8 Μονάδα Μέτρησης του Working Set

Ο λόγος χρήσης της μετρικής του working set στην ουσία υπαγορεύει και τη μονάδα μέτρησης που θα χρησιμοποιηθεί για την εκτίμησή της. Κάποια από τα πιθανά σενάρια είναι τα εξής:

- Μέτρηση της κύριας μνήμης μια διεργασίας με σκοπό να αποτραπεί το swapping. Το working set θα μετρηθεί σε bytes κατά τη διάρκεια ενός μεγάλου χρονικού διαστήματος, π.χ ενός λεπτού.

- Βελτιστοποίηση των caches του επεξεργαστή. Η μονάδα μέτρησης σε αυτή την περίπτωση θα είναι διακριτές cachelines και το χρονικό διάστημα στο οποίο θα γίνεται η μέτρηση θα είναι μικρότερο του ενός δευτερολέπτου. Το μέγεθος μιας cacheline εξαρτάται από την αρχιτεκτονική του επεξεργαστή και συνήθως είναι 64 bytes.

- Βελτιστοποίηση των TLB caches. Το working set στο συγκεκριμένο σενάριο θα εξυπηρετούσε να μετράται σε διακριτές σελίδες ανά χρονικά διαστήματα που διαρκούν λιγότερο του ενός δευτερολέπτου. Το μέγεθος της σελίδας εξαρτάται επίσης από την αρχιτεκτονική του επεξεργαστή όπως και από το λειτουργικό, και συνήθως είναι 4 KBytes.

2.9 Τεχνικές Μέτρησης Working Set

2.9.1 Swapping

Οι μετρικές που αφορούν το swapping είναι διαθέσιμες σε πολλά λειτουργικά συστήματα. Πολλά από αυτά που ανήκουν στην οικογένεια του Unix, όπως τα Linux και BSD, παρέχουν τέτοιες μετρικές μέσω του εργαλείου παρακολούθησης vmstat. Πέρα από αυτές υπάρχουν και μετρικές που παρέχονται από εργαλεία που παρακολουθούν τη χρήση μνήμης από το σύστημα και τη διαθεσιμότητά της. Τα συμπεράσματα που μπορούν να προκύψουν για το working set από το swapping ανάγονται στις εξής περιπτώσεις:

- Συνεχόμενο swapping: Το μέγεθος του working set είναι μεγαλύτερο από αυτό της κύριας μνήμης.

- Μηδαμινό swapping με συνεχόμενη παρακολούθηση μνήμης (scanning): Το working set είναι κοντά σε μέγεθος με την κύρια μνήμη.

- Μηδαμινό swapping και scanning: Το working set είναι συγκριτικά μικρότερο από την κύρια μνήμη.

Αξίζει να αναφερθεί πως εάν παρατηρηθεί συνεχόμενο swapping, μπορούν να εξαχθούν πιο αξιόπιστα συμπεράσματα σχετικά με τη μνήμη που χρησιμοποιεί η διεργασία, σε σύγκριση με αυτά που προκύπτουν από μετρικές όπως η εικονική μνήμη, η ενεργή/ανενεργή μνήμη και το resident set size (RSS). Αυτό συμβαίνει γιατί στην περίπτωση του swapping γνωρίζουμε πως το σύστημα χρησιμοποιεί ενεργά τη μνήμη τη συγκεκριμένη χρονική στιγμή για να εκτελέσει τις λειτουργίες του, ενώ στην περίπτωση π.χ του RSS δε μπορεί να διαπιστωθεί η ενεργή χρήση της μνήμης. Στο Linux θα πρέπει να υπάρχει μια μνήμη για swap, στην οποία θα μεταφέρονται οι σελίδες όταν δε χωράνε στην κύρια μνήμη, κάτι το οποίο δεν αποτελεί κανόνα για άλλα λειτουργικά συστήματα. Χωρίς μια τέτοια μνήμη, το πρόγραμμα του Linux, γνωστό ως Linux out-of-memory (OOM) killer, θα θυσιάζει, σε περιπτώσεις εξάντλησης μνήμης, κάποιες διεργασίες σκοτώνοντάς τες για να ελευθερώσει χώρο και δε μπορεί να παραχθεί κανένα συμπέρασμα για το μέγεθος του working set.

2.9.2 Performance Monitoring Counters

Οι μετρητές απόδοσης του Linux ή αλλιώς performance monitoring counters (PMCs) μπορούν να συνεισφέρουν στον υπολογισμό του μεγέθος του working set και παρέχονται από το εργαλείο perf. Παραδείγματα τέτοιων μετρητών είναι τα dTLB-load-misses, L1-dcache-load-misses και LLC-load-misses, από τους οποίους μπορούν να προκύψουν συμπεράσματα για το αν χωράει το working set στην κύρια μνήμη ή όχι και να γίνουν εκτιμήσεις για το μέγεθός του.

2.9.3 PTE Accessed Bit

Αυτός ο τρόπος μέτρησης του working set κάνει χρήση του accessed bit των καταχωρήσεων του πίνακα σελίδων (Page Table Entries), το οποίο ανανεώνεται από τη μονάδα διαχείρισης μνήμης (MMU) της CPU όταν γίνεται πρόσβαση σε κάποια σελίδα της μνήμης και μπορεί να διαβαστεί όπως και να ″καθαριστεί″ από τον πυρήνα. Τα βήματα για την εκτίμηση του WSS περιλαμβάνουν τον ″καθαρισμό″ των accessed bits των σελίδων της διεργασίας, την αναμονή για ένα χρονικό διάστημα και τέλος το διάβασμα των bits ώστε να μετρηθούν οι ″ενεργές″ σελίδες. Πλεονέκτημα αυτής της μεθόδου είναι η απουσία επιπλέον overhead από την ανανέωση των accessed bits, καθώς αυτό αποτελεί βήμα που κάνει έτσι και αλλιώς η MMU.

Linux Referenced Page Flag

Η συγκεκριμένη μέθοδος μέτρησης χρησιμοποιεί μια λειτουργικότητα του Linux που προστέθηκε από την έκδοση 2.6.22 και μετά, και επιτρέπει το διάβασμα και τον 'καθαρισμό' (reset) του referenced page flag των σελίδων από το χώρο χρήστη (user space) με σκοπό την ανάλυση της χρήσης μνήμης. Το referenced page flag στην ουσία αποτελεί το PTE accessed bit (_PAGE_BIT_ACCESSED), που προαναφέρθηκε.

Ο τρόπος που δουλεύει η μέθοδος είναι 'καθαρίζοντας' το referenced flag των σελίδων της διεργασίας και διαβάζοντας μετά από ένα χρονικό διάστημα πόσες από αυτές το είχαν ξανά σε θέση set. Το reset των επιθυμητών bits γίνεται γράφοντας στο αρχείο `/proc/[PID]/clear_refs` ενώ το διάβασμά τους μπορεί να ληφθεί από τη μετρική Referenced που παρέχεται από το `/proc/[PID]/smaps`. Αξίζει να σημειωθεί πως με αυτόν τον τρόπο το μέγεθος του working set στρογγυλοποιείται ώστε να αποτελεί πολλαπλάσιο του μεγέθους σελίδας (π.χ 4Kb), οπότε στην ουσία η τιμή του αφορά το χειρότερο σενάριο. Στην περίπτωση που γίνεται χρήση των huge pages, το μέγεθος της σελίδας (π.χ 2Mb) θα μεγεθύνει το εκτιμώμενο μέγεθος του working set σε μεγάλο βαθμό.

Η χρήση των `/proc/[PID]/clear_refs` και `/proc/[PID]/smaps` μπορεί να προκαλέσει μεγάλη καθυστέρηση της εφαρμογής ειδικά για διεργασίες που δεσμεύουν μεγάλο κομμάτι μνήμης, καθώς ο πυρήνας θα πρέπει να διατρέξει το σύνολο των σελίδων μνήμης για έλεγχο των bits τους. Πέρα από το πιθανό overhead που προσθέτει η συγκεκριμένη μέθοδος, αξίζει να αναφερθεί πως η ανάμειξη του χρήστη στις τιμές του referenced flag μπορεί να οδηγήσει τον πυρήνα σε σύγχυση σε σχέση με ποιες σελίδες πρέπει να ανακαλέσει όταν είναι ενεργοποιημένη η επιλογή swapping. Άλλο ένα πρόβλημα που σχετίζεται με τη συγκεκριμένη μέθοδο, ειδικά όταν μελετούνται διεργασίες με μεγάλη μνήμη, αφορά το πόσες επιπλέον σελίδες έχουν 'μαρκαριστεί' ως ενεργές ενώ η διαδικασία του διαβάσματος των bits βρισκόταν σε εξέλιξη. Αυτό με τη σειρά του μπορεί να οδηγήσει σε ανακρίβεια της εκτίμησης του μεγέθους του working set τη συγκεκριμένη χρονική στιγμή της μέτρησης.

Linux Idle Page Flag

Το συγκεκριμένο flag προστέθηκε από την έκδοση πυρήνα Linux 4.3 και μετά, μαζί με το young page flag για πιο αξιόπιστη εκτίμηση του μεγέθους του working set και αποφυγή ανάμειξης με το μηχανισμό ανάκλησης μνήμης που παρατηρούνταν με το reference flag. Η συγκεκριμένη μέθοδος ανήκει και αυτή στη γενικότερη κατηγορία προσέγγισης μέσω PTE accessed-bit, καθώς τα idle και young page flags αποτελούν επέκταση των καταχωρήσεων του πίνακα σελίδων (page_ ext_ flags) και βοηθούν στη λογική ανάκλησης σελίδων.

Η διεπαφή χρήστη με το συγκεκριμένο flag γίνεται μέσω του idle page tracking API που βρίσκεται στο μονοπάτι `/sys/kernel/mm/page_idle` και περιλαμβάνει το αρχείο ανάγνωσης-εγγραφής `/sys/kernel/mm/page_idle/bitmap`. Θα πρέπει να σημειωθεί πως το συγκεκριμένο μονοπάτι υπάρχει μόνο αν έχει ενεργοποιηθεί η επιλογή CONFIG_IDLE_PAGE_TRACKING στο configuration file του πυρήνα. Το παραπάνω αρχείο υλοποιεί ένα bitmap, έναν πίνακα δηλαδή από bits, καθένα από τα οποία αντιστοιχεί σε μία σελίδα μνήμης. Το bitmap απεικονίζεται ως ένας πίνακας από ακεραίους των 8 bytes και κάθε σελίδα με page frame number (PFN) #i αντιστοιχίζεται στο i%64 bit του στοιχείου του πίνακα με θέση #i/64. Στην περίπτωση που το bit έχει την τιμή 1, προκύπτει πως η σελίδα στην οποία αντιστοιχίζεται είναι 'ανενεργή' (idle), ενώ αν έχει την τιμή 0 τότε αυτή είναι 'ενεργή'.

Μια σελίδα θεωρείται idle όταν δεν έχει χρησιμοποιηθεί ξανά από τη διεργασία από τη χρονική στιγμή που μαρκαρίστηκε ως idle. Για να μαρκαριστεί μια σελίδα ως idle θα πρέπει να εγγραφεί η τιμή 1 στο bit, που αντιστοιχίζεται στο PFN της, κάτι το οποίο γίνεται αντίστοιχα με την εγγραφή στο αρχείο bitmap. Η τιμή που εγγράφεται εντέλει στο bit είναι αποτέλεσμα της λογικής πράξης OR μεταξύ της τιμής που γράφει ο χρήστης και της προϋπάρχουσας του bitmap.

Θα πρέπει να σημειωθεί πως με το συγκεκριμένο εργαλείο παρακολουθούνται μόνο οι προσβάσεις στις σελίδες μνήμης των διεργασιών χρήστη, καθώς αυτές έχουν αντιστοιχηθεί στο χώρο διευθύνσεων της διεργασίας, στην page cache, swap cache αλλά και στους buffers. Για σελίδες άλλου τύπου (π.χ SLAB pages) η απόπειρα να μαρκαριστεί μια σελίδα ως idle αγνοείται από τον πυρήνα και συνεπώς τέτοιες σελίδες δε μπορούν να παρακολουθηθούν. Στην περίπτωση των huge pages το idle flag γίνεται set μόνο στην κεφαλίδα της σελίδας, οπότε για τη μέτρησή τους θα πρέπει να διαβαστούν τα αντίστοιχα bits από το `/proc/kpageflags`.

Η κλήση των συναρτήσεων read ή write για το αρχείο `/sys/kernel/mm/page_idle/bitmap` επιστρέφει τιμή -EINVAL εάν η θέση του δείκτη από την οποία ξεκινάει η ανάγνωση/εγγραφή ή το μέγεθος της τιμής που διαβάζεται/εγγράφεται δεν είναι πολλαπλάσιο των 8 bytes. Επιπλέον, η απόπειρα εγγραφής σε bits του bitmap που αντιστοιχούν σε PFNs πέρα του max PFN θα επιστρέψει -ENXIO.

Για τον υπολογισμό των σελίδων που δεν ανήκουν στο working set μιας διεργασίας ακολουθούνται τα παρακάτω βήματα:

1. Μαρκάρισμα όλων των σελίδων της διεργασίας ως idle, γράφοντας την τιμή 1 στα αντίστοιχα bits του `/sys/kernel/mm/page_idle/bitmap`. Οι ζητούμενες σελίδες μπορούν να βρεθούν διαβάζοντας το αρχείο `/proc/[PID]/pagemap`.

2. Αναμονή για ένα χρονικό διάστημα ώστε η διεργασία να αποκτήσει πρόσβαση στο working set της.

3. Διάβασμα των κατάλληλων bits από το `/sys/kernel/mm/page_idle/bitmap` και μέτρηση των idle σελίδων.

Λεπτομέρειες υλοποίησης idle page tracking Ο πυρήνας παρακολουθεί τις προσβάσεις που γίνονται στις σελίδες μνήμης των διεργασιών χρήστη, ώστε να ανακαλέσει αυτές που δεν έχουν χρησιμοποιηθεί πρόσφατα σε περιπτώσεις εξάντλησης μνήμης. Μία σελίδα θεωρείται 'ενεργή' (referenced), εάν έχει χρησιμοποιηθεί μέσω του χώρου διευθύνσεων μιας διεργασίας. Στη συγκεκριμένη περίπτωση ένα ή περισσότερα PTEs στα οποία θα έχει αντιστοιχηθεί θα έχουν το accessed bit σε κατάσταση set ή θα έχουν μαρκαριστεί ειδικά από τον πυρήνα (μέσω της mark_page_accessed()). Το δεύτερο σενάριο συναντάται όταν:

- μια διεργασία χρήστη διαβάζει ή γράφει μια σελίδα μέσω system call (read(2), write(2))

- μια σελίδα, που έχει δεσμευθεί για τους buffers του συστήματος αρχείων, διαβάζεται ή γράφεται επειδή η διεργασία χρειάζεται τα metadata που είναι αποθηκευμένα σε αυτήν

- μια σελίδα χρησιμοποιείται από έναν driver συσκευής μέσω της συνάρτησης get_user_pages()

Αξίζει να σημειωθεί επίσης πως όταν μία dirty σελίδα μεταφέρεται στην swap μνήμη ή στον δίσκο λόγω του μηχανισμού ανάκλησης μνήμης ή λόγω υπέρβασης του ορίου dirty σελίδων, τότε αυτή δε μαρκάρεται ως referenced.

Το idle flag που προστέθηκε λόγω της νέας λειτουργικότητας idle page tracking, όπως προαναφέρθηκε, 'καθαρίζεται' αυτόματα κάθε φορά που μία σελίδα γίνεται referenced όπως ορίστηκε παραπάνω. Όταν μία σελίδα μαρκάρεται ως idle, τότε το accessed bit πρέπει να 'καθαριστεί' από όλα τα PTEs στα οποία έχει αντιστοιχηθεί, διαφορετικά δε θα μπορούν να εντοπιστούν περαιτέρω προσβάσεις σε αυτήν από το χώρο διευθύνσεων της διεργασίας. Για να αποφευχθεί οποιαδήποτε ανάμειξη με το μηχανισμό ανάκλησης μνήμης γίνεται χρήση ενός ακόμα flag, του young flag. Κάθε φορά που 'καθαρίζεται' το PTE accessed bit λόγω της ανανέωσης της τιμής του idle flag, το young flag τίθεται σε κατάσταση set. Ο μηχανισμός ανάκλησης μνήμης το

χρησιμοποιεί στην ουσία σαν ένα επιπλέον PTE accessed bit και συνεπώς μια τέτοια σελίδα θα θεωρηθεί ως referenced. Επειδή το εργαλείο του idle memory tracking βασίζεται στον παραπάνω μηχανισμό, λαμβάνονται υπόψη μόνο οι σελίδες που βρίσκονται στη λίστα LRU, ενώ οι υπόλοιπες αγνοούνται. Παραδείγματα τέτοιων σελίδων αποτελούν αυτές που είναι απομονωμένες και ας αποτελούν μέρος της μνήμης διεργασίας χρήστη, όπως και αυτές που είναι κλειδωμένες ώστε να αποφευχθεί οποιαδήποτε καθυστέρηση στο διάβασμα του bitmap. Συνήθως η ολιγωρία όσον αφορά τέτοιες σελίδες δεν οδηγεί σε μεγάλες αποκλίσεις από το πραγματικό μέγεθος του working set.

Κεφάλαιο 3

Μεθοδολογία

Στο κεφάλαιο αυτό θα παρουσιαστούν εκτενώς τα χαρακτηριστικά του υπολογιστικού συστήματος στο οποίο εκτελέστηκαν τα διάφορα πειράματα όπως και τα διάφορα εργαλεία που χρησιμοποιήθηκαν για τη συλλογή των αποτελεσμάτων που χρειαζόμασταν. Τέλος θα γίνει έκθεση των διαφόρων μετροπρογραμμάτων (benchmarks) που χρησιμοποιήθηκαν στη μέλετη αυτή, για την αξιολόγηση της απόδοσης των διαφόρων μεθόδων μέτρησης μεγέθους working set, και θα γίνει αναφορά στις μετρικές αξιολόγησης που χρησιμοποίηθηκαν.

3.1 Πειραματικό Περιβάλλον

Η μελέτη αυτή βασίζεται σε ένα τετραπύρηνο (4-core) Intel(R) Core(TM) i5-7600 (Kaby Lake), που τρέχει στα 3.5 GHz και είναι εξοπλισμένο με 16Gb μνήμη. Η διανομή που χρησιμοποιεί το συγκεκριμένο υπολογιστικό μηχάνημα είναι Debian, με έκδοση 8.10 (jessie) και η έκδοση του πυρήνα του Linux η οποία χρησιμοποιήθηκε και τροποποιήθηκε είναι η 4.9.110.

3.2 Benchmarks

Για να γίνει σωστή αξιολόγηση της απόδοσης των εργαλείων μέτρησης μεγέθους working set επιλέχθηκαν εφαρμογές από τη σουίτα SPEC CPU2017. Στη συνέχεια, παρατίθεται μία καταγραφή όλων των μετροπρογραμμάτων που χρησιμοποιήθηκαν καθώς και μία σύντομη περιγραφή του καθενός. Η σουίτα SPEC CPU2017 περιλαμβάνει 43 benchmarks που μπορούν να χωριστούν στις εξής 4 κατηγορίες:

- SPECrate 2017 Integer
- SPECrate 2017 Floating Point
- SPECspeed 2017 Integer
- SPECspeed 2017 Floating Point

Τα benchmarks που απεικονίζονται ως ζεύγη των:

5xx.benchmark_r / 6xx.benchmark_s

είναι παρόμοια μεταξύ τους. Οι διαφορές τους αφορούν τα compile flags, το μέγεθος του φορτίου και τους κανόνες εκτέλεσής τους.

SPECrate 2017 Integer	SPECspeed 2017 Integer	Language	Application
500.perlbench_r	600.perlbench_s	C	Μεταφραστής Perl
502.gcc_r	602.gcc_s	C	GNU C μεταγλωττιστής
505.mcf_r	605.mcf_s	C	Προγραμματισμός δρομολόγησης
520.omnetpp_r	620.omnetpp_s	C++	Προσομοίωση διακριτών γεγονότων
523.xalancbmk_r	623.xalancbmk_s	C++	Μετατροπή XML σε HTML μέσω XSLT
525.x264_r	625.x264_s	C	Συμπίεση video
531.deepsjeng_r	631.deepsjeng_s	C++	Τεχνητή Νοημοσύνη: Αλγόριθμος αναζήτησης δέντρου alpha-beta tree search (σκάκι)
541.leela_r	641.leela_s	C++	Τεχνητή Νοημοσύνη: Αλγόριθμος αναζήτησης δέντρου Monte Carlo (Γκο)
548.exchange2_r	648.exchange2_s	Fortran	Τεχνητή Νοημοσύνη: αναδρομική γεννήτρια λύσεων (Sudoku)
557.xz_r	657.xz_s	C	Συμπίεση γενικών δεδομένων

Πίνακας 3.1: SPECrate 2017 Integer & SPECspeed 2017 Integer benchmarks

SPECrate 2017 Floating Point	SPECspeed 2017 Floating Point	Language	Application
503.bwaves_r	603.bwaves_s	Fortran	Μοντελοποίηση έκρηξης
507.cactuBSSN_r	607.cactuBSSN_s	C++, C, Fortran	Φυσική: Σχετικότητα
508.namd_r		C++	Δυναμική σωματιδίων
510.parest_r		C++	Βιοϊατρική απεικόνιση: οπτική τομογραφία με πεπερασμένα στοιχεία
511.povray_r		C++, C	Ανίχνευση ακτίνων
519.lbm_r	619.lbm_s	C	Δυναμική ρευστών
521.wrf_r	621.wrf_s	Fortran, C	Πρόβλεψη καιρού
526.blender_r		C++, C	3D απεικόνιση και κινούμενα σχέδια
527.cam4_r	627.cam4_s	Fortran, C	Μοντελοποίηση ατμόσφαιρας
	628.pop2_s	Fortran, C	Μοντελοποίηση μεγάλης κλίμακας ωκεανών
538.imagick_r	638.imagick_s	C	Χειρισμός εικόνων
544.nab_r	644.nab_s	C	Δυναμική σωματιδίων
549.fotonik3d_r	649.fotonik3d_s	Fortran	Υπολογιστικός ηλεκτρομαγνητισμός
554.roms_r	654.roms_s	Fortran	Περιφερειακή μοντελοποίηση ωκεανών

Πίνακας 3.2: SPECrate 2017 Floating Point & SPECspeed 2017 Floating Point benchmarks

3.3 Μετρικές Αξιολόγησης

Για την αξιολόγηση των τεχνικών μέτρησης μεγέθους working set χρησιμοποιήθηκαν οι εξής μετρικές:

- Μετρική Απόδοσης: Αυτή αντικατοπτρίζεται στο overhead που εισάγει κάθε τεχνική στην εκτέλεση της εφαρμογής και υπολογίζεται ως:

  $$\%overhead = \frac{time_{mon} - time_{or}}{time_{or}} \cdot 100\%,$$

  όπου $time_{mon}$ ο χρόνος εκτέλεσης της εφαρμογής όταν συνεκτελείται με το monitoring tool και $time_{or}$ ο χρόνος εκτέλεσης της εφαρμογής όταν αυτή δεν παρακολουθείται από κανένα εργαλείο.

- Μετρική Ακρίβειας: Με αυτή την μετρική αξιολογούμε πόσο αντιπροσωπευτικό είναι το μέγεθος του working set που προσμετρήθηκε με τη χρήση της εκάστοτε μεθόδου σε σχέση με το πραγματικό της εφαρμογής.

- Μετρική Πλήθους Μετρήσεων: Με αυτή την μετρική αξιολογούμε εν μέρει πάλι την ακρίβεια του εκτιμώμενου μεγέθους του working set. Όσο περισσότερες μετρήσεις προσλαμβάνουμε στο χρόνο εκτέλεσης της εφαρμογής τόσο πιο αντιπροσωπευτικές είναι οι εκτιμήσεις μας για το wss της.

- Μετρική Χρόνου Απενεργοποίησης Monitoring Tool: Η συγκεκριμένη τεχνική απεικονίζεται μέσω του ποσοστού μείωσης των μετρήσεων που λάβαμε χρησιμοποιώντας το εκάστοτε εργαλείο μέτρησης wss και τη λαμβάνουμε υπόψη μας για την αξιολόγηση των αποτελεσμάτων της μεθόδου Intermittent Page Tracking.

Κεφάλαιο 4

Ανάλυση Τεχνικών

Στην ενότητα αυτή θα αναλύσουμε τις τεχνικές που χρησιμοποιήσαμε για τη μέτρηση του μεγέθους του working set των εφαρμογών που αναφέραμε στο Κεφάλαιο 2.

4.1 Idle Page Tracking

Στη συγκεκριμένη τεχνική χρησιμοποιούμε τα δύο εργαλεία, όπως τα εισήγαγε ο Brendan Gregg [24], που εκτιμούν το WSS βάση του idle page tracking.

wss-proc: βελτιστοποιημένο για μικρές διεργασίες Η συγκεκριμένη εκδοχή του εργαλείου διατρέχει τις δομές των σελίδων μία μία και κάνει set/reset τα συγκεκριμένα bits που τους αντιστοιχούν στο bitmap. Για διεργασίες με μεγάλο WSS, η συγκεκριμένη διαδικασία μπορεί να πάρει αρκετά λεπτά μέχρι να ολοκληρώσει τη μέτρηση την εκάστοτε χρονική στιγμή, γι΄ αυτό προτείνεται για διεργασίες με μικρό working set. Επιπλέον, επειδή διαβάζει και τροποποιεί page flags καταναλώνει επεξεργαστικό χρόνο, οπότε προκαλεί κάποιο βαθμό καθυστέρησης στην εφαρμογή.

wss-sys: βελτιστοποιημένο για μεγάλες διεργασίες Στη συγκεκριμένη εκδοχή του εργαλείου, λαμβάνεται ένα αντίγραφο όλου του bitmap, που αντιστοιχεί στο σύνολο των διεργασιών που τρέχουν στο σύστημα και όχι μόνο σε αυτή που μας ενδιαφέρει, στο userspace. Επιπλέον κατά τη διαδικασία του 'καθαρισμού' που προηγείται του διαβάσματος των idle page flags, γίνεται reset των bits σε όλο το bitmap. Η διαφοροποίηση αυτή σε σχέση με το προηγούμενο εργαλείο έχει ως αποτέλεσμα να επιταχύνεται η λήψη μετρήσεων για διεργασίες με μεγάλο working set, αφού δεν καταναλώνεται χρόνος στα αλλεπάλληλα syscalls για read και write στα bits που μας ενδιαφέρουν.

4.2 Referenced Page Tracking

Το συγκεκριμένο εργαλείο [24] αρχικά κάνει reset τα referenced flags των σελίδων της διεργασίας και έπειτα τα ξαναελέγχει για να δει ποιες προσπελάστηκαν στο τρέχον διάστημα. Το reset γίνεται γράφοντας την τιμή 1 στο αρχείο `/proc/[PID]/clear_refs` και η λήψη της εκτιμήσης για το WSS γίνεται διαβάζοντας το μέγεθος Referenced από το αρχείο `/proc/[PID]/smaps`. Το συγκεκριμένο εργαλείο λαμβάνει μετρήσεις πολύ γρηγορότερα σε σχέση με τα προηγούμενα, ωστόσο μπορεί να οδηγήσει τον πυρήνα σε σύγχυση επειδή τροποποιεί τα referenced flags που χρησιμοποιούνται για το σύστημα ανάκλησης σελίδων μνήμης.

4.3 Sampling

Στη συγκεκριμένη μέθοδο βασιζόμενοι στην ιδέα που προτάθηκε από τον Waldspurger [3] δεν ελέγχουμε εξαντλητικά όλες τις σελίδες της διεργασίας και τα idle flags τους αλλά μόνο ένα δείγμα από αυτές. Ο αριθμός των δειγμάτων είναι είτε σταθερός είτε επιλέγεται αναλογικά με το συνολικό πλήθος των σελίδων της εφαρμογής. Έπειτα κάνουμε reset και διαβάζουμε μόνο τα flags του δείγματός μας, εξάγοντας το συνολικό wss της διεργασίας αναλογικά με βάση αυτό που μετρήσαμε για το δείγμα.

4.4 Intermittent Page Tracking

Η συγκεκριμένη τεχνική αποτελεί υλοποίηση της μεθόδου που σχεδίασαν οι Zhao et al. [13]. Η βασική ιδέα πίσω από αυτήν είναι η απενεργοποίηση του wss monitoring όταν το πρόγραμμα μπαίνει στη λεγόμενη 'σταθερή φάση', δηλαδή το working set του δε διαφοροποιείται κατά μεγάλο βαθμό, και η επανενεργοποίησή του όταν εκκινεί μια νέα φάση. Για να προβλεφθεί πότε εκκινεί νέα φάση working set ελέγχουμε συνεχώς κάποιους performance counters, μέσω του εργαλείου perf stat, και διατηρούμε ένα φίλτρο κινούμενου μέσου όρου για την αποθορυβοποίηση των τιμών που λαμβάνουμε. Όταν το φίλτρο γεμίσει με k δεδομένα, θεωρούμε v_j την τρέχουσα ληφθείσα τιμή, f_{mean} τον τρέχοντα κινούμενο μέσο όρος και $err_r = ((v_j - f_{mean})/f_{mean}) * 100$ τη σχετική διαφορά μεταξύ ληφθείσας τιμής και μέσου όρου. Εάν το err_r είναι εντός επιθυμητού εύρους που έχουμε ορίσει, τότε θεωρούμε πως το πρόγραμμα διανύει μια 'σταθερή' φάση. Διαφορετικά, θεωρούμε πως έχει εκκινήσει μια νέα φάση και φροντίζουμε να καθαριστεί το φίλτρου κινούμενου μέσου όρου, ώστε να μη συνυπολογίζονται πια τιμές της προηγούμενης φάσης. Παράλληλα με το φίλτρο που αφορά τους performance counters διατηρούμε και ένα που αφορά τις τιμές wss, ώστε να απενεργοποίησουμε με την ίδια λογική το monitoring όποτε θεωρούμε πως οι εκτιμήσεις έχουν σταθεροποιηθεί.

4.4.1 Performance Counters

Για την υλοποίηση της συγκεκριμένης τεχνικής χρησιμοποιήθηκαν οι hardware performance counters, dTLB-load-misses και instructions, ώστε να λαμβάνουμε την τιμή των dTLB-load-misses ανά χιλιάδα εντολών.

4.5 Kernel Extensions

Συγκρίνοντας τις τεχνικές που κάνουν χρήση των referenced και idle flags, προκύπτουν τα εξής συμπεράσματα:

- Η τεχνική του Referenced Page Tracking υστερεί σε σύγκριση με αυτήν του Idle Page Tracking σε ό,τι αφορά την επεμβατικότητα στα page flags, το οποίο όπως έχει προαναφερθεί μπορεί να προκαλέσει σύγχυση στον πυρήνα λόγω του συστήματος ανάκλησης σελίδων μνήμης κατά το swapping.

- Η έκδοση του Idle Page Tracking που είναι βελτιστοποιημένη για μικρές διεργασίες υστερεί σε σχέση με τις άλλες δύο σε ό,τι αφορά το πλήθος των μετρήσεων. Λόγω των πολλών syscalls που γίνεται με τις επαναλαμβανόμενες αναγνώσεις και εγγραφές στο bitmap καταναλώνεται πολύς χρόνος με αποτέλεσμα να καθυστερεί η λήψη μετρήσεων για το WSS.

- Η έκδοση του Idle Page Tracking που είναι βελτιστοποιημένη για μεγάλες διεργασίες υστερεί σε σχέση με τις άλλες δύο σε ό,τι αφορά την επεμβατικότητα σε page flags που αφορούν σελίδες και άλλων διεργασιών πέραν αυτής που αποτελεί αντικείμενο μετρήσεων. Πριν τη διαδικασία της ανάγνωσης των bits που αντιστοιχούν στις σελίδες της διεργασίας που παρακολουθούμε, γίνεται reset όλου του bitmap στο οποίο αντιστοιχούνται τα idle flags όλων των διεργασιών χρήστη του συστήματος.

Σύμφωνα με τα παραπάνω προκύπτει πως η βέλτιστη μέθοδος όσον αφορά το βαθμό επεμβατικότητας είναι αυτή που χρησιμοποιεί Idle Page Tracking και κάνει reset/set μόνο τα bits που αντιστοιχούν στις σελίδες της διεργασίας που είναι υπό παρακολούθηση. Για να είναι βέλτιστη και ως προς τις εκτιμήσεις του WSS που παρέχει καθόλη τη διάρκεια εκτέλεσης της διεργασίας, θα πρέπει να μειωθεί ο βαθμός καθυστέρησης στο στάδιο του reset του bitmap, που συνδέεται σε μεγάλο βαθμό όπως προαναφέρθηκε με το πλήθος των syscalls που γίνονται.

Με αφορμή το συγκεκριμένο πρόβλημα, δημιουργήθηκε η ανάγκη για επέκταση του κώδικα πυρήνα Linux ώστε να υποστηρίζονται κάποιες επιπλέον λειτουργείες και να βελτιστοποιηθεί η απόδοση του συγκεκριμένου εργαλείου. Παρακάτω αναλύουμε τις επεκτάσεις στον κώδικα όπως και την επέκταση του εικονικού συστήματος αρχείων `\proc` που θα αποτελεί τη διεπαφή για τις νέες λειτουργείες.

4.5.1 /proc Filesystem

Το `/proc` σύστημα αρχείων (filesystem) περιλαμβάνει μια ιεραρχία ειδικών αρχείων που αντιπροσωπεύουν την τρέχουσα κατάσταση του πυρήνα. Στην ουσία αποτελεί ένα interface για τις δομές των τρέχοντων διεργασιών ώστε να διευκολυνθεί η χρήση εργαλείων αποσφαλμάτωσης. Λόγω της αυξανόμενης πολυπλοκότητας του `/proc` filesystem το Linux δημιούργησε το sysfs filesystem για να απλοποιηθούν κάποιες διαδικασίες.

Τα αρχεία και οι υποφακέλοι του `/proc` περιέχουν πληροφορίες για το hardware και τις τρέχουσες διεργασίες του συστήματος. Πρόκειται στην ουσία για εικονικά αρχεία και για αυτό το `/proc` αναφέρεται ως εικονικό σύστημα αρχείων. Ένα subdirectory του είναι και το `/proc/[PID]` που περιλαμβάνει με τη σειρά του πολλούς ακόμα υποφακέλους και εικονικά αρχεία. Σε αυτό τον υποφάκελο προσθέσαμε δύο εικονικά αρχεία το `/proc/[PID]/clear_idle` και το `/proc/[PID]/wss`.

Όσον αφορά το `/proc/[PID]/clear_idle` η λειτουργία του είναι παρόμοια με αυτή του `/proc/[PID]/clear_refs`, αρχείο που αναλύθηκε παραπάνω στην ενότητα Referenced Page Tracking. Πιο συγκεκριμένα, γράφοντας σε αυτό το αρχείο γίνεται reset στα idle flags όλων των σελίδων που αντιστοιχούν στη διεργασία που βρίσκεται υπό παρακολούθηση. Όσον αφορά το `/proc/[PID]/wss` η λειτουργία του είναι παρόμοια με αυτή του `/proc/[PID]/smaps` με τη διαφορά ότι τυπώνεται μόνο η εκτίμηση για το WSS και όχι τα υπόλοιπα στατιστικά, ώστε να μην προστεθεί επιπλέον καθυστέρηση λόγω της λήψης τους.

Οι επιτρεπτές τιμές που μπορούν να γραφούν στο `/proc/[PID]/clear_idle` εξαρτώνται από την τεχνική που θα ακολουθηθεί κατά τη λήψη εκτιμήσεων για το μέγεθος του working set. Για να περιοριστεί κατά μεγαλύτερο βαθμό η καθυστέρηση που εισάγει η τεχνική Idle Page Tracking έχουμε προσθέσει στον πυρήνα τη δυνατότητα για δειγματοληψία που θα αναλυθεί παρακάτω.

Υλοποίηση /proc/[PID]/clear_idle

Η κύρια συνάρτηση στην οποία υλοποιείται το συγκεκριμένο interface είναι η clear_idle_write(). Αφού ελεγχθεί ότι η τιμή που έγραψε ο χρήστης στο αρχείο είναι αποδεκτή τότε εκκινεί η διάσχιση των διαστημάτων εικονικής μνήμης που αντιστοιχούν στο συγκεκριμένο mm_struct της διεργασίας, για να γίνει το reset των bits. Για τη λειτουργία αυτή εκτελούνται οι ίδιες συναρτήσεις που καλούνται στην page_idle_bitmap_write(). Η τελευταία καλείται όταν γράφουμε στο αρχείο `/sys/kernel/mm/page_idle/bitmap`. Το κοινό τους κομμάτι είναι το εξής:

Listing 4.1: Τμήμα κώδικα που υλοποιούν οι clear_idle_write και page_idle_bitmap_write

```
{
        ...
```

```
        page = page_idle_get_page(pfn);
        if (page) {
                page_idle_clear_pte_refs(page);
                set_page_idle(page);
                put_page(page);
        }
        ...
}
```

Αξίζει να αναφερθεί σε αυτό το σημείο πως η τεχνική του Idle Page Tracking λαμβάνει υπόψη μόνο τις σελίδες μνήμης χρήστη, και για τους υπόλοιπους τύπους σελίδων το idle flag είναι πάντα σε κατάσταση unset ή οποιαδήποτε προσπάθεια να γίνει set αγνοείται από τον πυρήνα. Θεωρούμε πως μια σελίδα ανήκει στη μνήμη χρήστη όταν ανήκει στη λίστα LRU, καθώς έτσι είναι ασφαλές να την 'περάσουμε' ως όρισμα στη συνάρτηση rmap_walk(), που είναι απαραίτητο για το idle page tracking. Έχοντας τη λίστα αυτή ως ένδειξη για τις σελίδες μνήμης χρήστη, καταλήγουμε να αγνοούμε τις απομονωμένες σελίδες μνήμης, αλλά επειδή αυτές στις περισσότερες περιπτώσεις είναι λίγες το αποτέλεσμα της εκτίμησης δε διαφέρει πολύ από το πραγματικό. Η page_idle_get_page() επιτελεί τον παραπάνω σκοπό, προσπαθεί δηλαδή να επιστρέψει μια σελίδα μνήμης χρήστη με βάση το pfn που λαμβάνει ως όρισμα.

Η συνάρτηση rmap_walk() που προαναφέραμε στην ουσία διασχίζει όλες τις αλυσιδωτές αντιστοιχίσεις της σελίδας της οποίας το δείκτη λαμβάνει ως όρισμα. Όταν αυτή καλείται από την try_to_munlock() ο σημαφόρος mmap_sem της δομής mm_struct, που περιλαμβάνει το διάστημα εικονικής μνήμης vma στο οποίο βρέθηκε η σελίδα, θα κρατηθεί για εγγραφή. Με αυτόν τον τρόπο δε θα χρειαστεί να ελεγχθούν ξανά τα vm_flags για το συγκεκριμένο vma.

Η συνάρτηση page_idle_clear_pte_refs() που εκτελείται στο παραπάνω τμήμα κώδικα καλεί την rmap_walk() σε συνδυασμό με τη συνάρτηση page_idle_clear_pte_refs_one() ώστε να διασχίσει τις αντιστοιχίσεις της σελίδας και να κάνει unset τα idle flags. Για τις σελίδες που είχαν το referenced bit set η συνάρτηση φροντίζει να κάνει set το young flag ώστε να μην προκαλέσει σύγχυση στο μηχανισμό ανάκλησης σελίδων.

Υλοποίηση /proc/[PID]/wss

Η συγκεκριμένη λειτουργικότητα υλοποιείται στη συνάρτηση show_wss() στην οποία ακολουθείται παρόμοια λογική με την show_smaps(). Πιο συγκεκριμένα, σε κατάσταση κανονικής λειτουργίας (όταν η δειγματοληψία είναι απενεργοποιημένη), καλείται η walk_page_vma() η οποία διατρέχει το εκάστοτε διάστημα εικονικής μνήμης, κρατώντας τον σημαφόρο mm-¿mmap_sem, και ελέγχει για κάθε σελίδα που συναντά έαν έχει set το young page flag ή unset το idle page flag. Εάν ισχύει μία από τις δύο αυτές συνθήκες τότε προσθέτει το μέγεθος της σελίδας αυτής στη μεταβλητή που κρατάει για το wss του συγκεκριμένου vma. Το τύπωμα των στατιστικών γίνεται και

σε αυτή την περίπτωση σαν αυτό του /proc/[PID]/smaps εμφανίζοντας τις εκτιμήσεις των wss για κάθε διάστημα εικονικής μνήμης που αντιστοιχεί στη διεργασία.

4.5.2 Sampling

Βασιζόμενοι στη προσέγγιση στατιστικής δειγματοληψίας που χρησιμοποιείται στον ESX Server για τη λήψη εκτιμήσεων WSS και προτάθηκε από τον Waldspurger [3], επεκτείναμε τον κώδικα του πυρήνα ώστε να υποστηρίζει δυνατότητα δειγματοληψίας (sampling). Υλοποιήσαμε δύο εκδοχές της συγκεκριμένης μεθόδου που αναλύονται παρακάτω.

Sampling με σταθερό αριθμό σελίδων

Στη συγκεκριμένη εκδοχή του sampling επιτρεπτές τιμές για το αρχείο `/proc/[PID]/clear_idle` θεωρούνται το 0 και το 1. Στην περίπτωση του 0 η λήψη εκτιμήσεων γίνεται συνυπολογίζοντας όλες τις σελίδες της διεργασίας, ενώ σε αυτή του 1 ενεργοποιείται η μέθοδος της δειγματοληψίας. Την πρώτη φορά που ο χρήστης θα γράψει 1 στο αρχείο `/proc/[PID]/clear_idle` θα επιλεχθούν και θα αποθηκευτούν συνολικά N σελίδες μνήμης από αυτές που αντιστοιχούν στη διεργασία. Όταν ξαναγραφεί η τιμή 1 στο συγκεκριμένο αρχείο ή όταν διαβαστεί το `/proc/[PID]/smaps` θα ελέγχονται και θα τροποποιούνται μόνο τα idle flags των N σελίδων που αποθηκεύσαμε.

Οι εικονικές διευθύνσεις των σελίδων που έχουν επιλεχθεί ως μέρος του δείγματος αποθηκεύονται σε λίστα, που είναι μέρος της δομής mm_struct, όπως έχει οριστεί στον τροποποιημένο κώδικα πυρήνα 4.9.110. Αρχικά βρίσκουμε το πλήθος δειγμάτων που αναλογεί σε κάθε διάστημα εικονικής μνήμης και έπειτα επιλέγουμε τον κατάλληλο αριθμό έγκυρων δειγμάτων για καθένα από αυτά. Έγκυρα θεωρούνται τα δείγματα των οποίων οι εικονικές διευθύνσεις ικανοποιούν τους κατάλληλους ελέγχους σελίδων. Στην περίπτωση που δεν έχει αποθηκευτεί κανένα έγκυρο δείγμα για κάποιο vma τότε αλλάζουμε τον ρυθμό με τον οποίο το διατρέχουμε μέχρι να βρούμε.

Sampling με συγκεκριμένο ρυθμό

Στη συγκεκριμένη εκδοχή του sampling επιτρεπτές τιμές για το αρχείο `/proc/[PID]/clear_idle` θεωρούνται όσες είναι πάνω από 1 και αφορούν το ρυθμό με τον οποίο θα διαβάζονται οι σελίδες σε κάθε διάστημα εικονικών διευθύνσεων. Συνεπώς για την τιμή 1 διαβάζονται όλες οι σελίδες και στην ουσία δε γίνεται δειγματοληψία. Σε αυτή την εκδοχή της μεθόδου η δομή struct page επεκτείνεται ώστε να περιλαμβάνει άλλο ένα πεδίο, το sampled, το οποίο θα διαχωρίζει τις σελίδες που είναι δείγματα από αυτές που δεν είναι. Την πρώτη φορά που ο χρήστης θα γράψει στο αρχείο `/proc/[PID]/clear_idle` τιμή πάνω από 1 θα διαβαστούν οι σελίδες σε καθέ διάστημα εικονικών διευθύνσεων με το επιθυμητό ρυθμό και θα επιλεχθούν οι δειγματοληπτημένες σελίδες, για τις οποίες το πεδίο sampled θα λάβει την τιμή 1, ενώ για

τις υπόλοιπες θα έχει την τιμή 0. Κατά το διάβασμα του αρχείου `/proc/[PID]/smaps` θα ληφθούν υπόψη μόνο αυτές που είναι μέρος του δείγματός μας και θα προκύψει η ανάλογη εκτίμηση για το WSS.

4.6 Sampling & Intermittent Page Tracking

Η συγκεκριμένη τεχνική συνδυάζει τις μεθόδους του sampling και του intermittent tracking. Πιο συγκεκριμένα το πρόγραμμα παρακολούθησης του working set της εφαρμογής εκτελεί δειγματοληψία γράφοντας στο αρχείο `/proc/[PID]/clear_idle` την τιμή 1 και διαβάζοντας από το `/proc/[PID]/smaps` το εκτιμώμενο μέγεθος WSS. Η λειτουργία του συγκεκριμένου διακόπτεται και επανεκκινεί ανάλογα με τη φάση που έχουμε θεωρήσει ότι βρισκόμαστε κρίνοντας από τους επιλεγμένους performance counters.

Κεφάλαιο 5

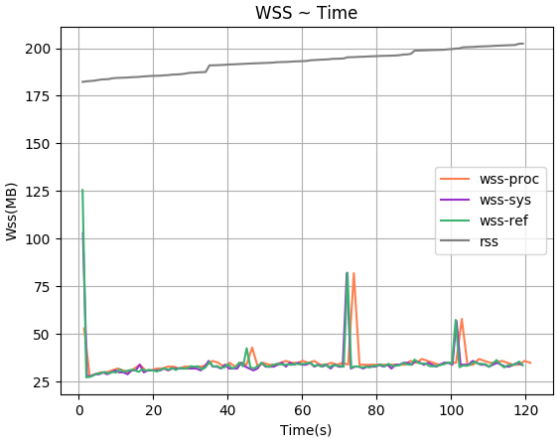
Μετρήσεις

Στο κεφάλαιο αυτό θα παρουσιαστούν τα αποτελέσματα των διάφορων τεχνικών μέτρησης του working set που αναλύθηκαν στο Κεφάλαιο 4.

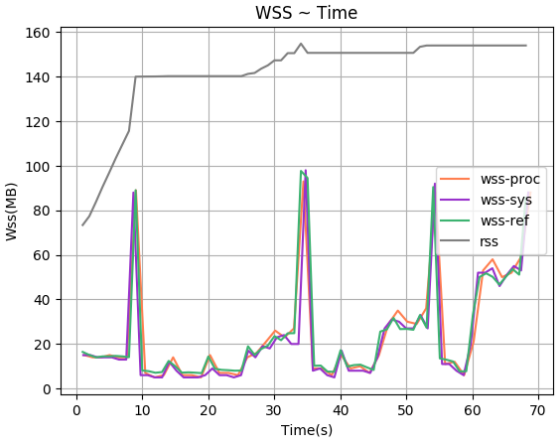
5.1 Μετρήσεις σε μη τροποποιημένο πυρήνα Linux

5.1.1 Idle & Referenced Page Tracking

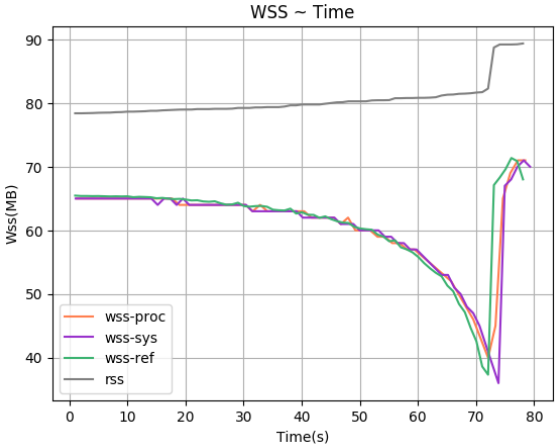
Αρχικά εκτελέσαμε τα πειράματα με τα benchmarks SPEC CPU2017 σε μη τροποποιημένο πυρήνα Linux 4.9.110 με τα υπάρχοντα εργαλεία μέτρησης working set με χρονικό διάστημα παρακολούθησης το 1 δευτερόλεπτο. Στα γραφήματα 5.1 - 5.62 μαζί με τις εκτιμήσεις των εργαλείων απεικονίζεται και η μέτρηση του μεγέθους της resident memory (rss) όπως λήφθηκε από το αρχείο `/proc/[PID]/smaps`. Το wss-ref αντιστοιχεί στο εργαλείο που εκτιμά το WSS βάση του referenced flag ενώ τα wss-proc, wss-sys στα εργαλεία που ελέγχουν τα idle flags και είναι βελτιστοποιημένα για μικρές και μεγάλες διεργασίες αντίστοιχα, όπως αναλύσαμε παραπάνω. Στα γραφήματα 5.63 και 5.64 απεικονίζονται τα overheads των benchmarks 5xx και 6xx αντίστοιχα, ενώ στα 5.65 και 5.66 απεικονίζεται το πλήθος των μετρήσεων που λάβαμε από τη συνεκτέλεσή τους με τα monitoring tools. Τέλος, εκτελέσαμε μετρήσεις με το εργαλείο wss-proc για διαστήματα παρακολούθησης 10s και 20s. Τα γραφήματα των working sets απεικονίζονται στα 5.67 - 5.93, ενώ τα overheads και το πλήθος μετρήσεων που παρατηρήθηκαν παρουσιάζονται στα 5.94 και 5.95.



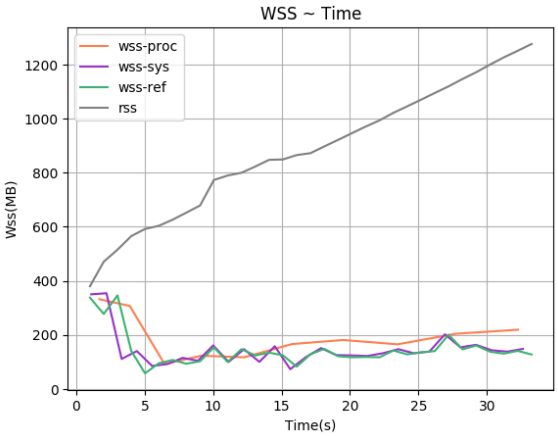
Σχήμα 5.1: 500.perlbench_r-ref-1[WSS, interval=1s, user level]



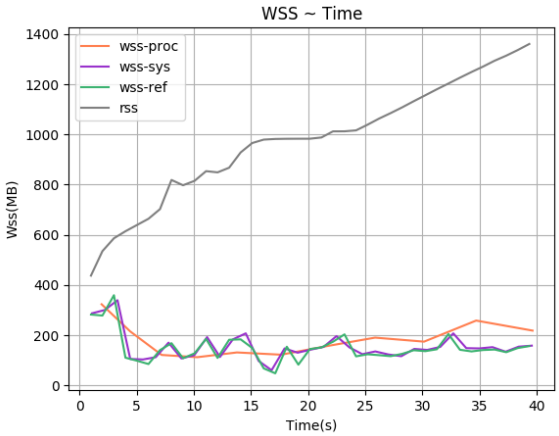
Σχήμα 5.2: 500.perlbench_r-ref-2[WSS, interval=1s, user level]



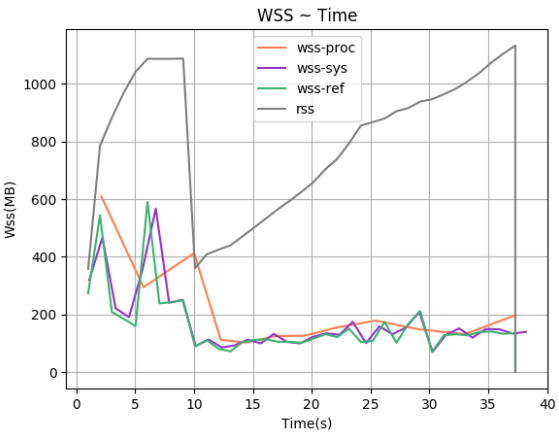
Σχήμα 5.3: 500.perlbench_r-ref-3[WSS, interval=1s, user level]



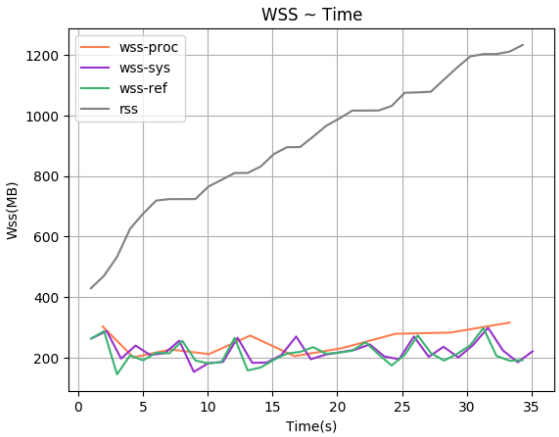
Σχήμα 5.4: 502.gcc_r-ref-1[WSS, interval=1s, user level]



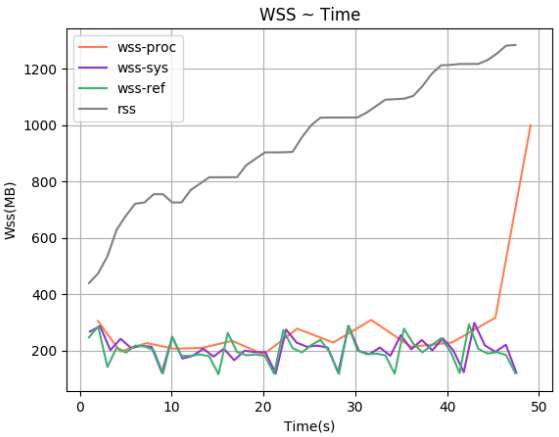
Σχήμα 5.5: 502.gcc_r-ref-2[WSS, interval=1s, user level]



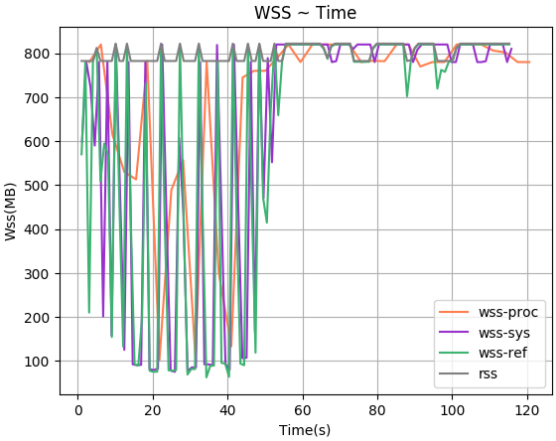
Σχήμα 5.6: 502.gcc_r-ref-3[WSS, interval=1s, user level]



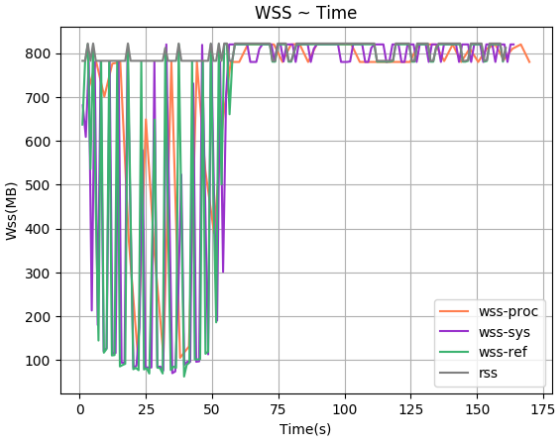
Σχήμα 5.7: 502.gcc_r-ref-4[WSS, interval=1s, user level]



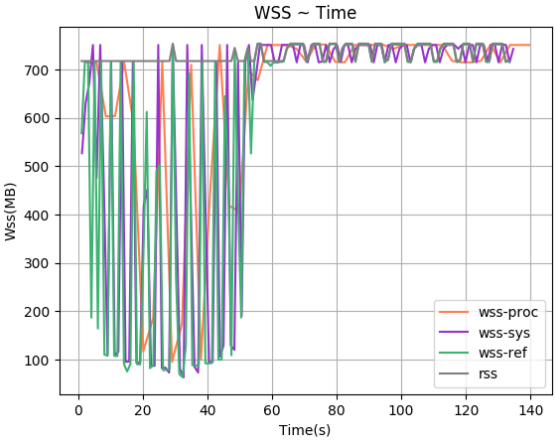
Σχήμα 5.8: 502.gcc_r-ref-5[WSS, interval=1s, user level]



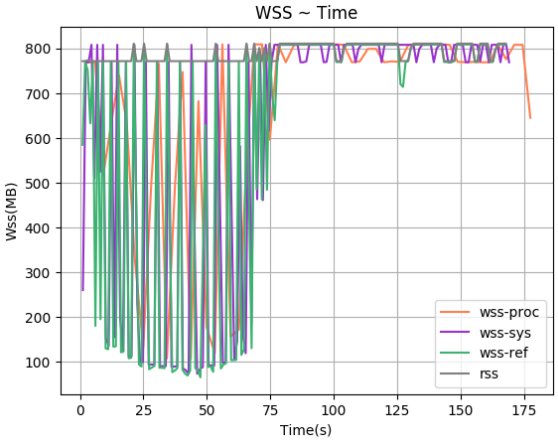
Σχήμα 5.9: 503.bwaves_r-ref-1[WSS, interval=1s, user level]



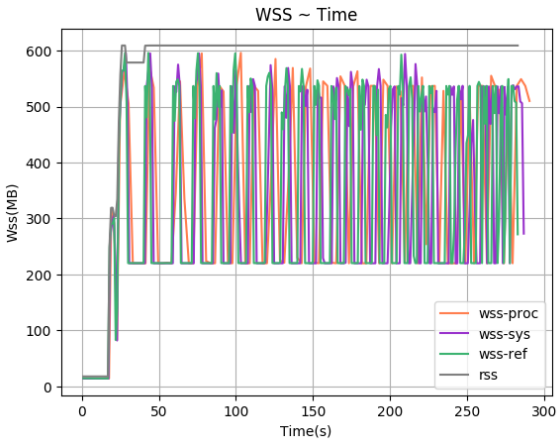
Σχήμα 5.10: 503.bwaves_r-ref-2[WSS, interval=1s, user level]



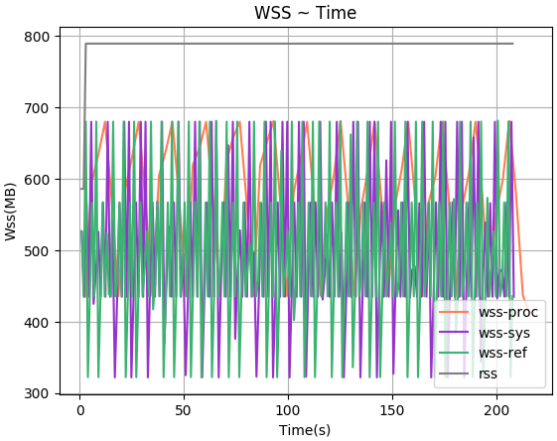
Σχήμα 5.11: 503.bwaves_r-ref-3[WSS, interval=1s, user level]



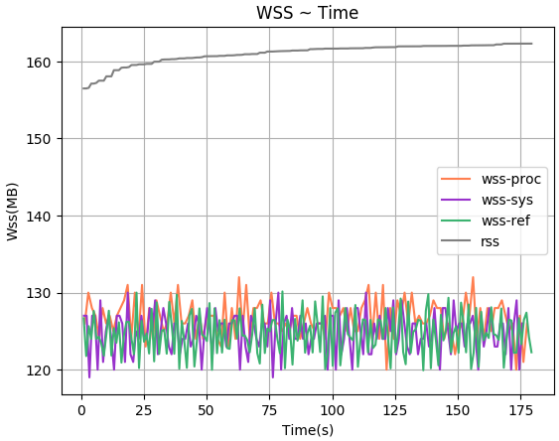
Σχήμα 5.12: 503.bwaves_r-ref-4[WSS, interval=1s, user level]



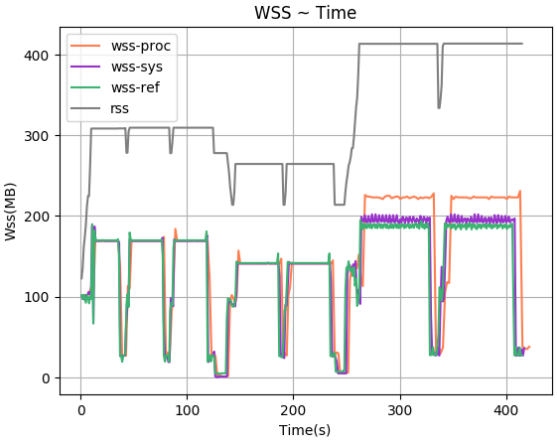
Σχήμα 5.13: 505.mcf_r-ref-1[WSS, interval=1s, user level]



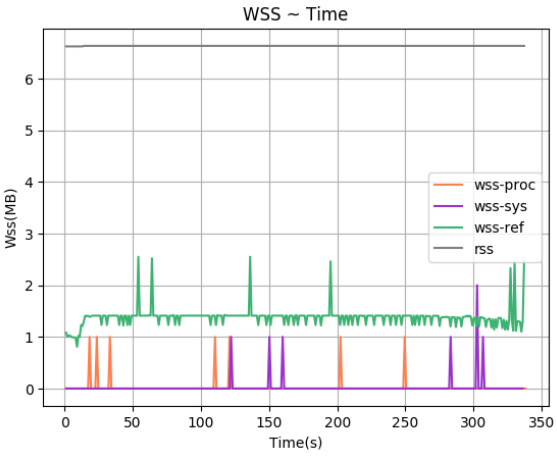
Σχήμα 5.14: 507.cactuBSSN_r-ref-1[WSS, interval=1s, user level]



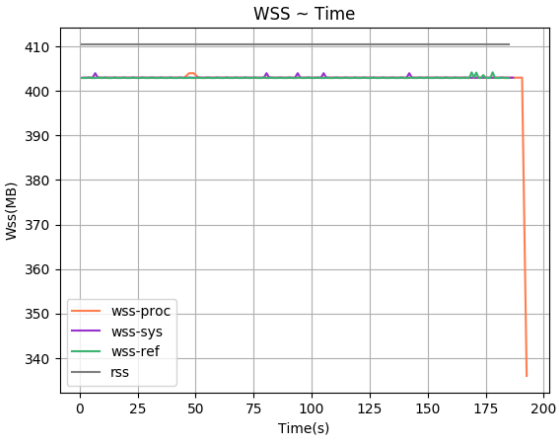
Σχήμα 5.15: 508.namd_r-ref-1[WSS, interval=1s, user level]



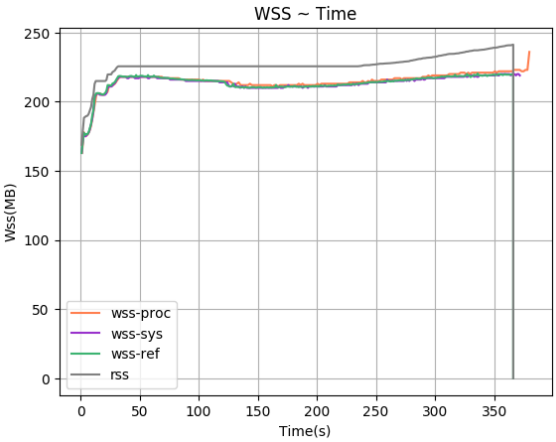
Σχήμα 5.16: 510.parest_r-ref-1[WSS, interval=1s, user level]



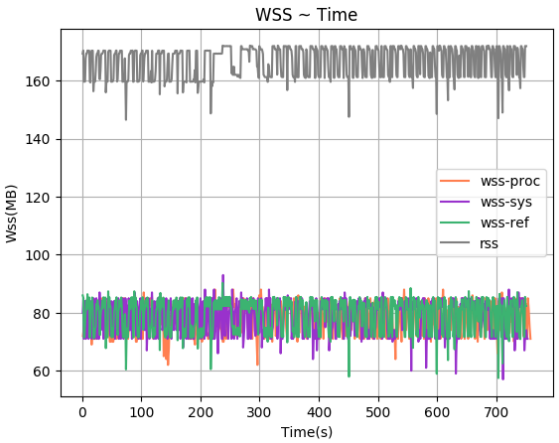
Σχήμα 5.17: 511.povray_r-ref-1[WSS, interval=1s, user level]



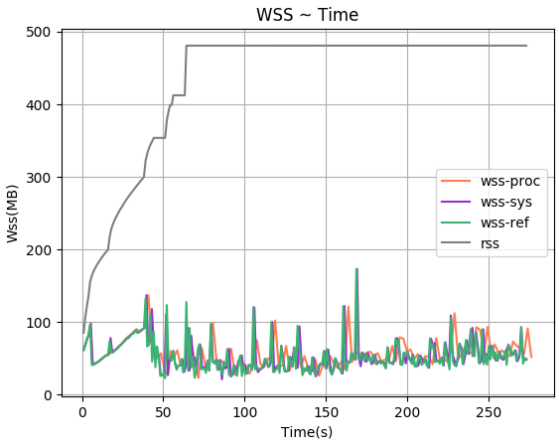
Σχήμα 5.18: 519.lbm_r-ref-1[WSS, interval=1s, user level]



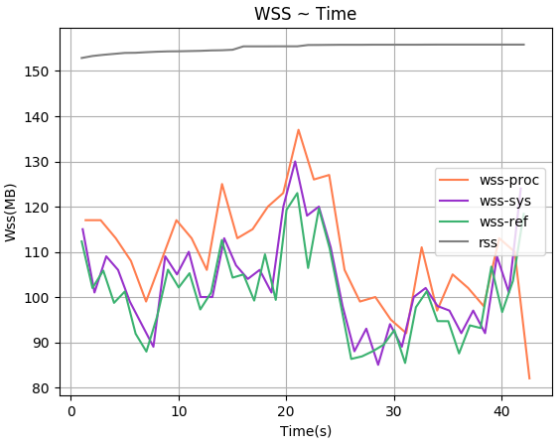
Σχήμα 5.19: 520.omnetpp_r-ref-1[WSS, interval=1s, user level]



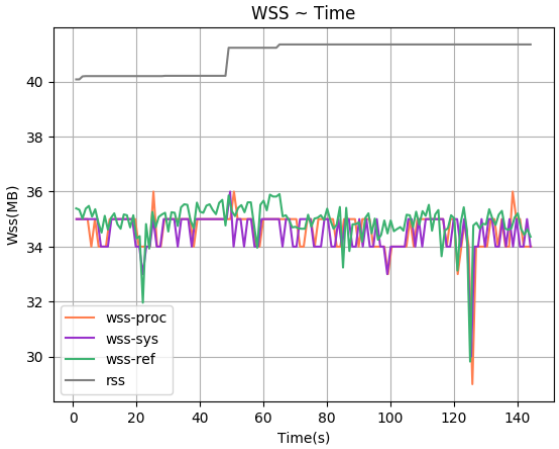
Σχήμα 5.20: 521.wrf_r-ref-1[WSS, interval=1s, user level]



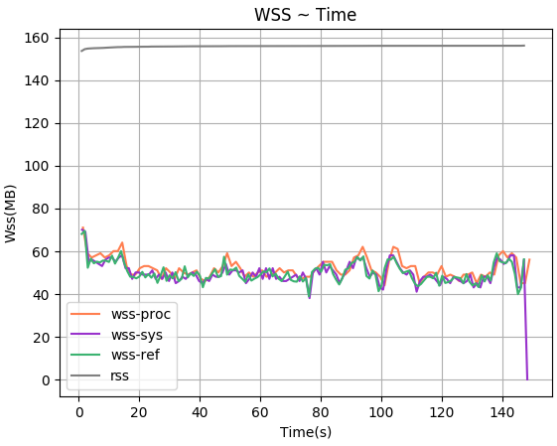
Σχήμα 5.21: 523.xalancbmk_r-ref-1[WSS, interval=1s, user level]



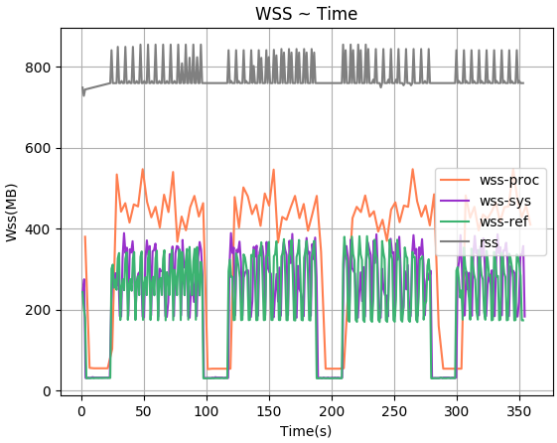
Σχήμα 5.22: 525.x264_r-ref-1[WSS, interval=1s, user level]



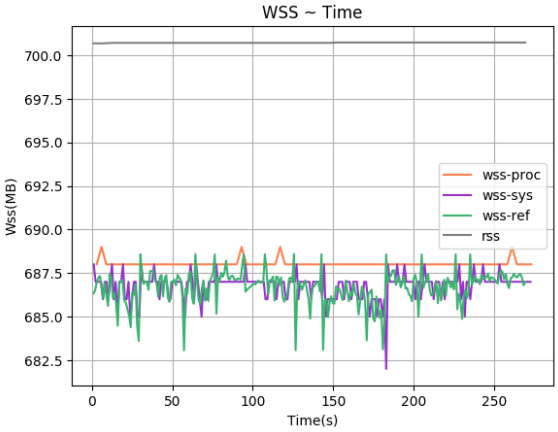
Σχήμα 5.23: 525.x264_r-ref-2[WSS, interval=1s, user level]



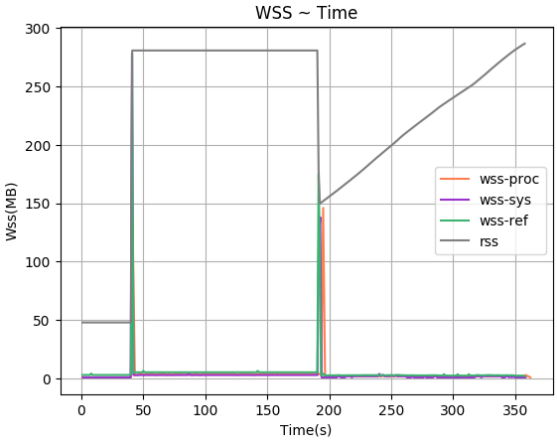
Σχήμα 5.24: 525.x264_r-ref-3[WSS, interval=1s, user level]



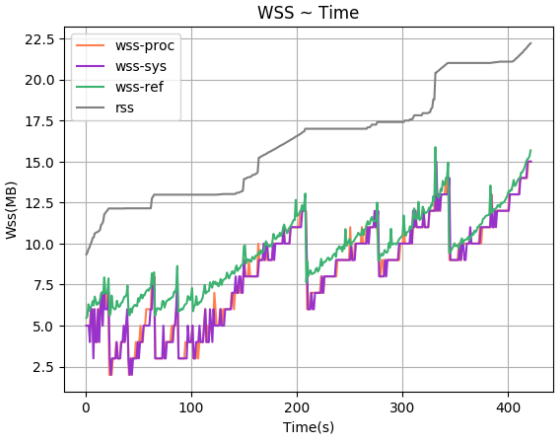
Σχήμα 5.25: 527.cam4_r-ref-1[WSS, interval=1s, user level]



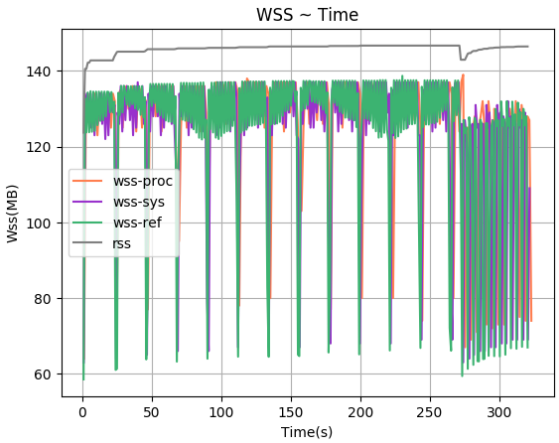
Σχήμα 5.26: 531.deepsjeng_r-ref-1[WSS, interval=1s, user level]



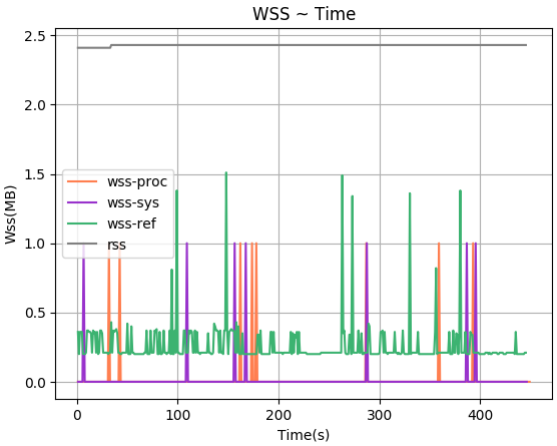
Σχήμα 5.27: 538.imagick_r-ref-1[WSS, interval=1s, user level]



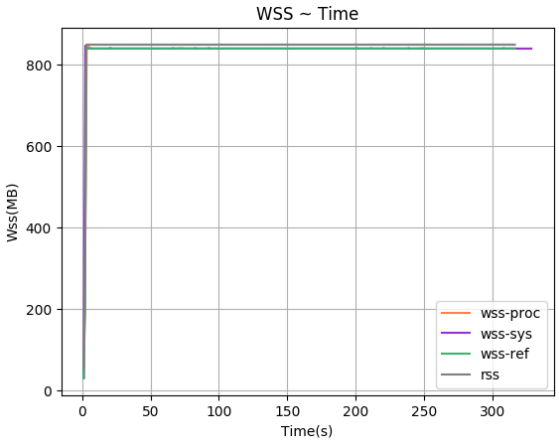
Σχήμα 5.28: 541.leela_r-ref-1[WSS, interval=1s, user level]



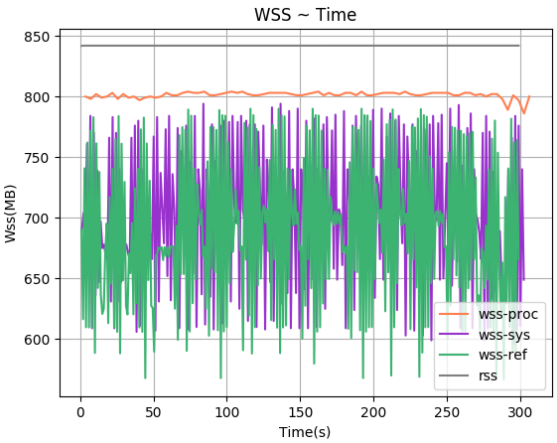
Σχήμα 5.29: 544.nab_r-ref-1[WSS, interval=1s, user level]



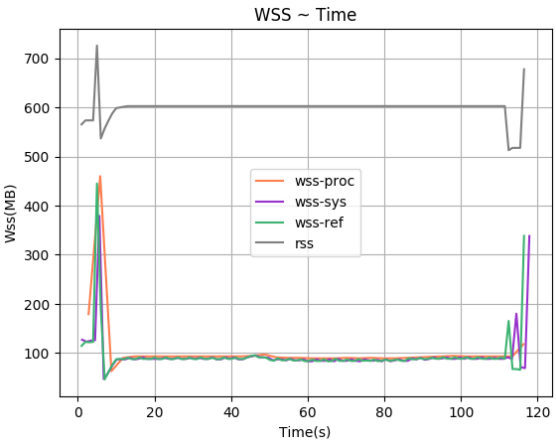
Σχήμα 5.30: 548.exchange2_r-ref-1[WSS, interval=1s, user level]



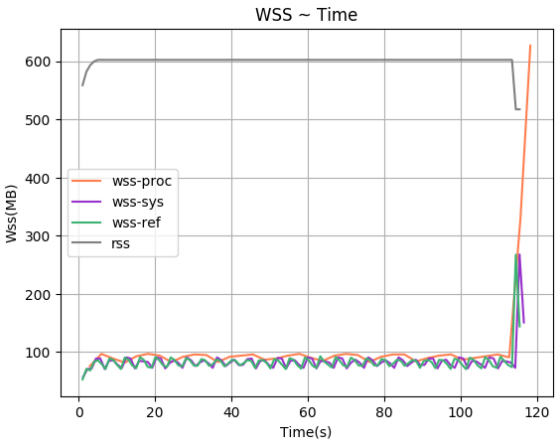
Σχήμα 5.31: 549.fotonik3d_r-ref-1[WSS, interval=1s, user level]



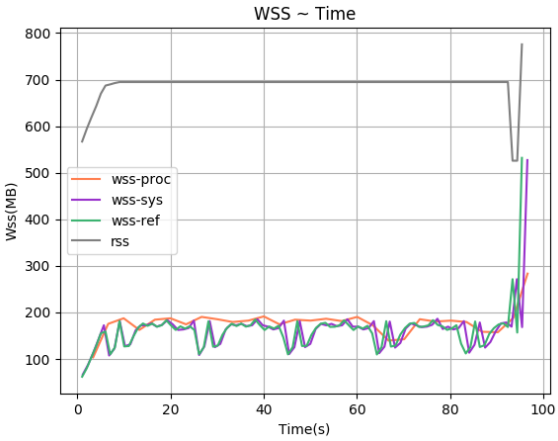
Σχήμα 5.32: 554.roms_r-ref-1[WSS, interval=1s, user level]



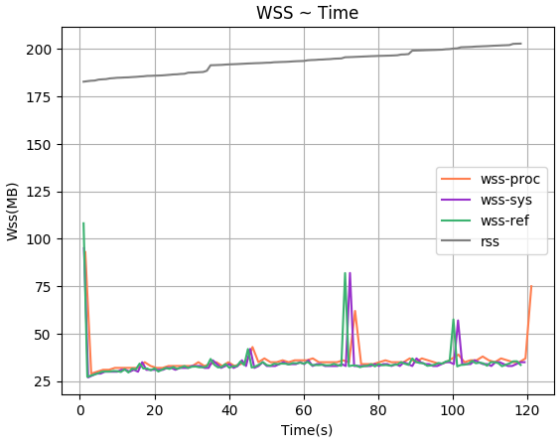
Σχήμα 5.33: 557.xz_r-ref-1[WSS, interval=1s, user level]



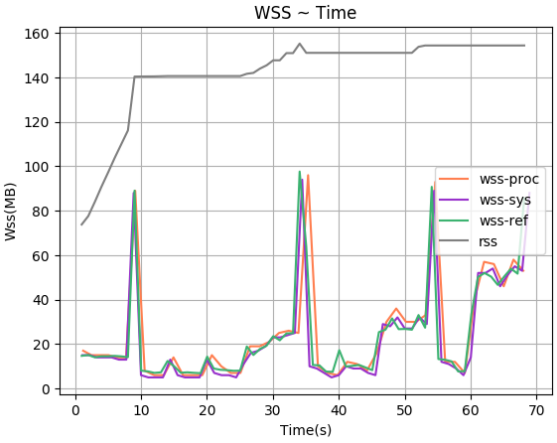
Σχήμα 5.34: 557.xz_r-ref-2[WSS, interval=1s, user level]



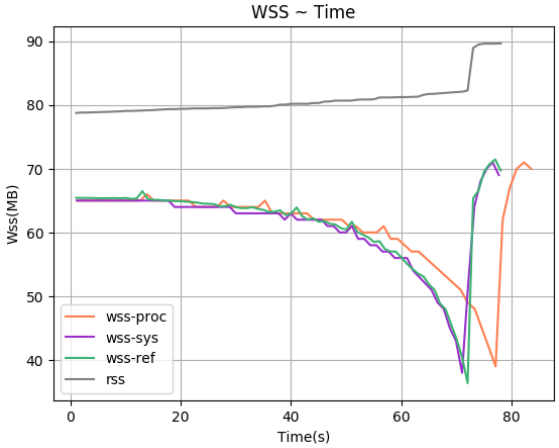
Σχήμα 5.35: 557.xz_r-ref-3[WSS, interval=1s, user level]



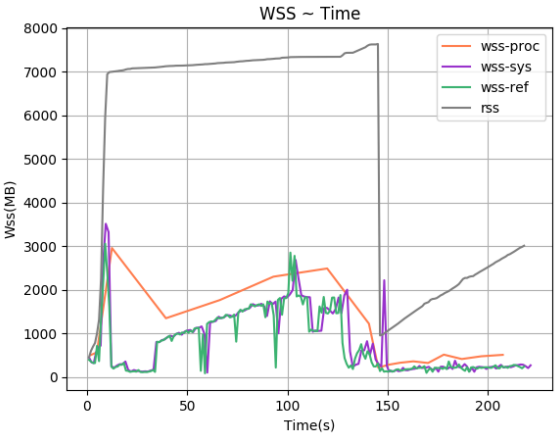
Σχήμα 5.36: 600.perlbench_s-ref-1[WSS, interval=1s, user level]



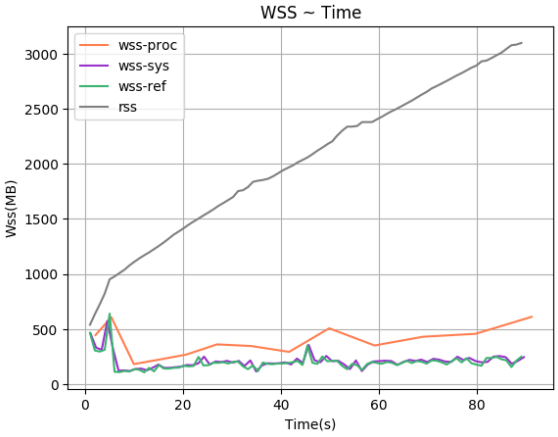
Σχήμα 5.37: 600.perlbench_s-ref-2[WSS, interval=1s, user level]



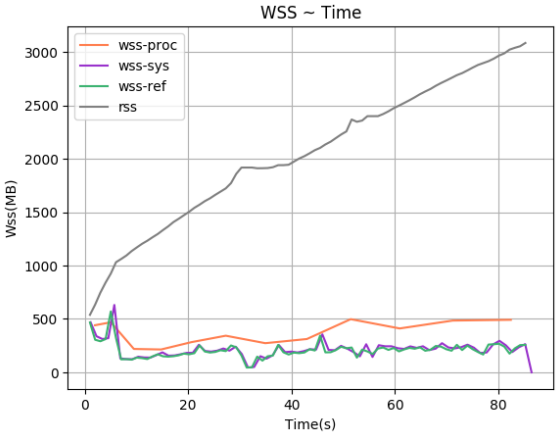
Σχήμα 5.38: 600.perlbench_s-ref-3[WSS, interval=1s, user level]



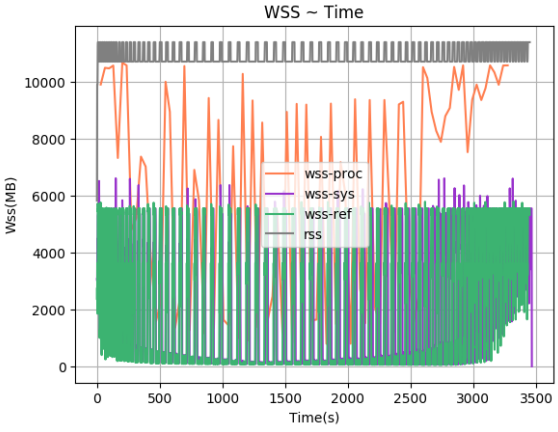
Σχήμα 5.39: 602.gcc_s-ref-1[WSS, interval=1s, user level]



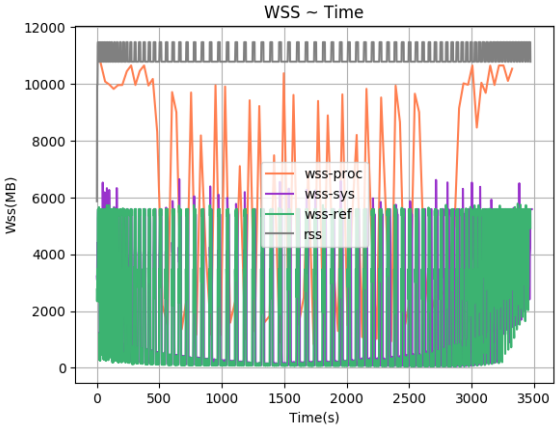
Σχήμα 5.40: 602.gcc_s-ref-2[WSS, interval=1s, user level]



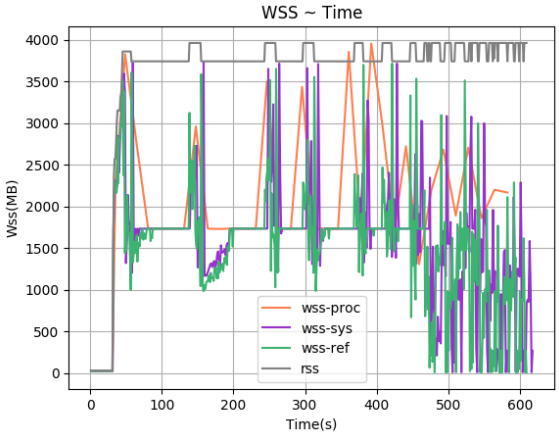
Σχήμα 5.41: 602.gcc_s-ref-3[WSS, interval=1s, user level]



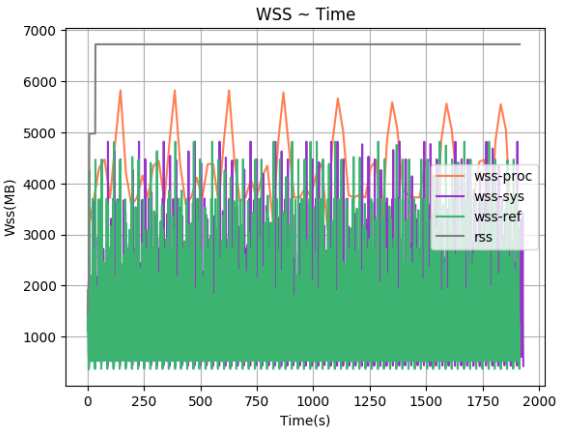
Σχήμα 5.42: 603.bwaves_s-ref-1[WSS, interval=1s, user level]



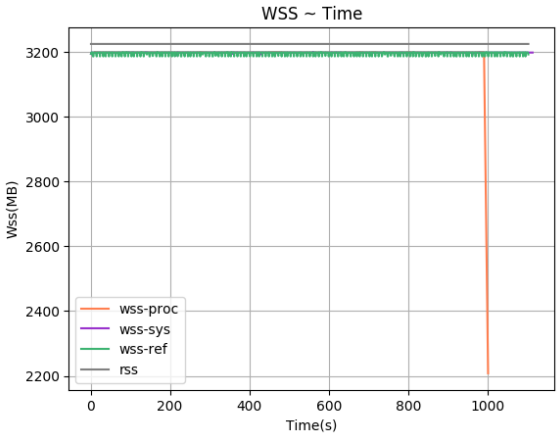
Σχήμα 5.43: 603.bwaves_s-ref-2[WSS, interval=1s, user level]



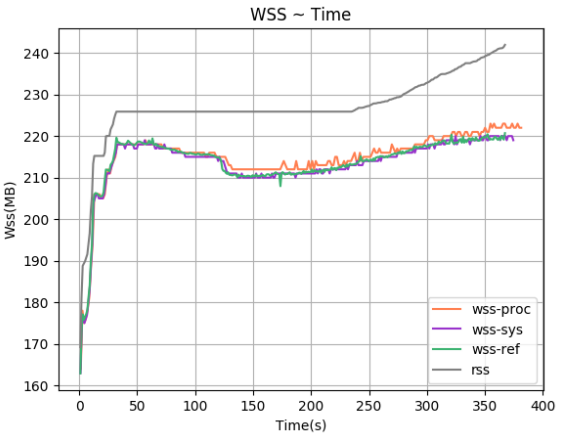
Σχήμα 5.44: 605.mcf_s-ref-1[WSS, interval=1s, user level]



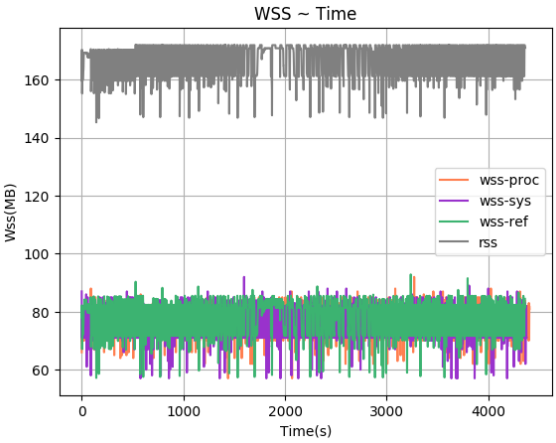
Σχήμα 5.45: 607.cactuBSSN_s-ref-1[WSS, interval=1s, user level]



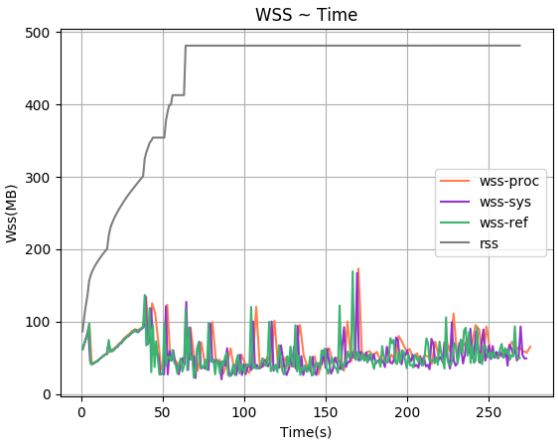
Σχήμα 5.46: 619.lbm_s-ref-1[WSS, interval=1s, user level]



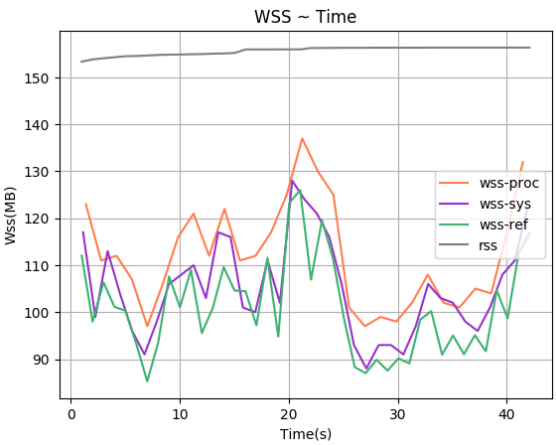
Σχήμα 5.47: 620.omnetpp_s-ref-1[WSS, interval=1s, user level]



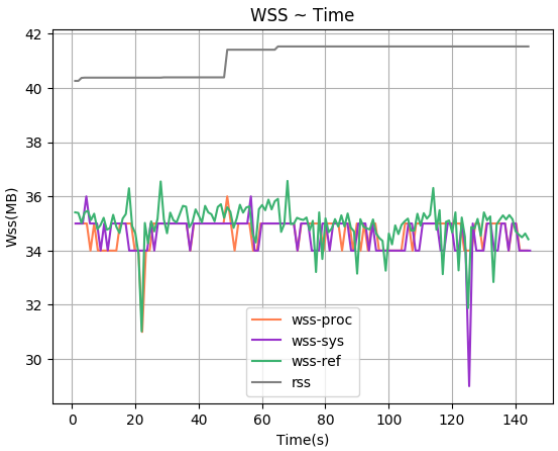
Σχήμα 5.48: 621.wrf_s-ref-1[WSS, interval=1s, user level]



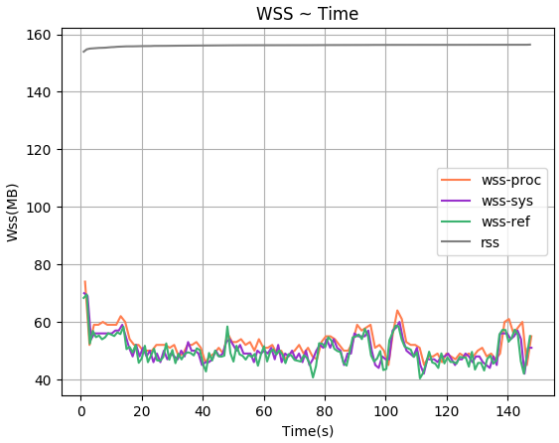
Σχήμα 5.49: 623.xalancbmk_s-ref-1[WSS, interval=1s, user level]



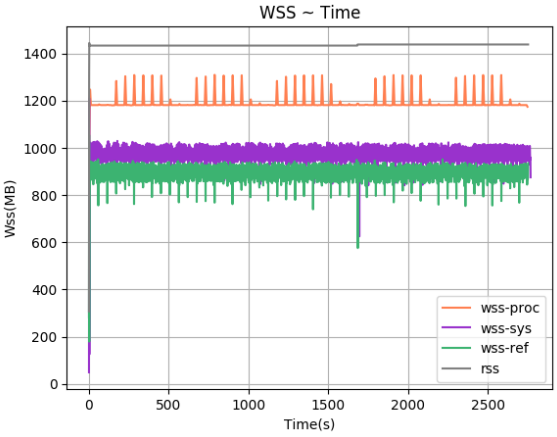
Σχήμα 5.50: 625.x264_s-ref-1[WSS, interval=1s, user level]



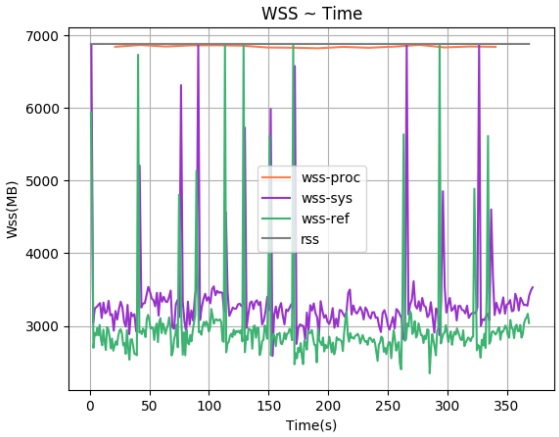
Σχήμα 5.51: 625.x264_s-ref-2[WSS, interval=1s, user level]



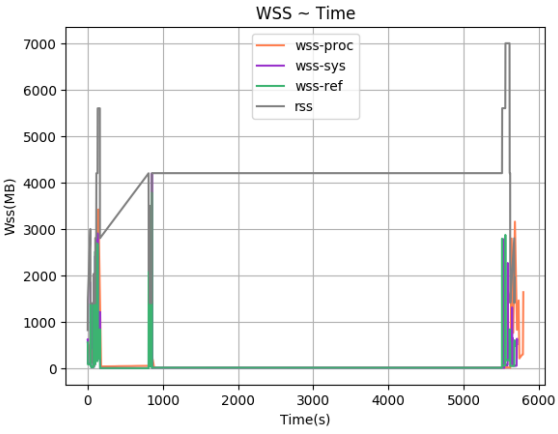
Σχήμα 5.52: 625.x264_s-ref-3[WSS, interval=1s, user level]



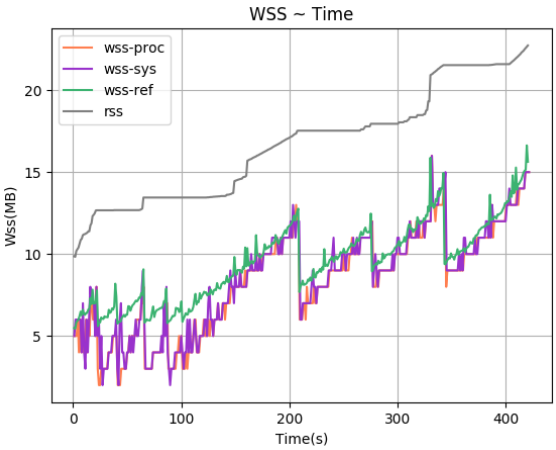
Σχήμα 5.53: 628.pop2_s-ref-1[WSS, interval=1s, user level]



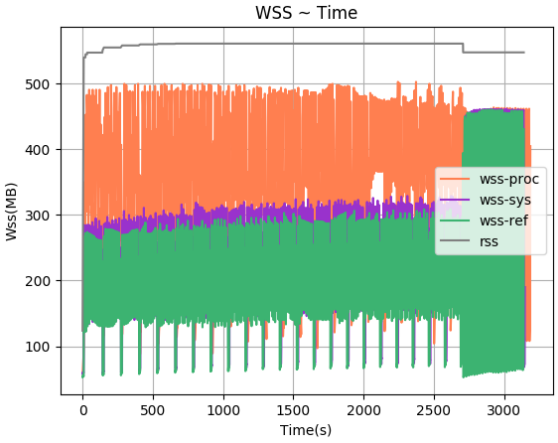
Σχήμα 5.54: 631.deepsjeng_s-ref-1[WSS, interval=1s, user level]



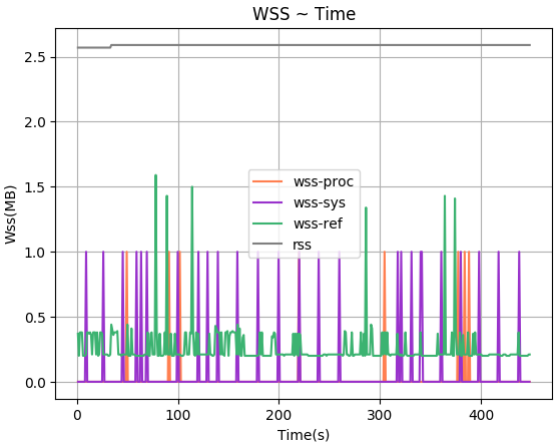
Σχήμα 5.55: 638.imagick_s-ref-1[WSS, interval=1s, user level]



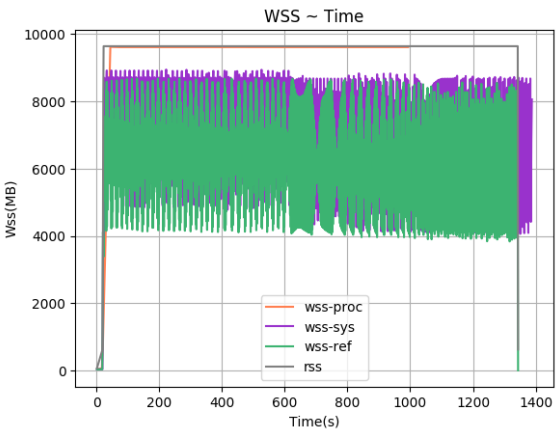
Σχήμα 5.56: 641.leela_s-ref-1[WSS, interval=1s, user level]



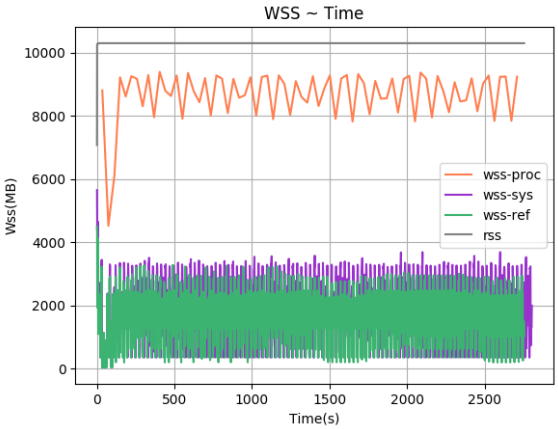
Σχήμα 5.57: 644.nab_s-ref-1[WSS, interval=1s, user level]



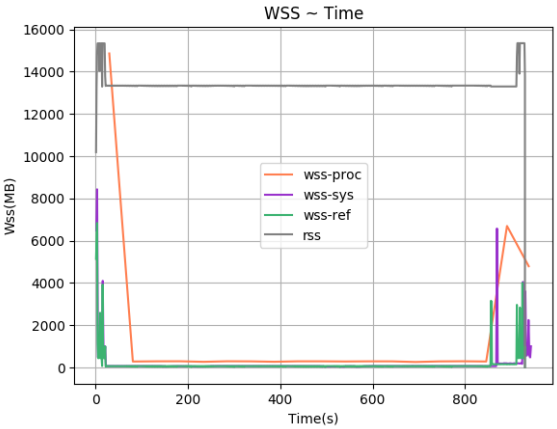
Σχήμα 5.58: 648.exchange2_s-ref-1[WSS, interval=1s, user level]



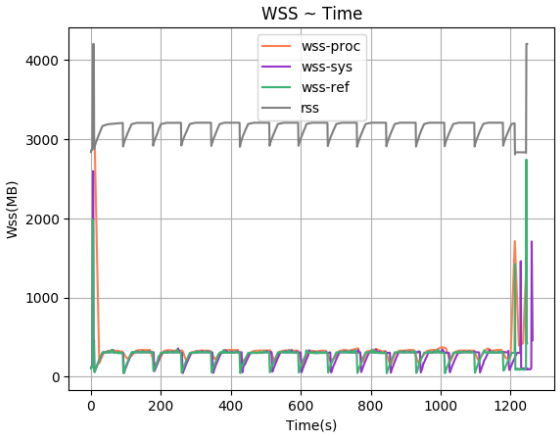
Σχήμα 5.59: 649.fotonik3d_s-ref-1[WSS, interval=1s, user level]



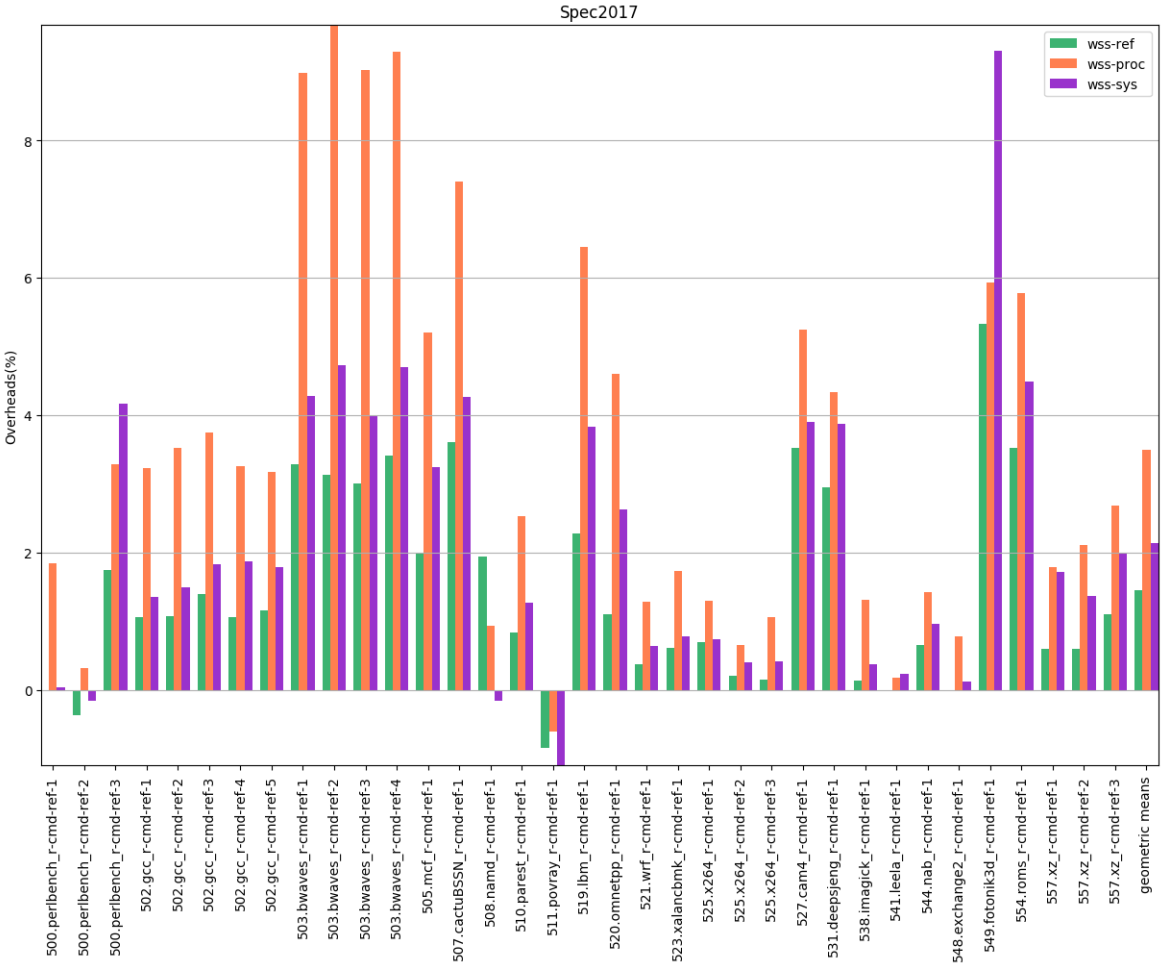
Σχήμα 5.60: 654.roms_s-ref-1[WSS, interval=1s, user level]



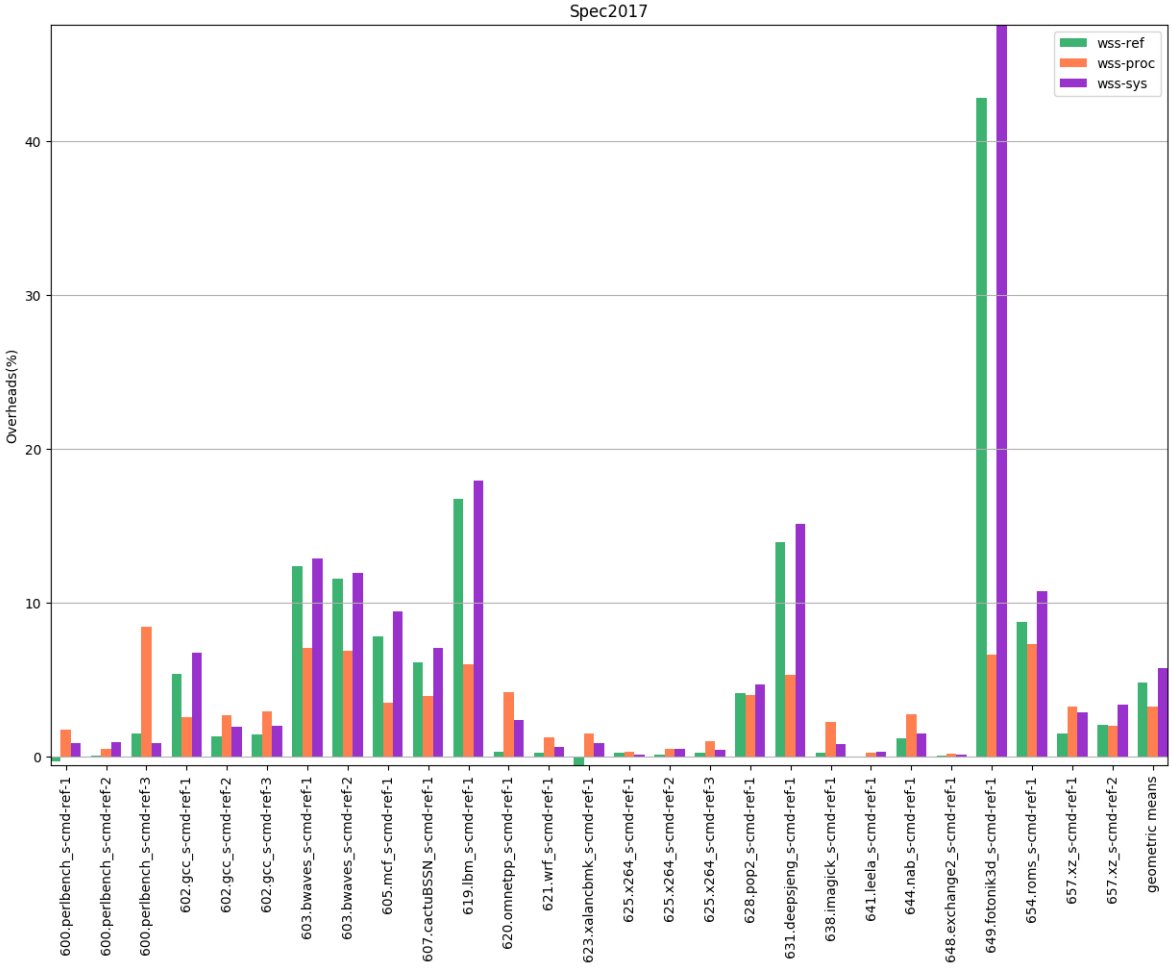
Σχήμα 5.61: 657.xz_s-ref-1[WSS, interval=1s, user level]



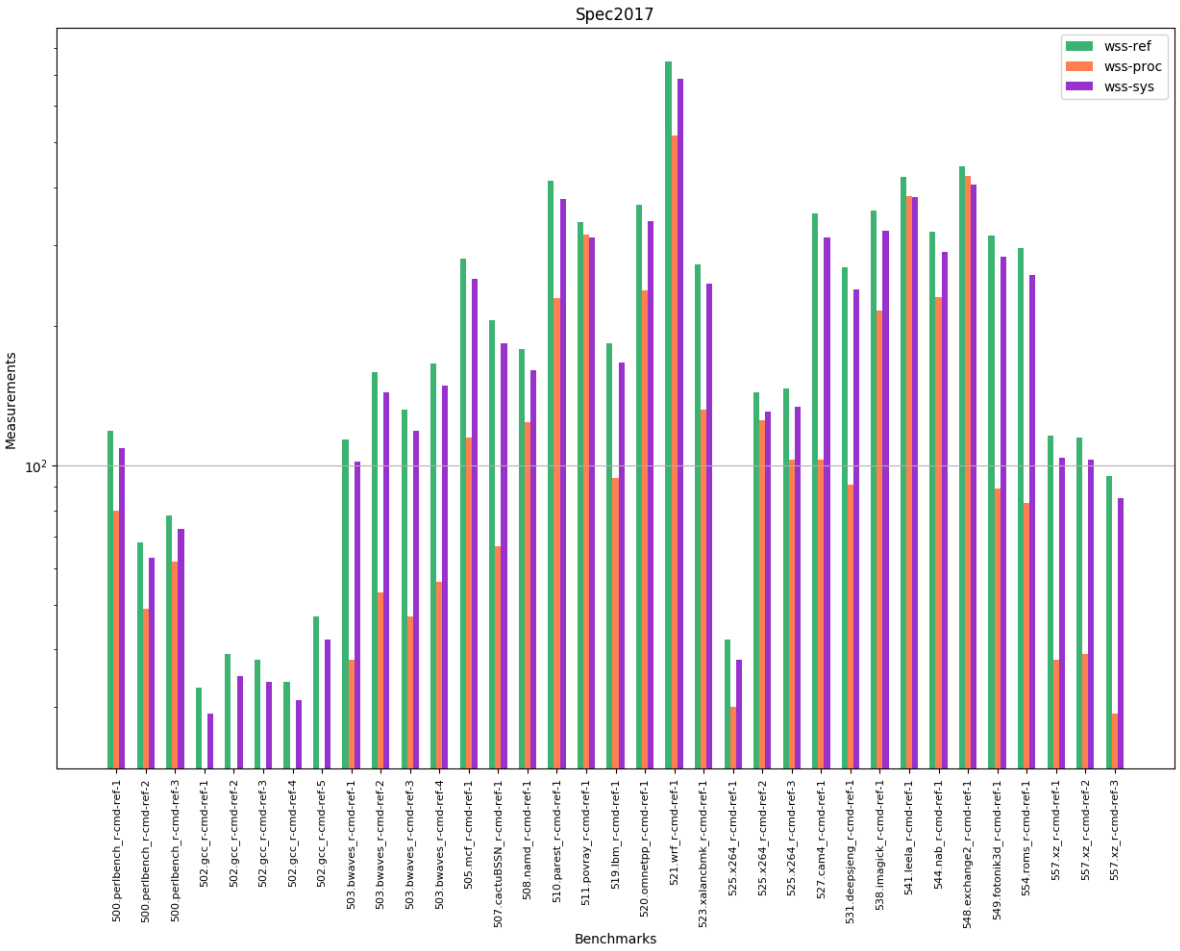
Σχήμα 5.62: 657.xz_s-ref-2[WSS, interval=1s, user level]



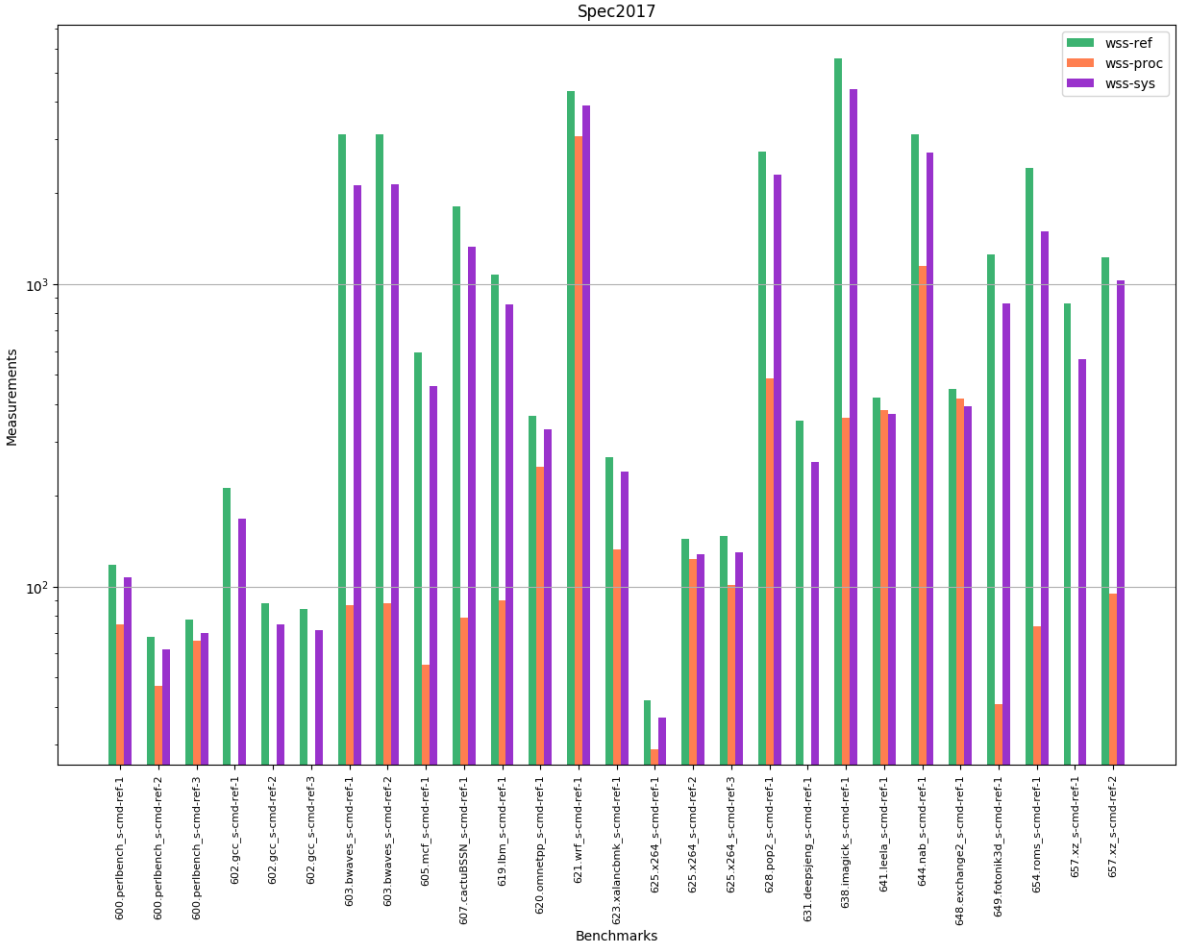
Σχήμα 5.63: 5xx benchmarks[Overheads, interval=1s, user level]



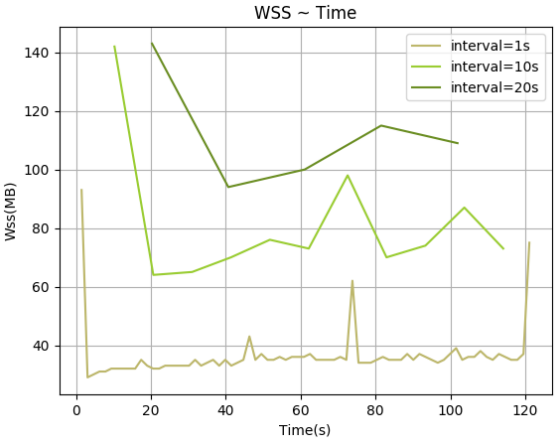
Σχήμα 5.64: 6xx benchmarks[Overheads, interval=1s, user level]



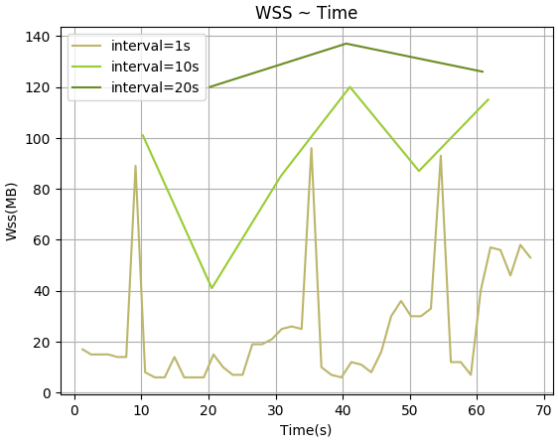
Σχήμα 5.65: 5xx benchmarks[Measurements, interval=1s, user level]



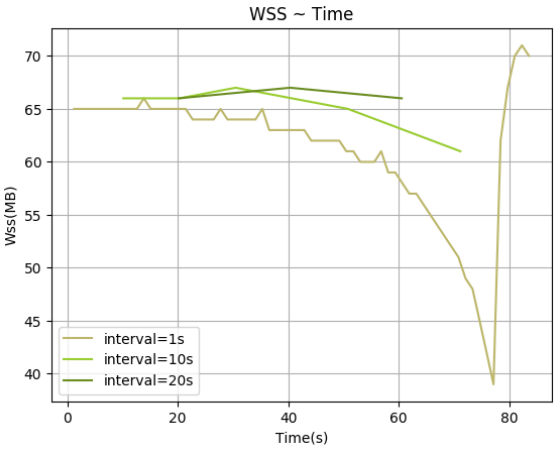
Σχήμα 5.66: 6xx benchmarks[Measurements, interval=1s, user level]



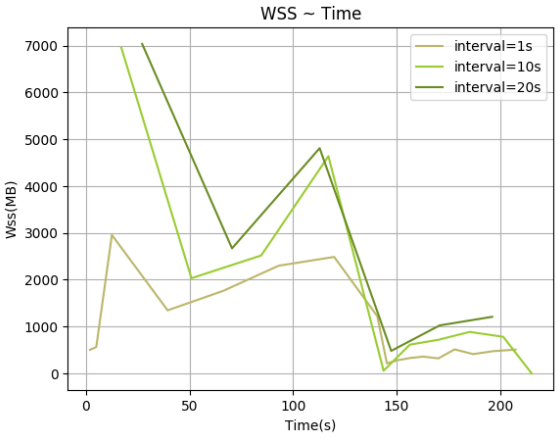
Σχήμα 5.67: 600.perlbench_s-ref-1[WSS, interval=1,10,20s, user level]



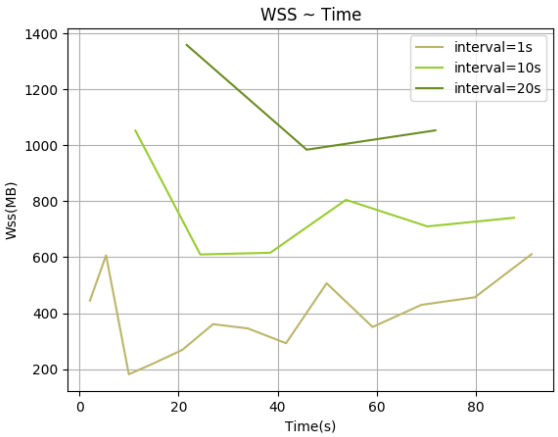
Σχήμα 5.68: 600.perlbench_s-ref-2[WSS, interval=1,10,20s, user level]



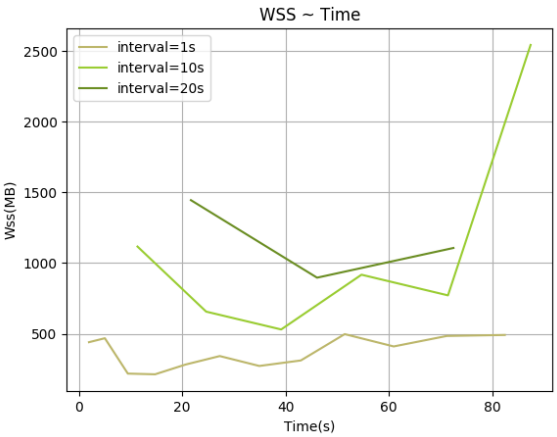
Σχήμα 5.69: 600.perlbench_s-ref-3[WSS, interval=1,10,20s, user level]



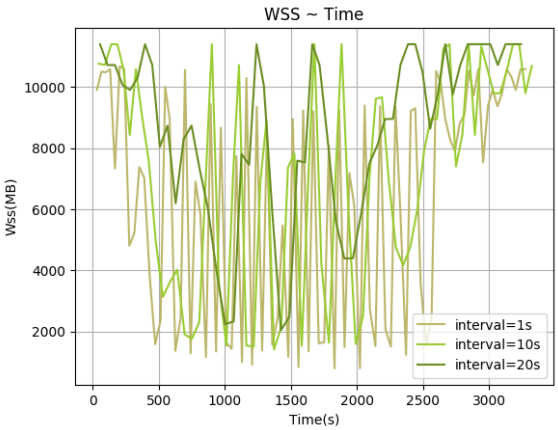
Σχήμα 5.70: 602.gcc_s-ref-1[WSS, interval=1,10,20s, user level]



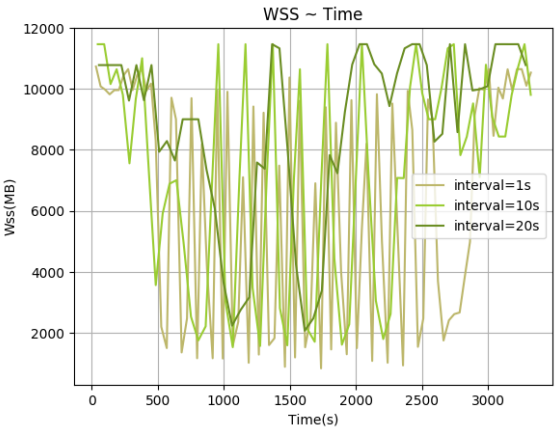
Σχήμα 5.71: 602.gcc_s-ref-2[WSS, interval=1,10,20s, user level]



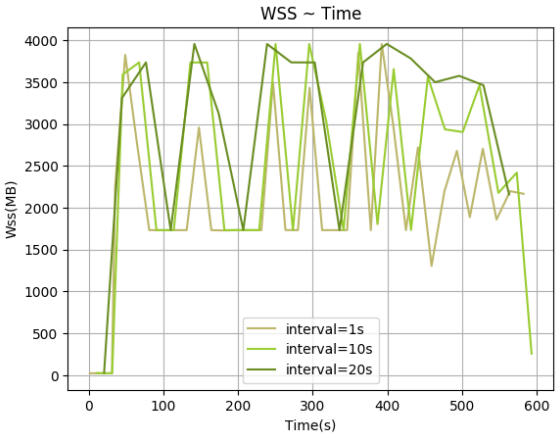
Σχήμα 5.72: 602.gcc_s-ref-3[WSS, interval=1,10,20s, user level]



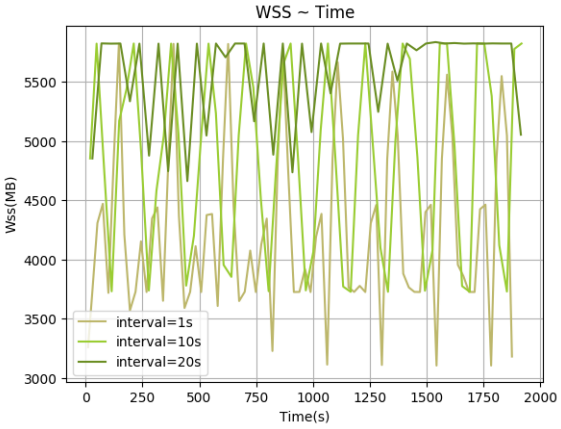
Σχήμα 5.73: 603.bwaves_s-ref-1[WSS, interval=1,10,20s, user level]



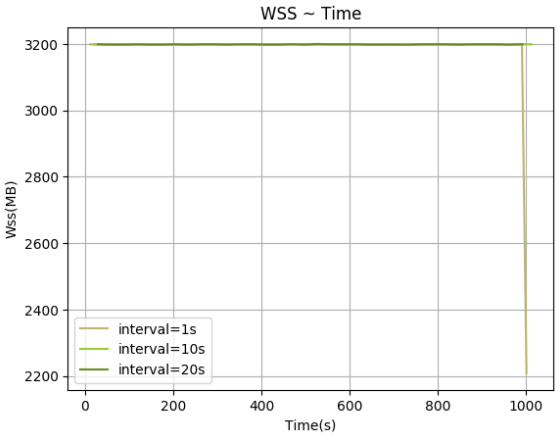
Σχήμα 5.74: 603.bwaves_s-ref-2[WSS, interval=1,10,20s, user level]



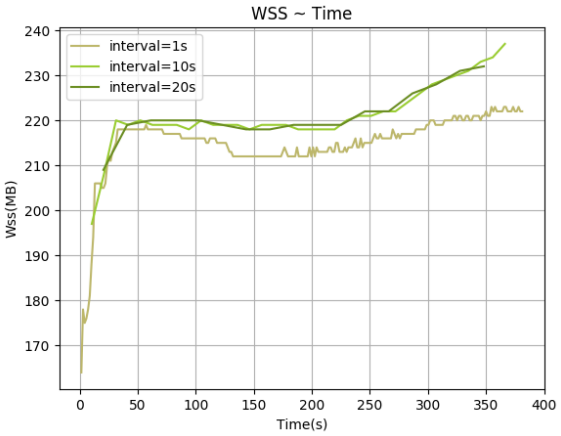
Σχήμα 5.75: 605.mcf_s-ref-1[WSS, interval=1,10,20s, user level]



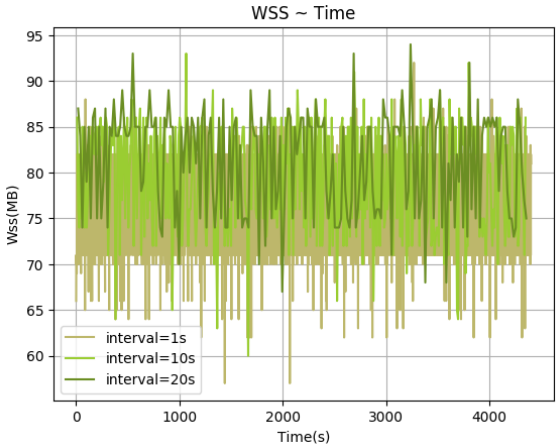
Σχήμα 5.76: 607.cactuBSSN_s-ref-1[WSS, interval=1,10,20s, user level]



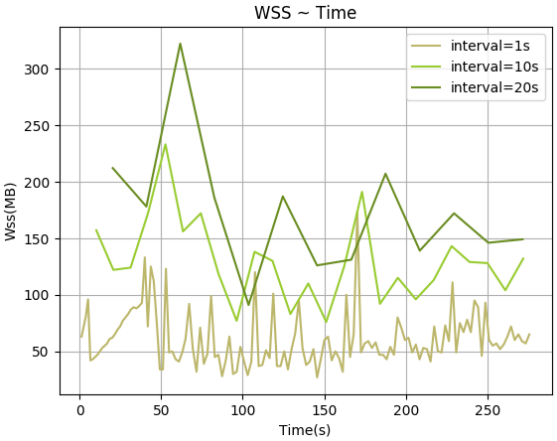
Σχήμα 5.77: 619.lbm_s-ref-1[WSS, interval=1,10,20s, user level]



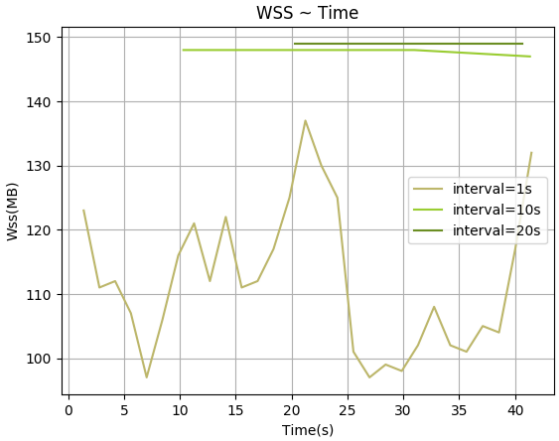
Σχήμα 5.78: 620.omnetpp_s-ref-1[WSS, interval=1,10,20s, user level]



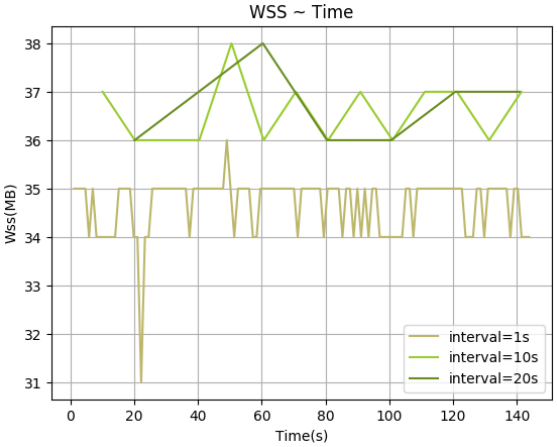
Σχήμα 5.79: 621.wrf_s-ref-1[WSS, interval=1,10,20s, user level]



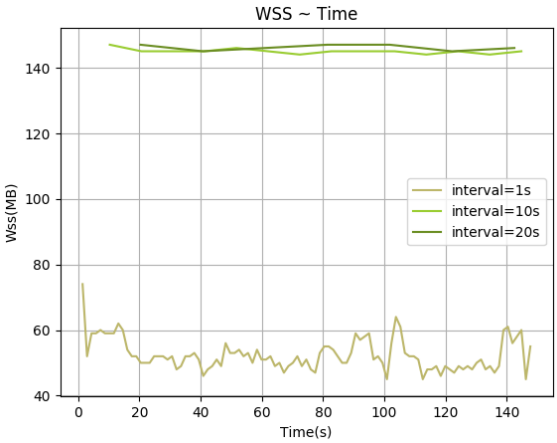
Σχήμα 5.80: 623.xalancbmk_s-ref-1[WSS, interval=1,10,20s, user level]



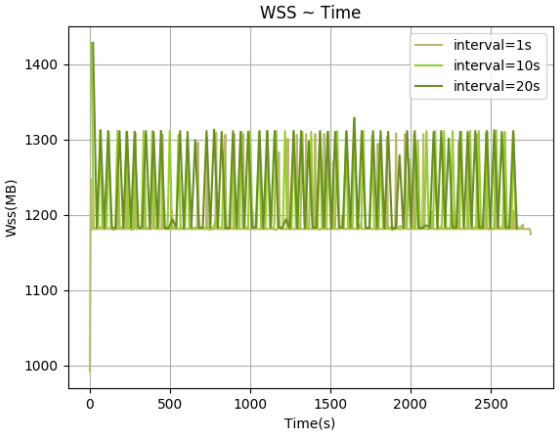
Σχήμα 5.81: 625.x264_s-ref-1[WSS, interval=1,10,20s, user level]



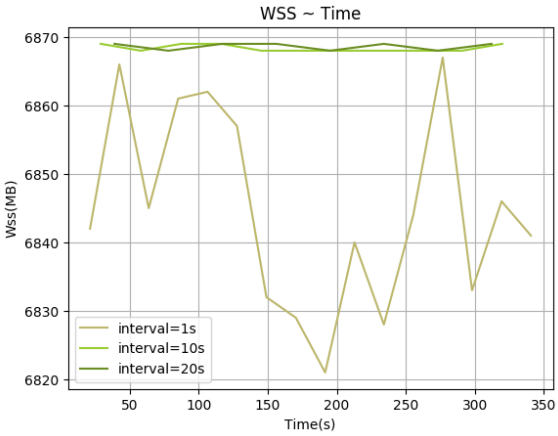
Σχήμα 5.82: 625.x264_s-ref-2[WSS, interval=1,10,20s, user level]



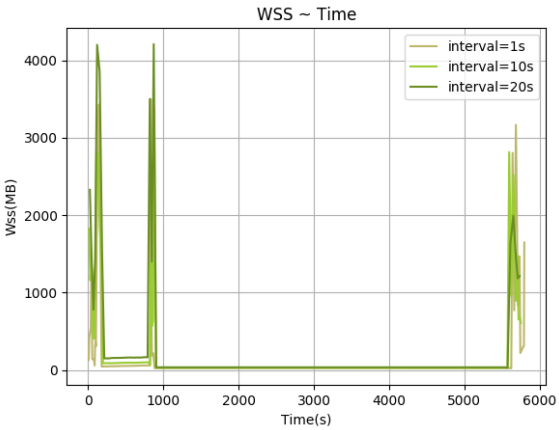
Σχήμα 5.83: 625.x264_s-ref-3[WSS, interval=1,10,20s, user level]



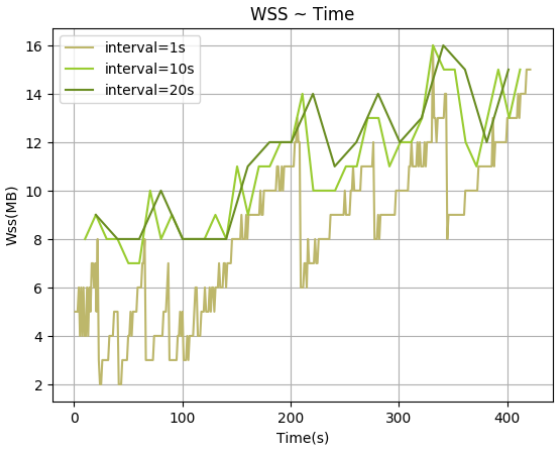
Σχήμα 5.84: 628.pop2_s-ref-1[WSS, interval=1,10,20s, user level]



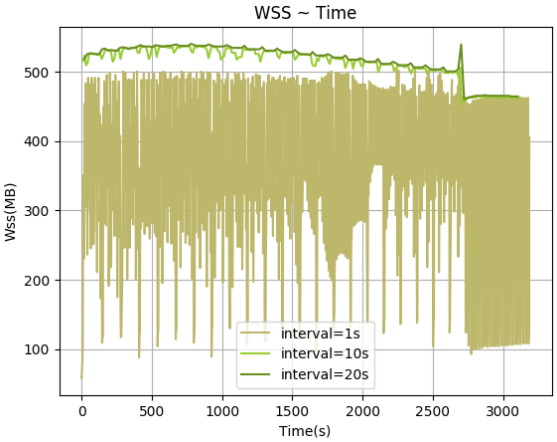
Σχήμα 5.85: 631.deepsjeng_s-ref-1[WSS, interval=1,10,20s, user level]



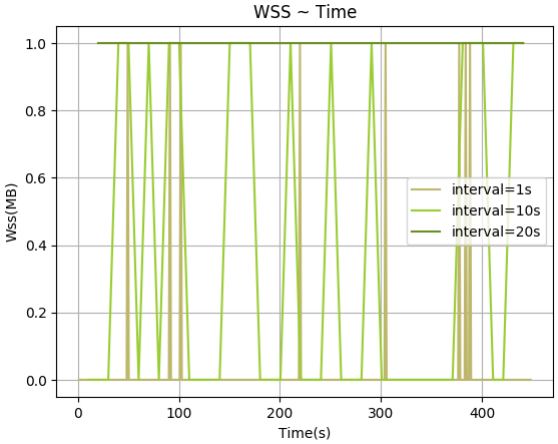
Σχήμα 5.86: 638.imagick_s-ref-1[WSS, interval=1,10,20s, user level]



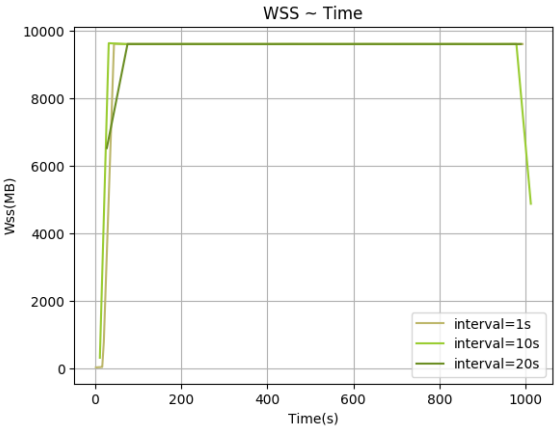
Σχήμα 5.87: 641.leela_s-ref-1[WSS, interval=1,10,20s, user level]



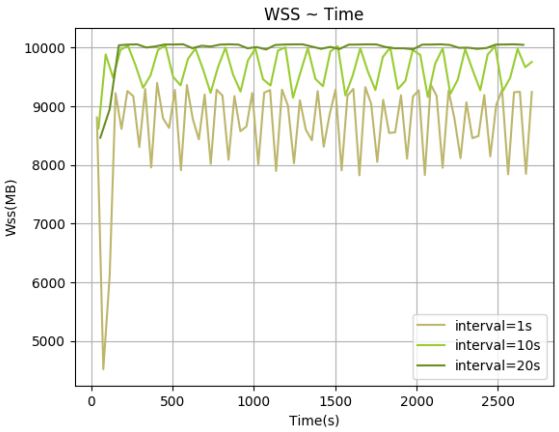
Σχήμα 5.88: 644.nab_s-ref-1[WSS, interval=1,10,20s, user level]



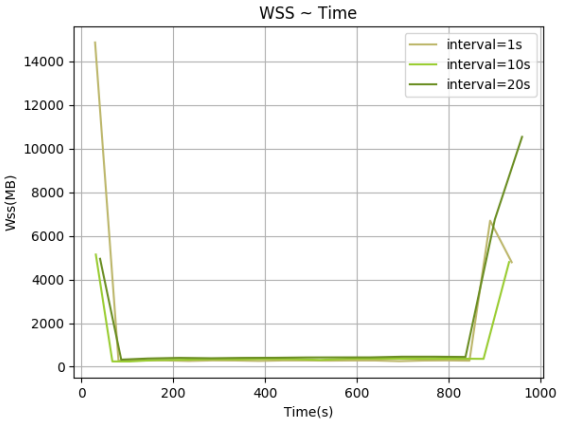
Σχήμα 5.89: 648.exchange2_s-ref-1[WSS, interval=1,10,20s, user level]



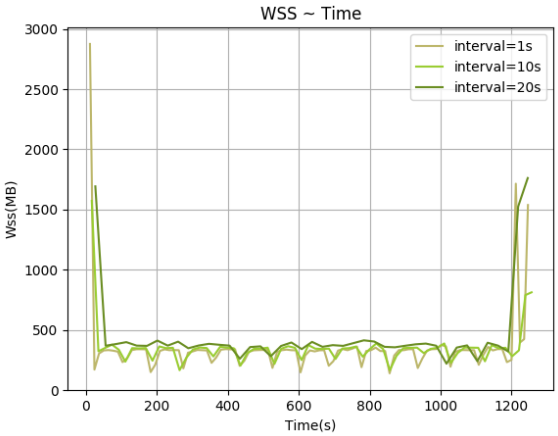
Σχήμα 5.90: 649.fotonik3d_s-ref-1[WSS, interval=1,10,20s, user level]



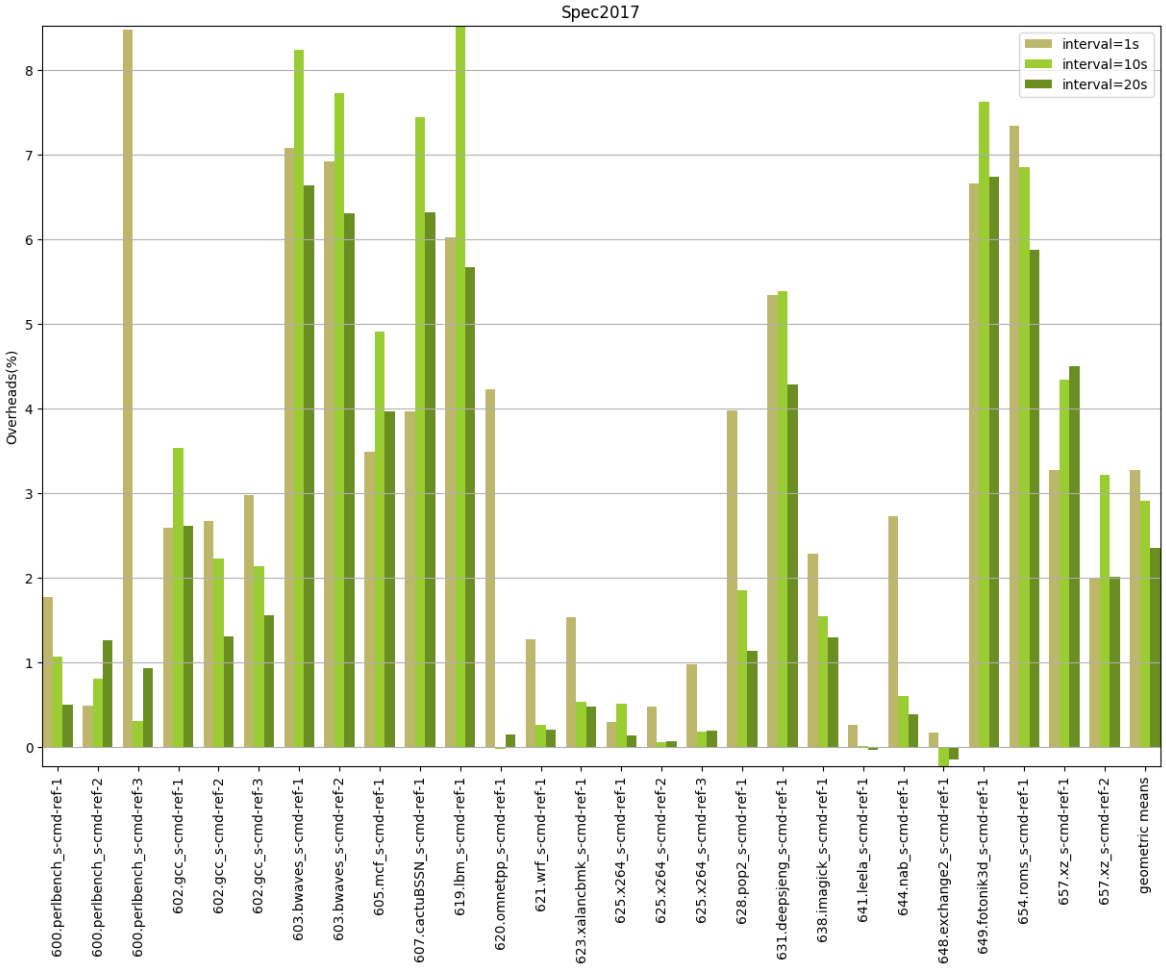
Σχήμα 5.91: 654.roms_s-ref-1[WSS, interval=1,10,20s, user level]



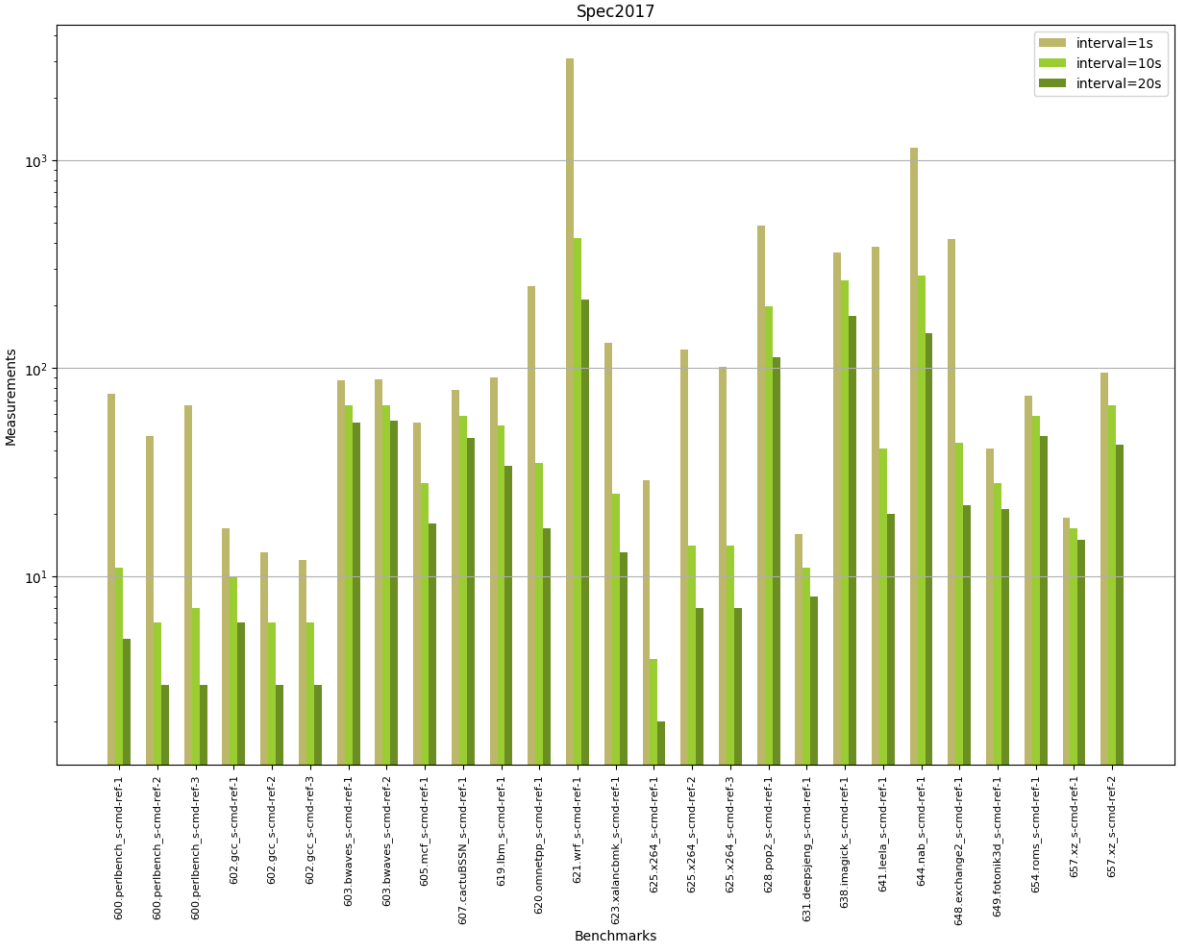
Σχήμα 5.92: 657.xz_s-ref-1[WSS, interval=1,10,20s, user level]



Σχήμα 5.93: 657.xz_s-ref-2[WSS, interval=1,10,20s, user level]



Σχήμα 5.94: 6xx benchmarks[Overheads, interval=1,10,20s, user level]



Σχήμα 5.95: 6xx benchmarks[Measurements, interval=1,10,20s, user level]

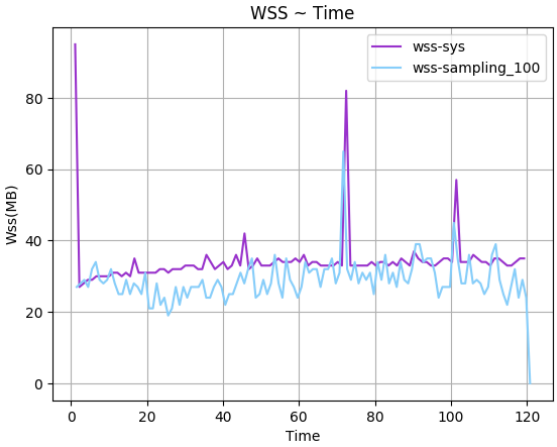
5.1.2 Sampling & Idle Page Tracking

Τα συγκεκριμένα πειράματα έγιναν με τροποποιημένο το εργαλείο wss-proc ώστε να εφαρμόσει δειγματοληψία στις σελίδες που ελέγχει. Η δειγματοληψία έγινε με δύο τρόπους που παρουσιάζονται παρακάτω μαζί με τα αποτελέσματα των μετρήσεων. Από το σημείο αυτό και στο εξής θα δίνουμε έμφαση στα 6xx benchmarks καθώς εμφανίζουν μεγαλύτερο working set και παρουσιάζουν μεγαλύτερο overhead όταν συνεκτελούνται με τα διάφορα monitoring tools. Πέρα από τα πειράματα που εκτελέσαμε σε τοπικό περιβάλλον Linux, τρέξαμε και ορισμένα σε εικονικό περιβάλλον Linux με QEMU/KVM hypervisor χρησιμοποιώντας τη δεύτερη μέθοδο δειγματοληψίας που παρουσιάζουμε παρακάτω όπως και τη μέθοδο idle page tracking με το εργαλείο wss-sys.

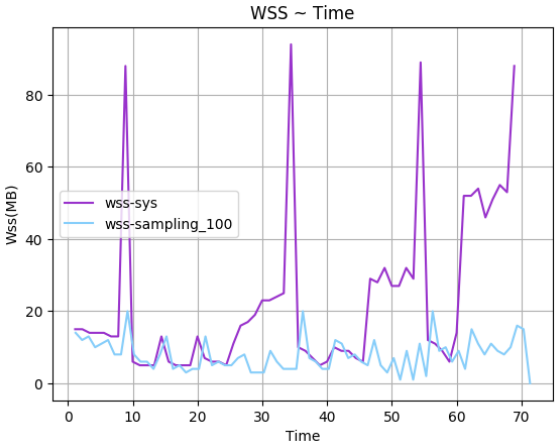
Sampling με σταθερό αριθμό δειγμάτων & Idle Page Tracking

Στη συγκεκριμένη μέθοδο δειγματοληψίας επιλέγουμε N δείγματα σελίδων από αυτές που αντιστοιχούν στη διεργασία, ώστε να κάνουμε reset και να διαβάζουμε μόνο τα bits αυτών. Έπειτα εξάγουμε αναλογικά την εκτίμηση για το συνολικό WSS της διεργασίας. Ως πλήθος δειγμάτων διαλέξαμε τις τιμές 100 και 1000. Στα γραφήματα 5.96 - 5.122 απεικονίζονται οι μετρήσεις με 100 δείγματα που προέκυψαν για το WSS των benchmarks με διάστημα παρακολούθησης 1s αντιπαραβαλλόμενες με αυτές που λήφθηκαν με το wss-sys, ενώ στα 5.123 και 5.124 παρουσιάζονται αντίστοιχα τα overheads τους και το πλήθος των μετρήσεων που έλαβαν. Στα 5.125 - 5.205 απεικονίζονται οι μετρήσεις WSS για διαστήματα παρακολούθησης 1, 10 και 20s αντιπαραβαλλόμενες με αυτές που λήφθηκαν με το wss-proc, ενώ στα 5.206 - 5.208 και 5.209 - 5.211 παρουσιάζονται αντίστοιχα τα overheads τους και το πλήθος των μετρήσεων που έλαβαν.

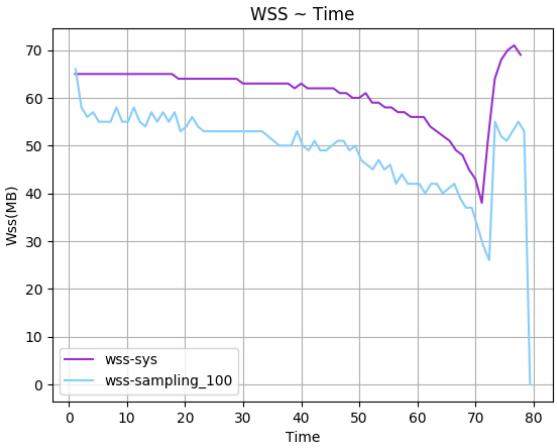
Στα γραφήματα 5.212 - 5.238 απεικονίζονται οι μετρήσεις με 1000 δείγματα που προέκυψαν για το WSS των benchmarks με διάστημα παρακολούθησης 1s αντιπαραβαλλόμενες με αυτές που λήφθηκαν με το wss-sys, ενώ στα 5.239 και 5.240 παρουσιάζονται αντίστοιχα τα overheads τους και το πλήθος των μετρήσεων που έλαβαν. Στα 5.241 - 5.321 απεικονίζονται οι μετρήσεις WSS για διαστήματα παρακολούθησης 1, 10 και 20s αντιπαραβαλλόμενες με αυτές που λήφθηκαν με το wss-proc, ενώ στα 5.322 - 5.324 και 5.325 - 5.327 παρουσιάζονται αντίστοιχα τα overheads τους και το πλήθος των μετρήσεων που έλαβαν.



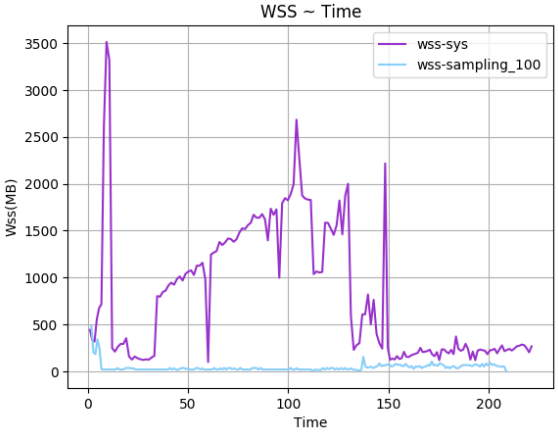
Σχήμα 5.96: 600.perlbench_s-ref-1[WSS, samples=100, interval=1s, user level]



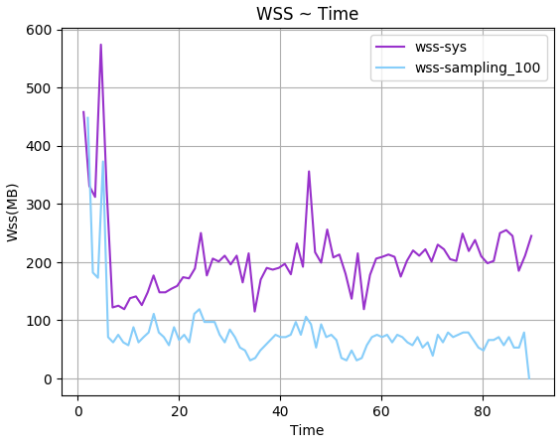
Σχήμα 5.97: 600.perlbench_s-ref-2[WSS, samples=100, interval=1s, user level]



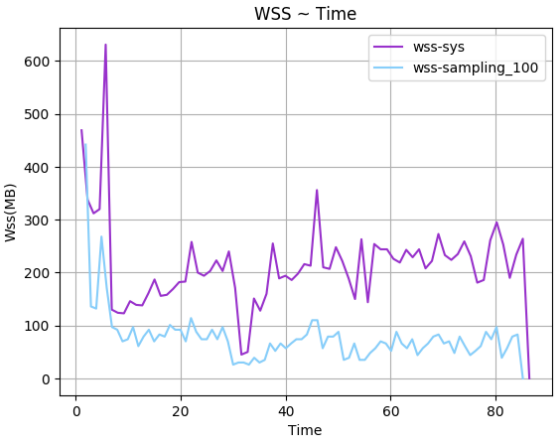
Σχήμα 5.98: 600.perlbench_s-ref-3[WSS, samples=100, interval=1s, user level]



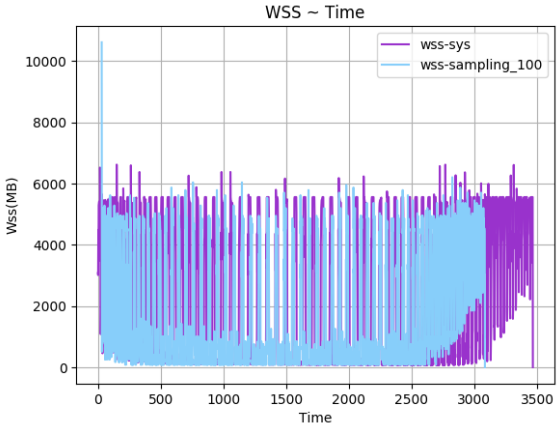
Σχήμα 5.99: 602.gcc_s-ref-1[WSS, samples=100, interval=1s, user level]



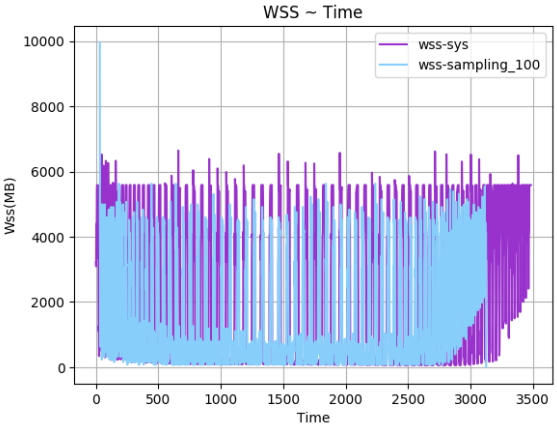
Σχήμα 5.100: 602.gcc_s-ref-2[WSS, samples=100, interval=1s, user level]



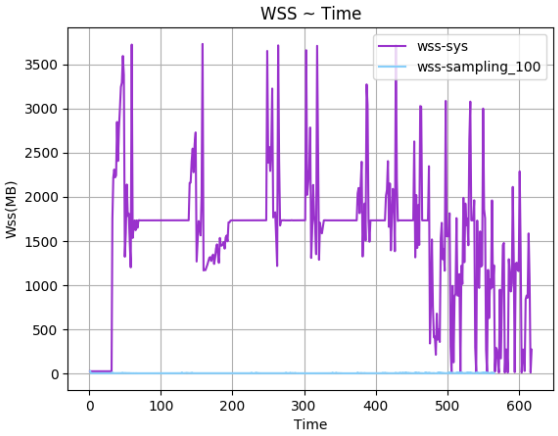
Σχήμα 5.101: 602.gcc_s-ref-3[WSS, samples=100, interval=1s, user level]



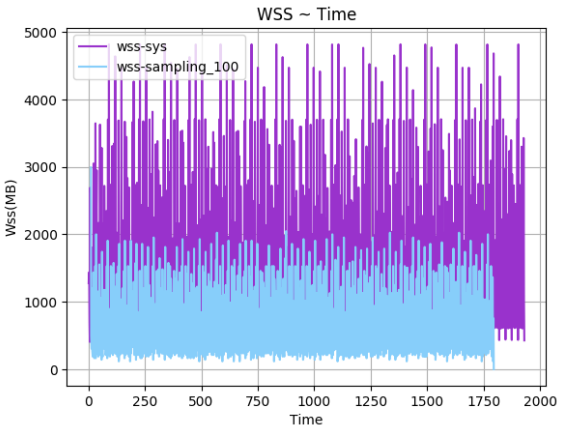
Σχήμα 5.102: 603.bwaves_s-ref-1[WSS, samples=100, interval=1s, user level]



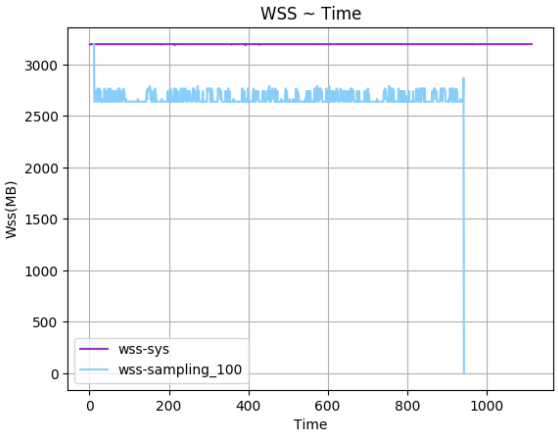
Σχήμα 5.103: 603.bwaves_s-ref-2[WSS, samples=100, interval=1s, user level]



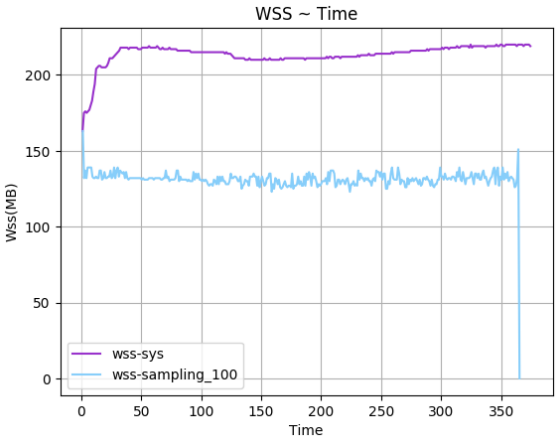
Σχήμα 5.104: 605.mcf_s-ref-1[WSS, samples=100, interval=1s, user level]



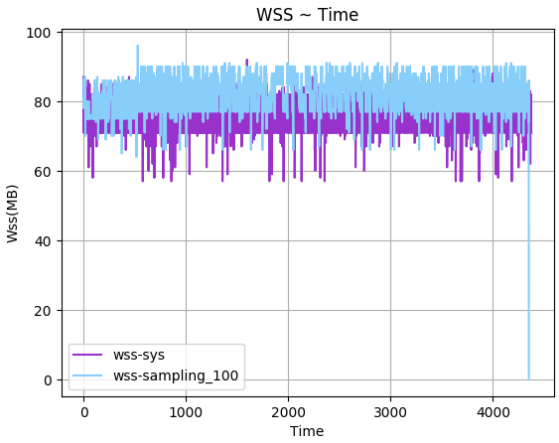
Σχήμα 5.105: 607.cactuBSSN_s-ref-1[WSS, samples=100, interval=1s, user level]



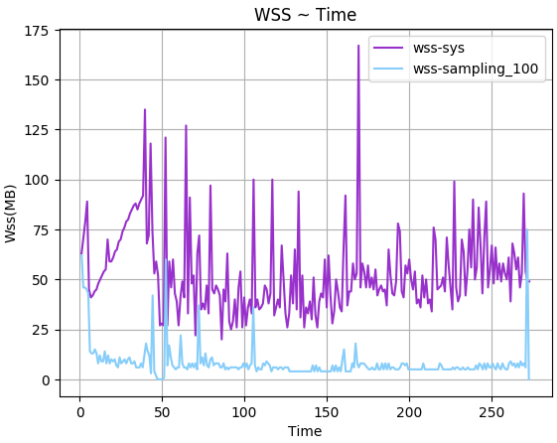
Σχήμα 5.106: 619.lbm_s-ref-1[WSS, samples=100, interval=1s, user level]



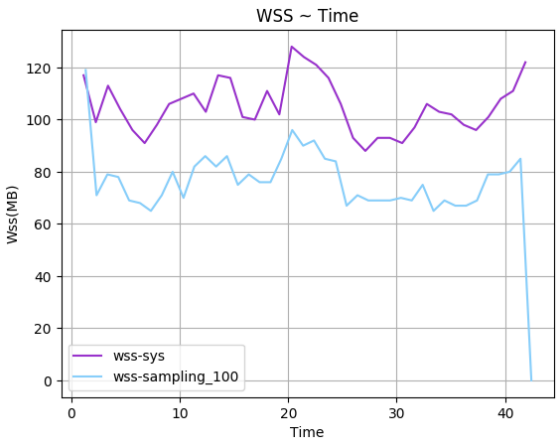
Σχήμα 5.107: 620.omnetpp_s-ref-1[WSS, samples=100, interval=1s, user level]



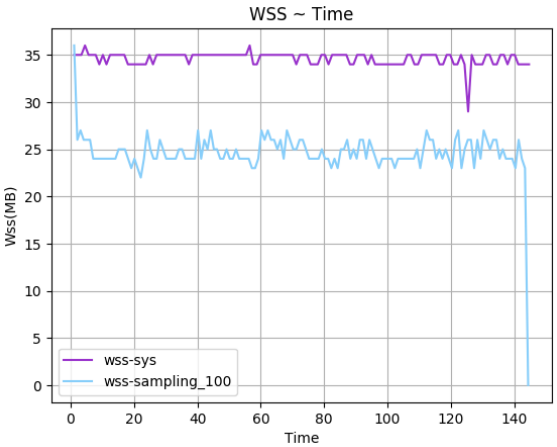
Σχήμα 5.108: 621.wrf_s-ref-1[WSS, samples=100, interval=1s, user level]



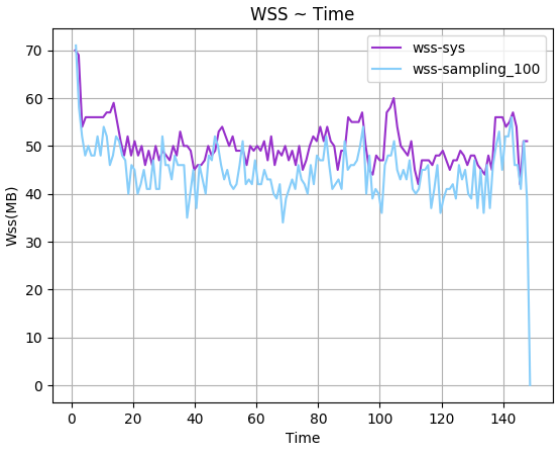
Σχήμα 5.109: 623.xalancbmk_s-ref-1[WSS, samples=100, interval=1s, user level]



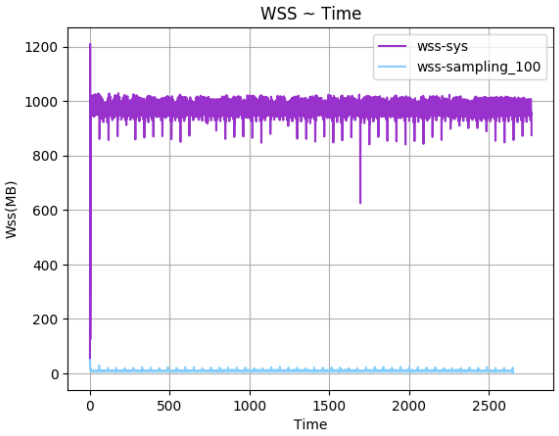
Σχήμα 5.110: 625.x264_s-ref-1[WSS, samples=100, interval=1s, user level]



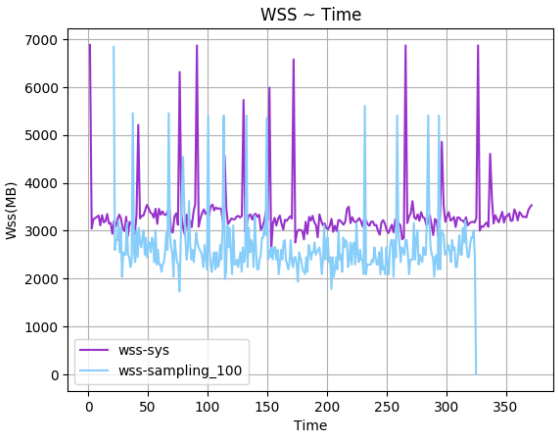
Σχήμα 5.111: 625.x264_s-ref-2[WSS, samples=100, interval=1s, user level]



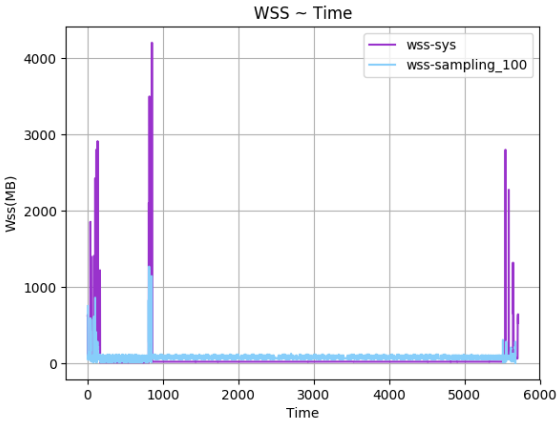
Σχήμα 5.112: 625.x264_s-ref-3[WSS, samples=100, interval=1s, user level]



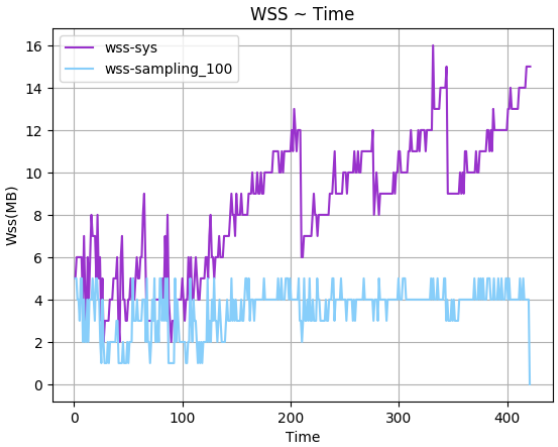
Σχήμα 5.113: 628.pop2_s-ref-1[WSS, samples=100, interval=1s, user level]



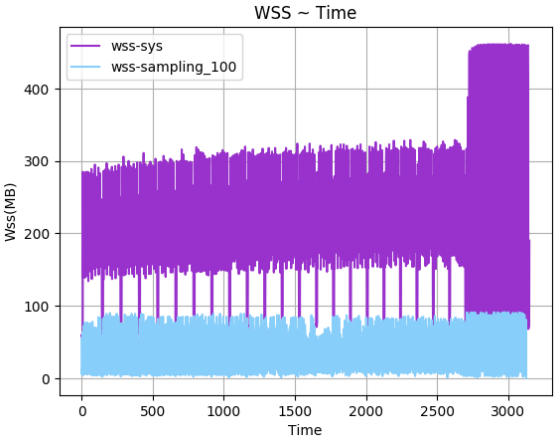
Σχήμα 5.114: 631.deepsjeng_s-ref-1[WSS, samples=100, interval=1s, user level]



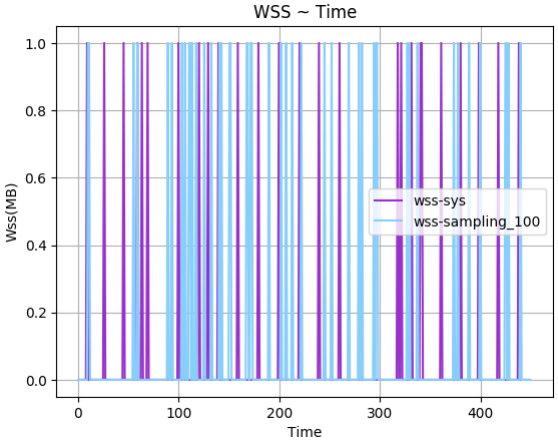
Σχήμα 5.115: 638.imagick_s-ref-1[WSS, samples=100, interval=1s, user level]



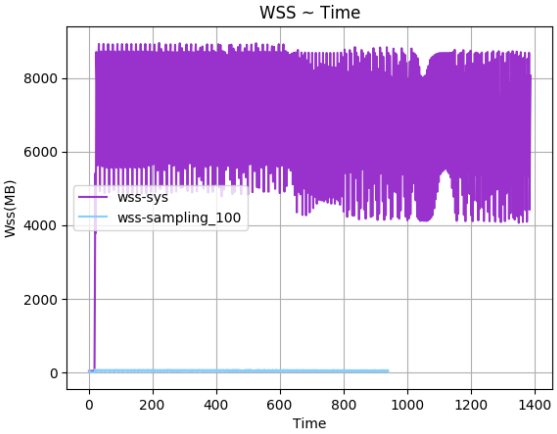
Σχήμα 5.116: 641.leela_s-ref-1[WSS, samples=100, interval=1s, user level]



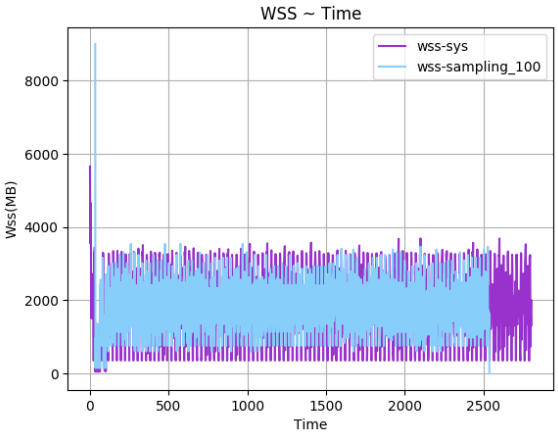
Σχήμα 5.117: 644.nab_s-ref-1[WSS, samples=100, interval=1s, user level]



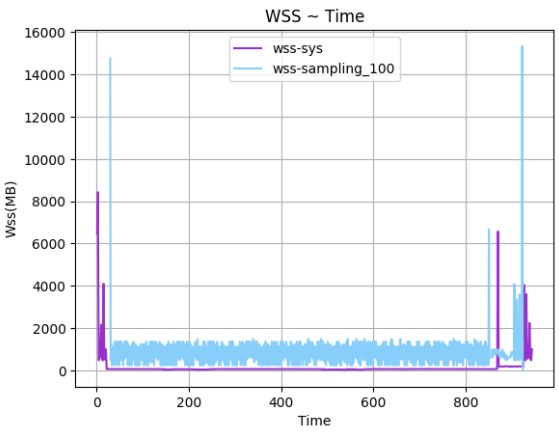
Σχήμα 5.118: 648.exchange2_s-ref-1[WSS, samples=100, interval=1s, user level]



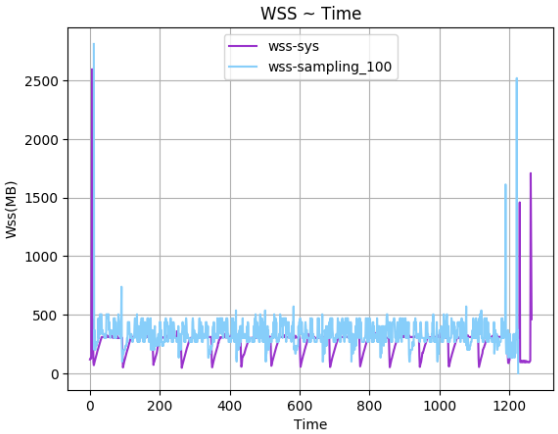
Σχήμα 5.119: 649.fotonik3d_s-ref-1[WSS, samples=100, interval=1s, user level]



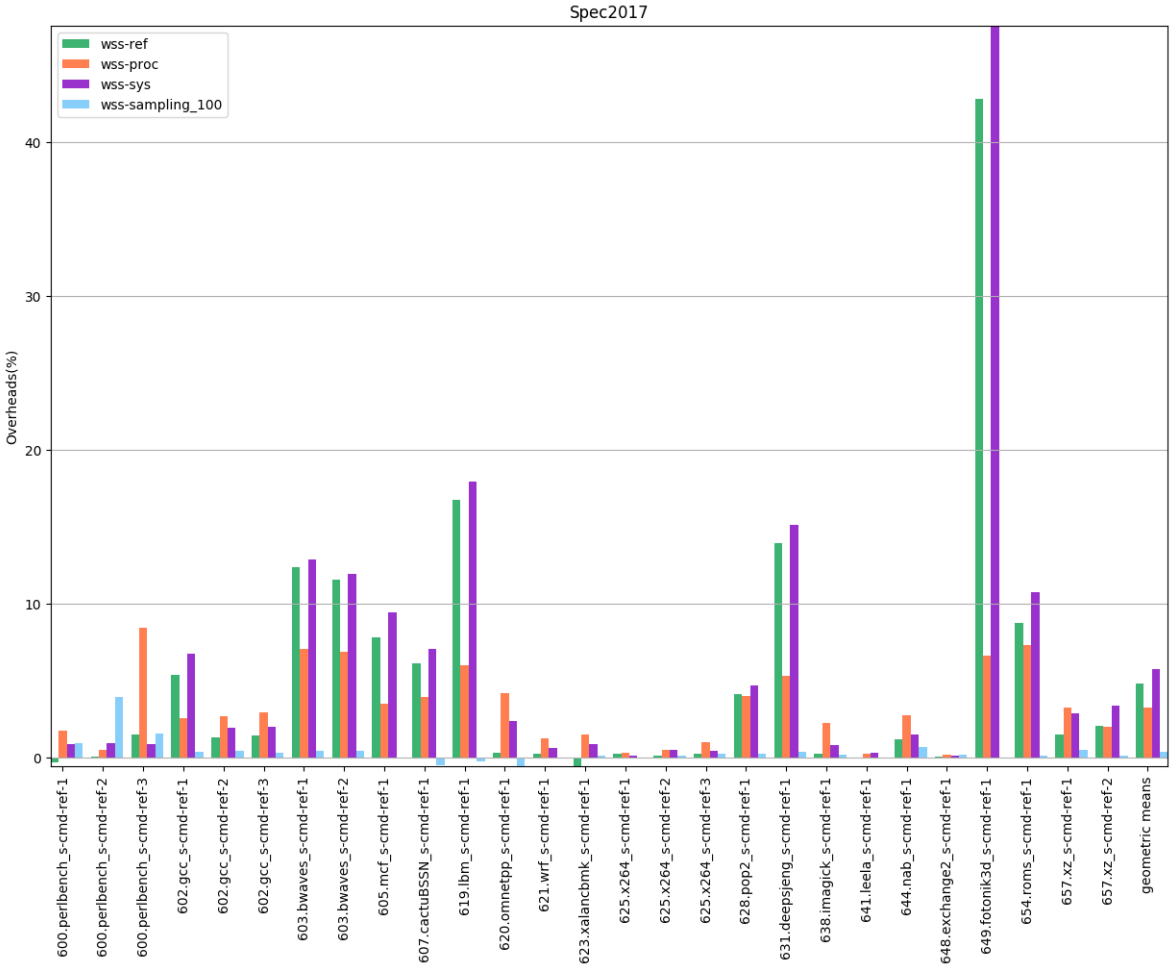
Σχήμα 5.120: 654.roms_s-ref-1[WSS, samples=100, interval=1s, user level]



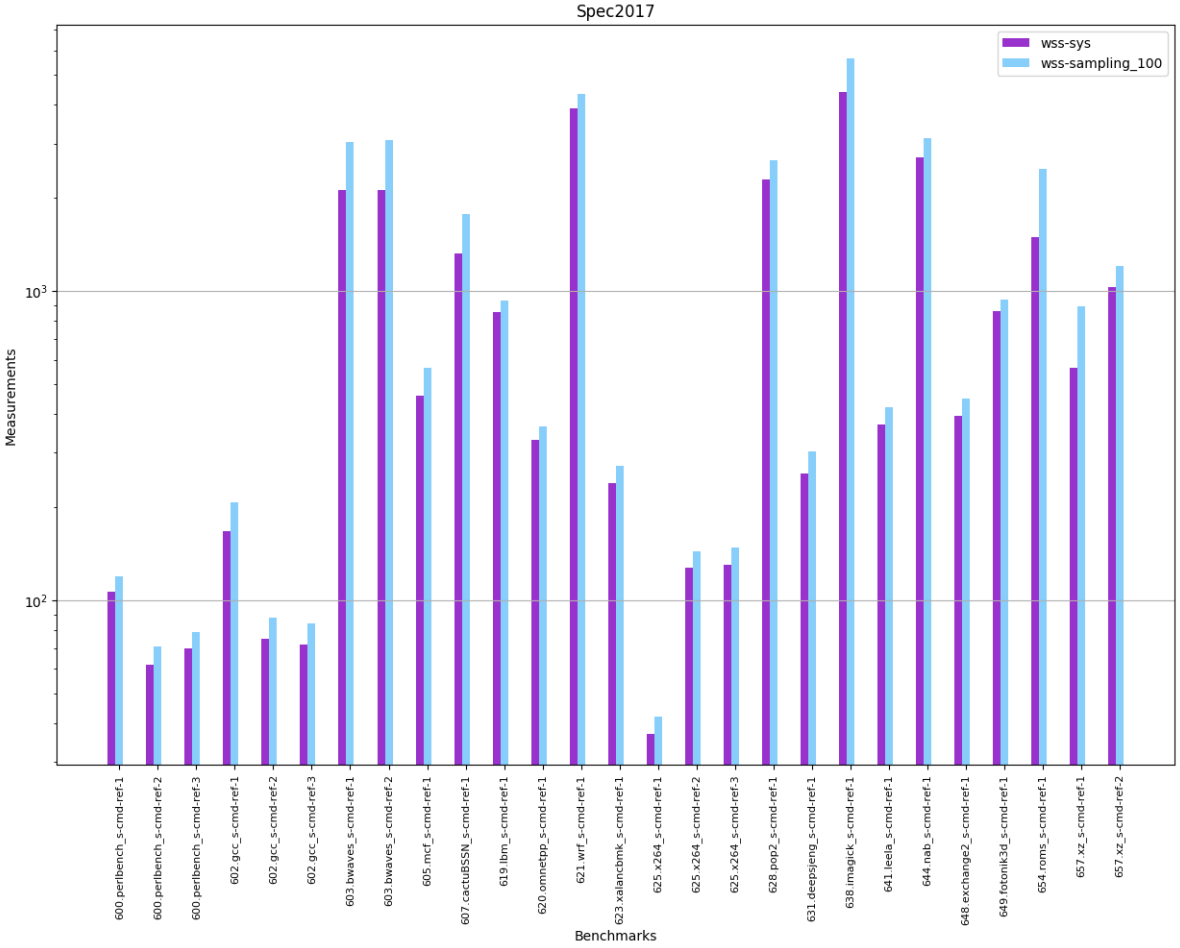
Σχήμα 5.121: 657.xz_s-ref-1[WSS, samples=100, interval=1s, user level]



Σχήμα 5.122: 657.xz_s-ref-2[WSS, samples=100, interval=1s, user level]

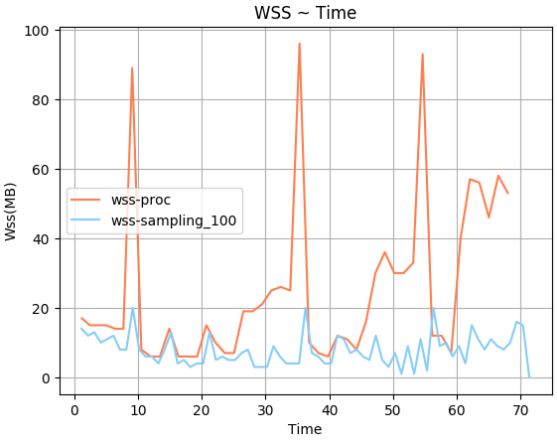


Σχήμα 5.123: 6xx benchmarks[Overheads, samples=100, interval=1s, user level]

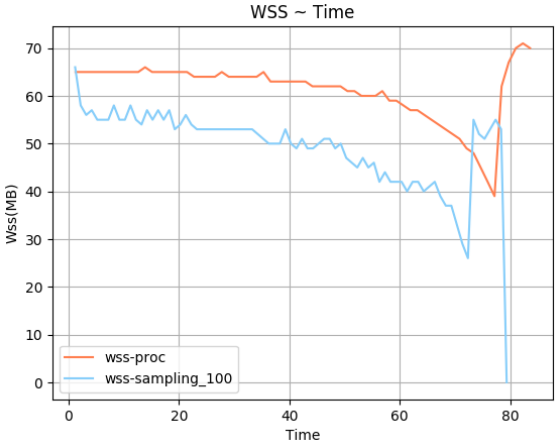


Σχήμα 5.124: 6xx benchmarks[Measurements, samples=100, interval=1s, user level]

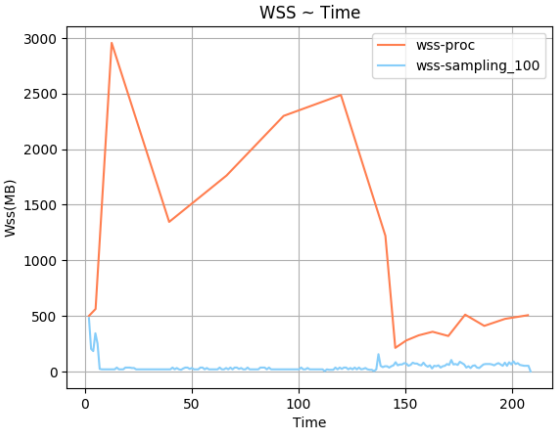
Σχήμα 5.125: 600.perlbench_s-ref-1[WSS, samples=100, interval=1s, user level]



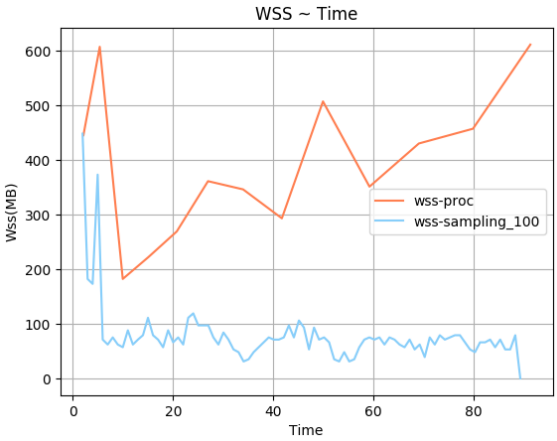
Σχήμα 5.126: 600.perlbench_s-ref-2[WSS, samples=100, interval=1s, user level]



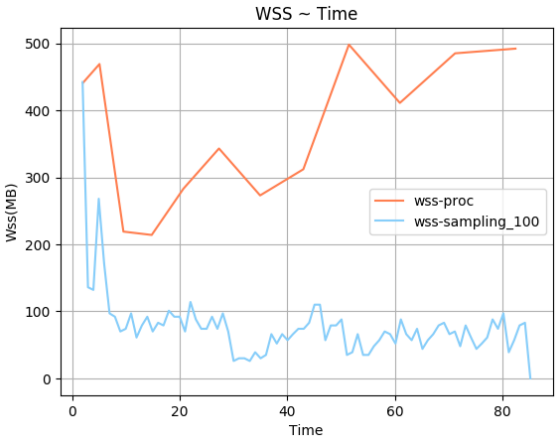
Σχήμα 5.127: 600.perlbench_s-ref-3[WSS, samples=100, interval=1s, user level]



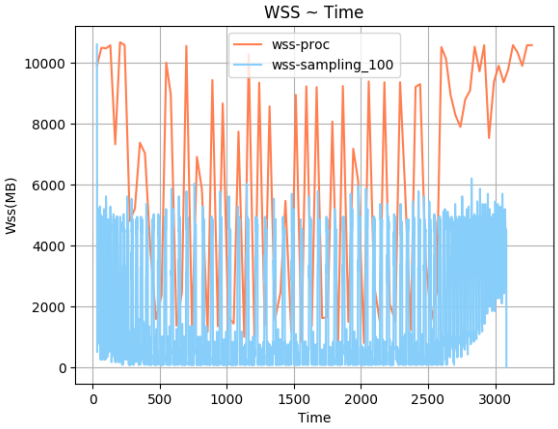
Σχήμα 5.128: 602.gcc_s-ref-1[WSS, samples=100, interval=1s, user level]



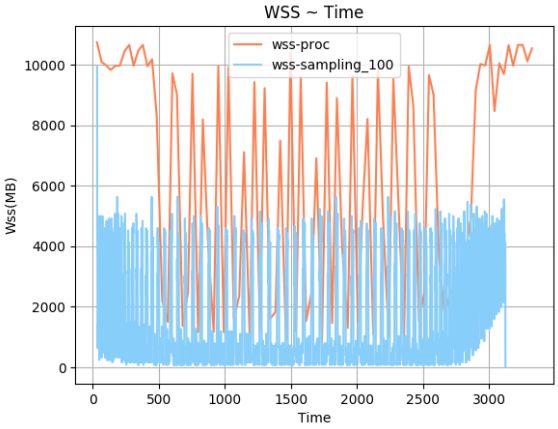
Σχήμα 5.129: 602.gcc_s-ref-2[WSS, samples=100, interval=1s, user level]



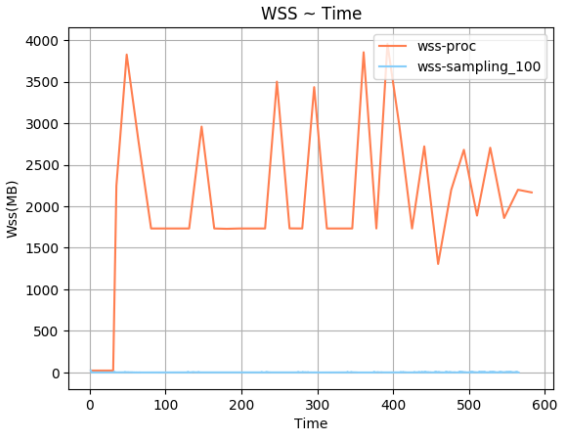
Σχήμα 5.130: 602.gcc_s-ref-3[WSS, samples=100, interval=1s, user level]



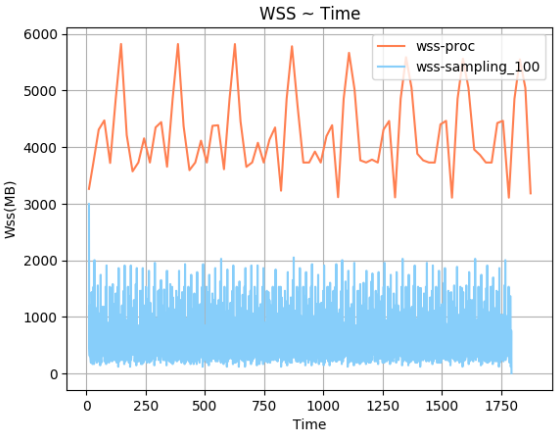
Σχήμα 5.131: 603.bwaves_s-ref-1[WSS, samples=100, interval=1s, user level]



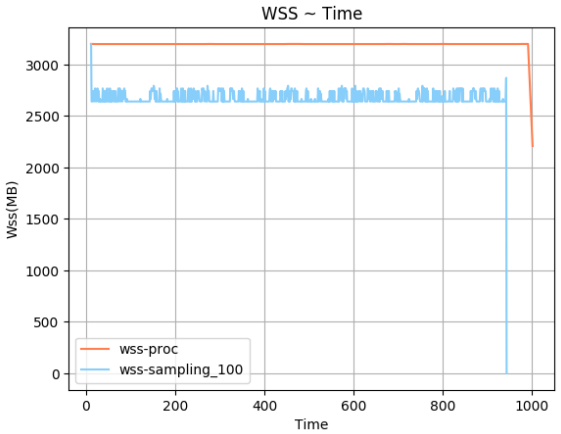
Σχήμα 5.132: 603.bwaves_s-ref-2[WSS, samples=100, interval=1s, user level]



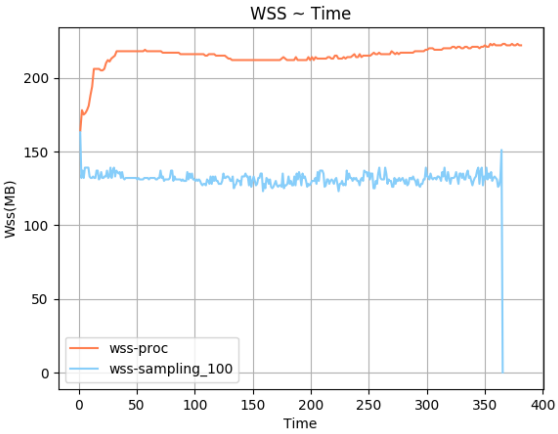
Σχήμα 5.133: 605.mcf_s-ref-1[WSS, samples=100, interval=1s, user level]



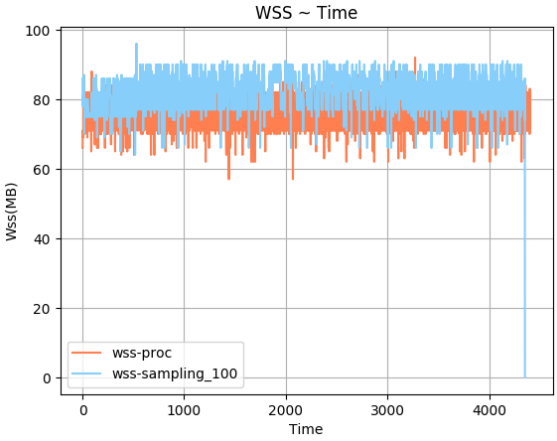
Σχήμα 5.134: 607.cactuBSSN_s-ref-1[WSS, samples=100, interval=1s, user level]



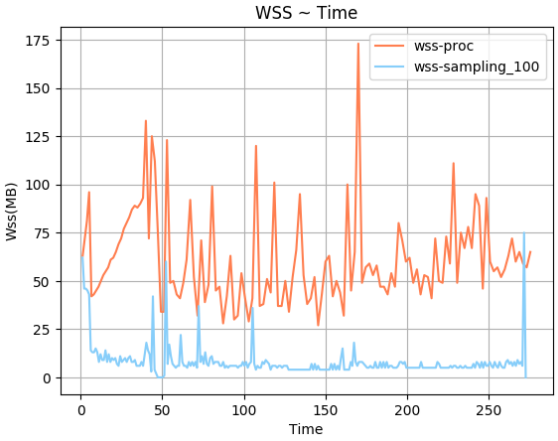
Σχήμα 5.135: 619.lbm_s-ref-1[WSS, samples=100, interval=1s, user level]



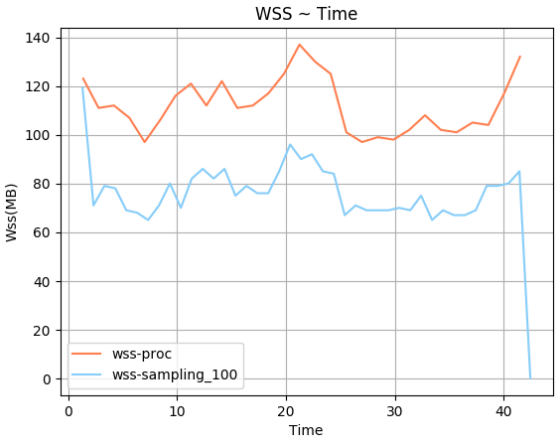
Σχήμα 5.136: 620.omnetpp_s-ref-1[WSS, samples=100, interval=1s, user level]



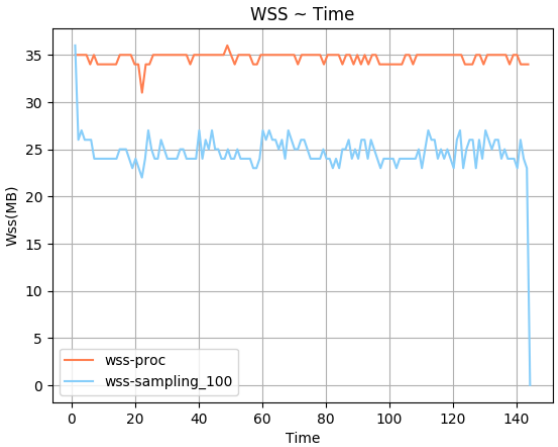
Σχήμα 5.137: 621.wrf_s-ref-1[WSS, samples=100, interval=1s, user level]



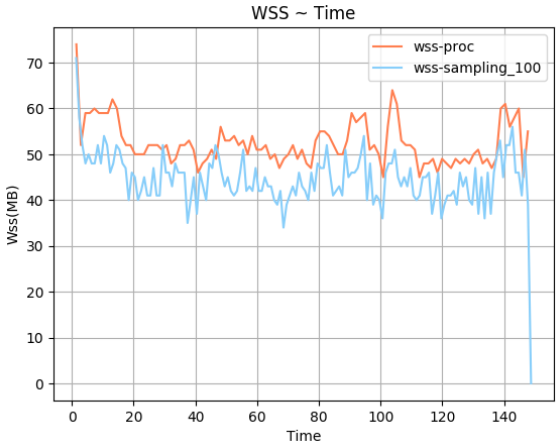
Σχήμα 5.138: 623.xalancbmk_s-ref-1[WSS, samples=100, interval=1s, user level]



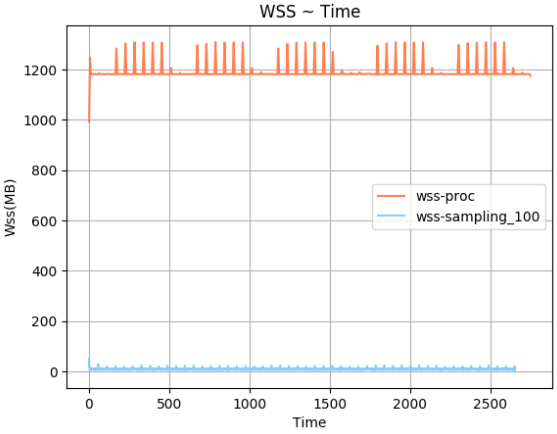
Σχήμα 5.139: 625.x264_s-ref-1[WSS, samples=100, interval=1s, user level]



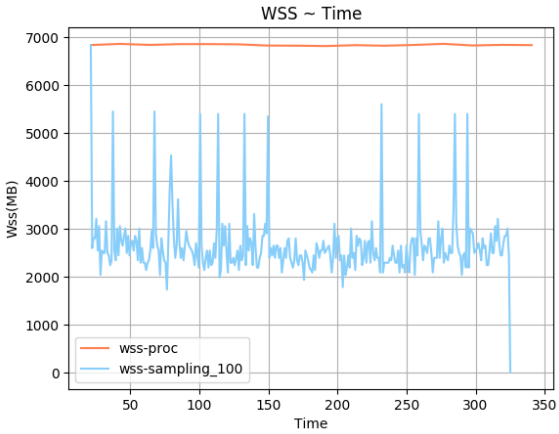
Σχήμα 5.140: 625.x264_s-ref-2[WSS, samples=100, interval=1s, user level]



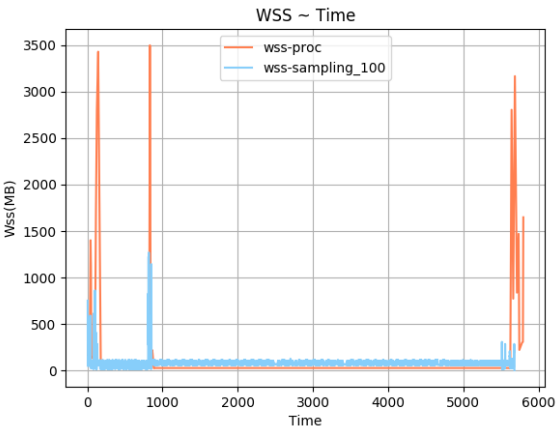
Σχήμα 5.141: 625.x264_s-ref-3[WSS, samples=100, interval=1s, user level]



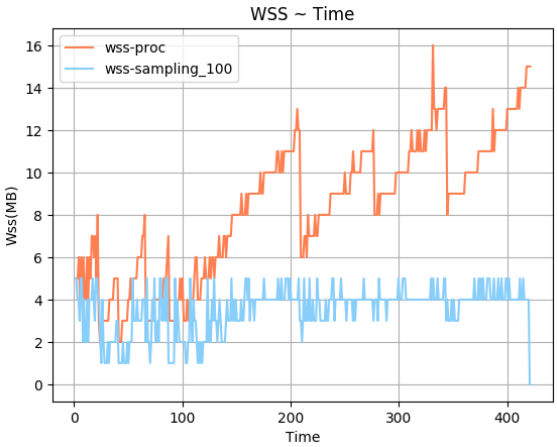
Σχήμα 5.142: 628.pop2_s-ref-1[WSS, samples=100, interval=1s, user level]



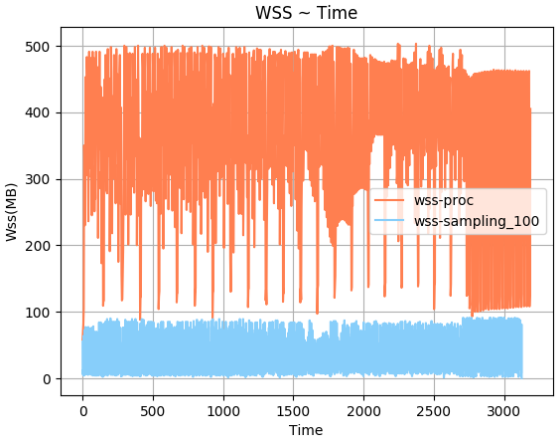
Σχήμα 5.143: 631.deepsjeng_s-ref-1[WSS, samples=100, interval=1s, user level]



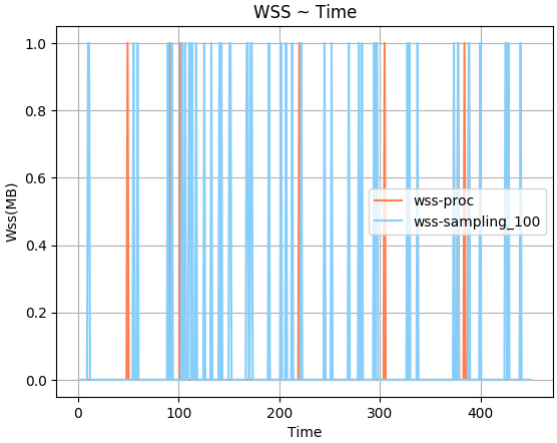
Σχήμα 5.144: 638.imagick_s-ref-1[WSS, samples=100, interval=1s, user level]



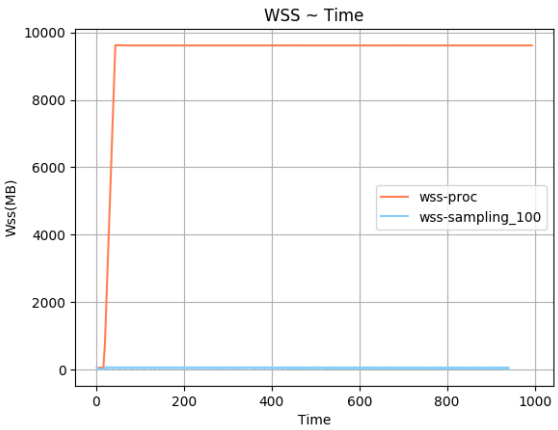
Σχήμα 5.145: 641.leela_s-ref-1[WSS, samples=100, interval=1s, user level]



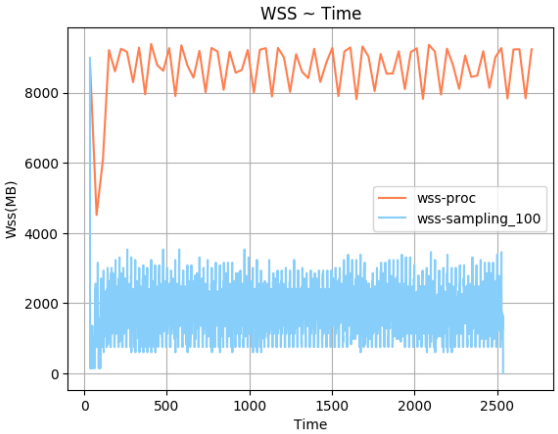
Σχήμα 5.146: 644.nab_s-ref-1[WSS, samples=100, interval=1s, user level]



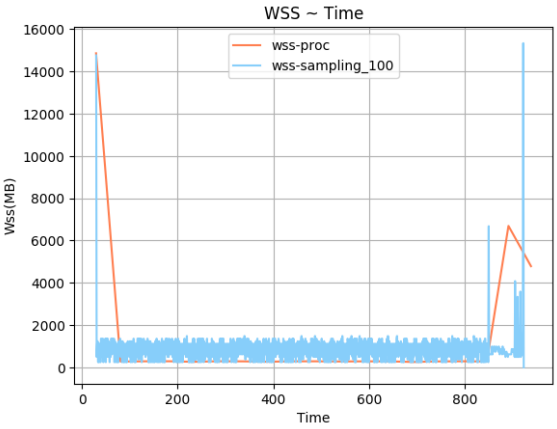
Σχήμα 5.147: 648.exchange2_s-ref-1[WSS, samples=100, interval=1s, user level]



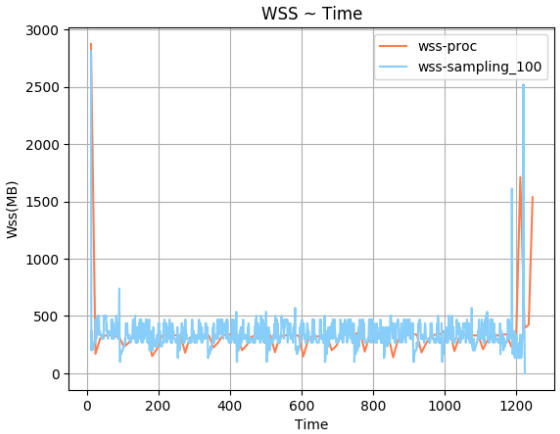
Σχήμα 5.148: 649.fotonik3d_s-ref-1[WSS, samples=100, interval=1s, user level]



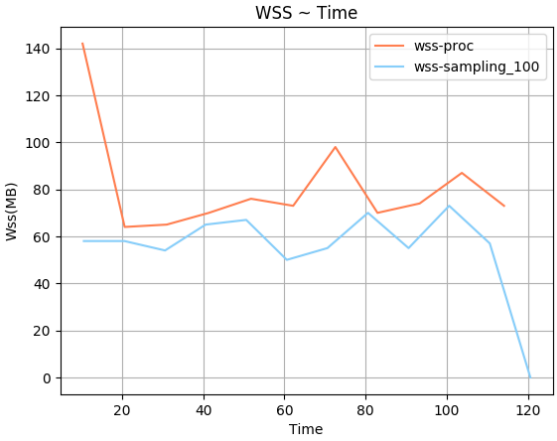
Σχήμα 5.149: 654.roms_s-ref-1[WSS, samples=100, interval=1s, user level]



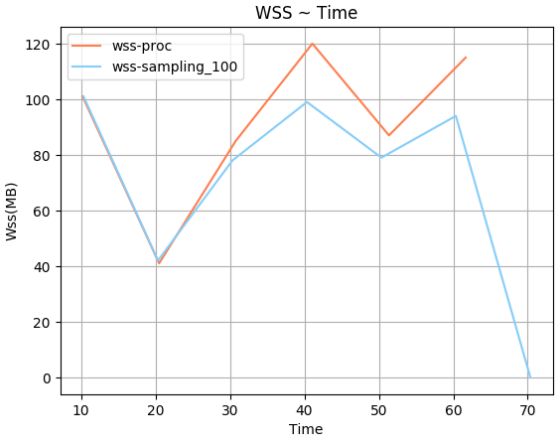
Σχήμα 5.150: 657.xz_s-ref-1[WSS, samples=100, interval=1s, user level]



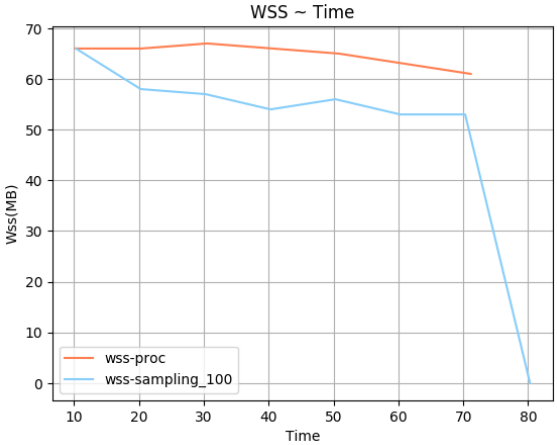
Σχήμα 5.151: 657.xz_s-ref-2[WSS, samples=100, interval=1s, user level]



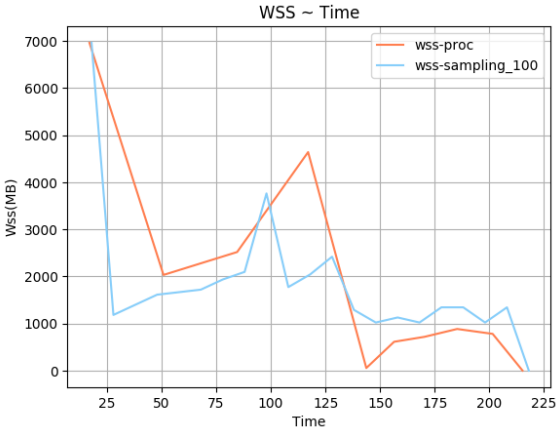
Σχήμα 5.152: 600.perlbench_s-ref-1[WSS, samples=100, interval=10s, user level]



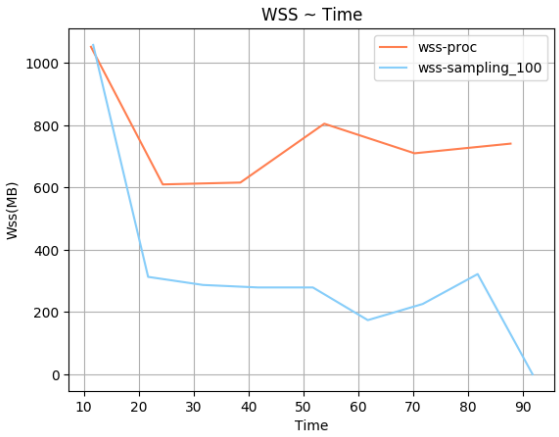
Σχήμα 5.153: 600.perlbench_s-ref-2[WSS, samples=100, interval=10s, user level]



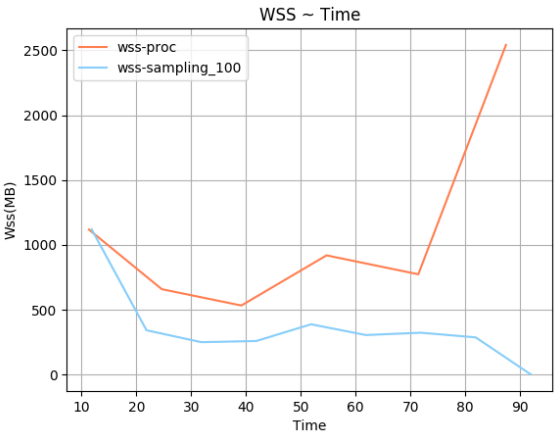
Σχήμα 5.154: 600.perlbench_s-ref-3[WSS, samples=100, interval=10s, user level]



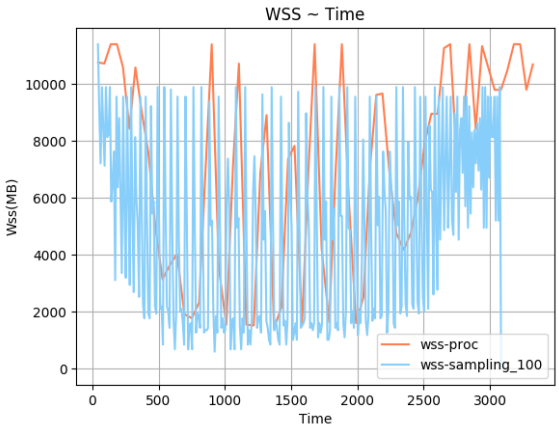
Σχήμα 5.155: 602.gcc_s-ref-1[WSS, samples=100, interval=10s, user level]



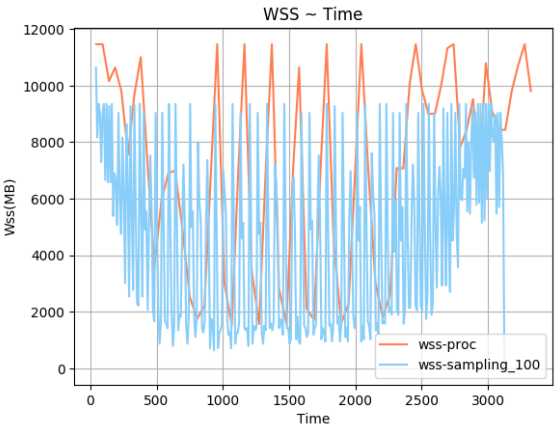
Σχήμα 5.156: 602.gcc_s-ref-2[WSS, samples=100, interval=10s, user level]



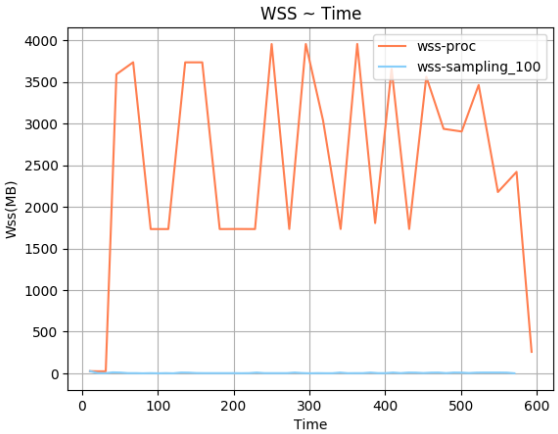
Σχήμα 5.157: 602.gcc_s-ref-3[WSS, samples=100, interval=10s, user level]



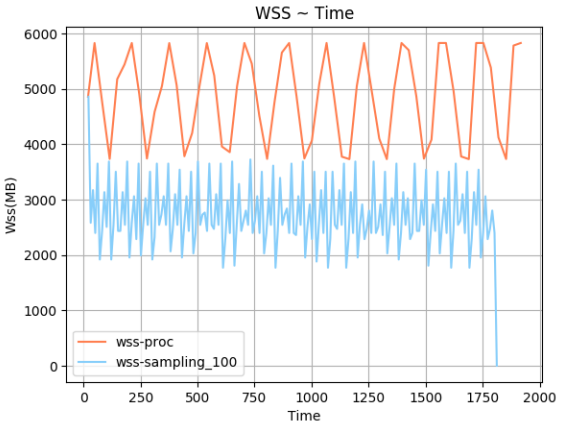
Σχήμα 5.158: 603.bwaves_s-ref-1[WSS, samples=100, interval=10s, user level]



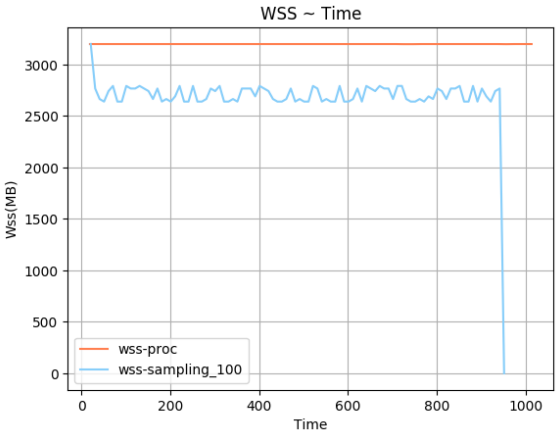
Σχήμα 5.159: 603.bwaves_s-ref-2[WSS, samples=100, interval=10s, user level]



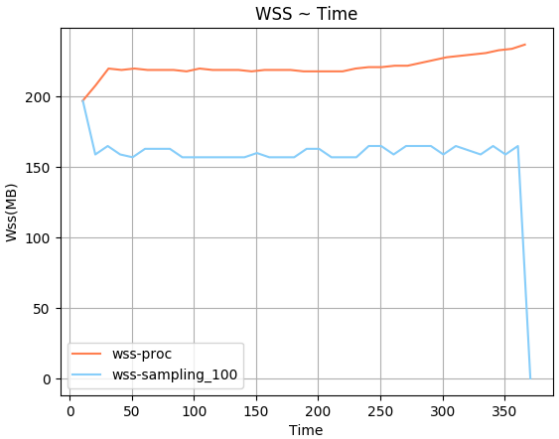
Σχήμα 5.160: 605.mcf_s-ref-1[WSS, samples=100, interval=10s, user level]



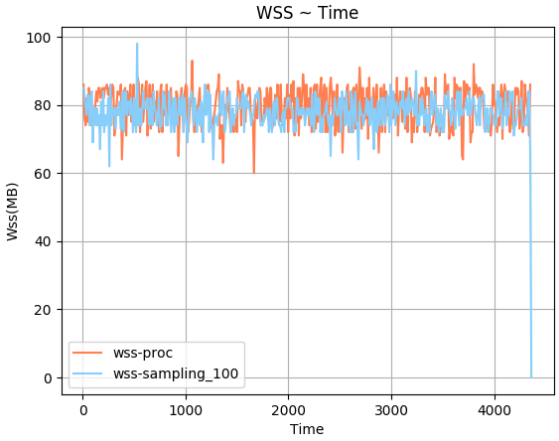
Σχήμα 5.161: 607.cactuBSSN_s-ref-1[WSS, samples=100, interval=10s, user level]



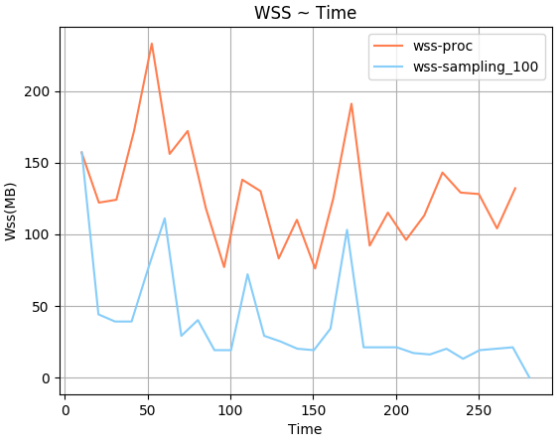
Σχήμα 5.162: 619.lbm_s-ref-1[WSS, samples=100, interval=10s, user level]



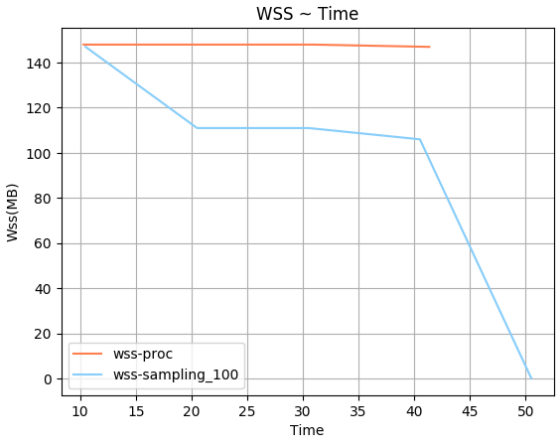
Σχήμα 5.163: 620.omnetpp_s-ref-1[WSS, samples=100, interval=10s, user level]



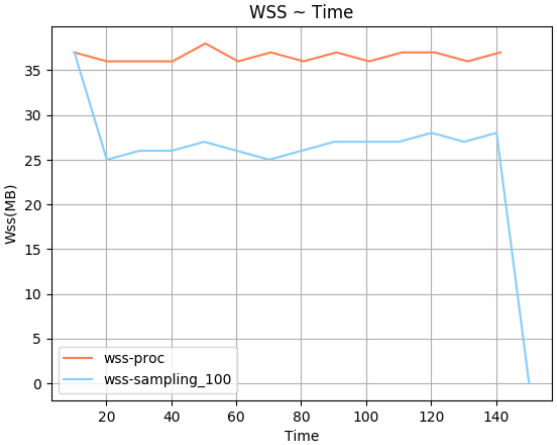
Σχήμα 5.164: 621.wrf_s-ref-1[WSS, samples=100, interval=10s, user level]



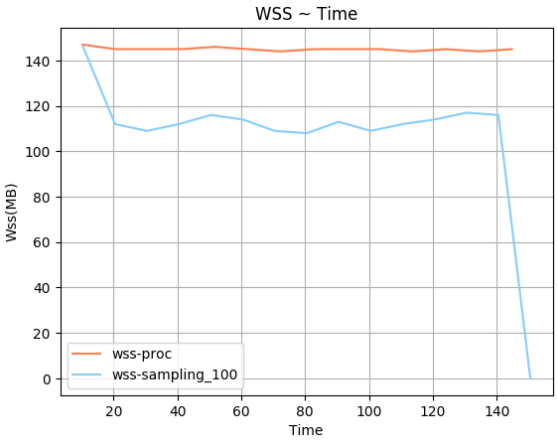
Σχήμα 5.165: 623.xalancbmk_s-ref-1[WSS, samples=100, interval=10s, user level]



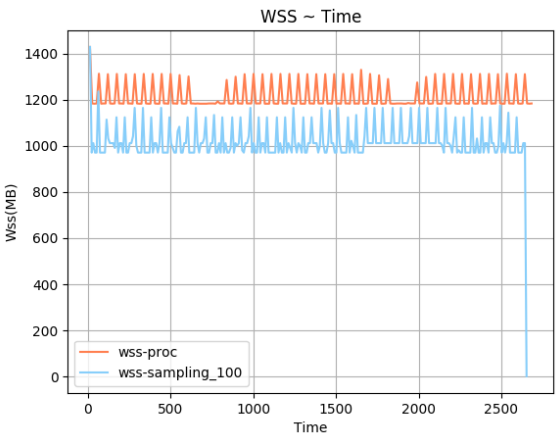
Σχήμα 5.166: 625.x264_s-ref-1[WSS, samples=100, interval=10s, user level]



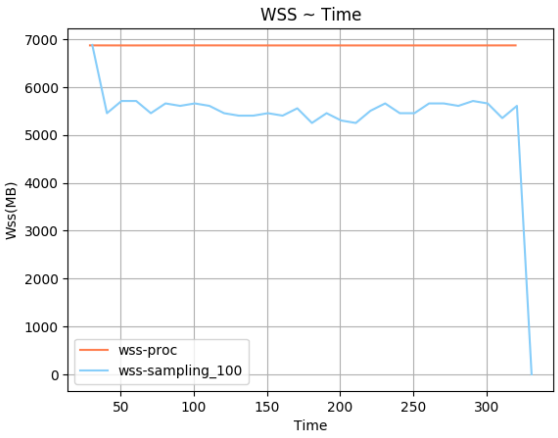
Σχήμα 5.167: 625.x264_s-ref-2[WSS, samples=100, interval=10s, user level]



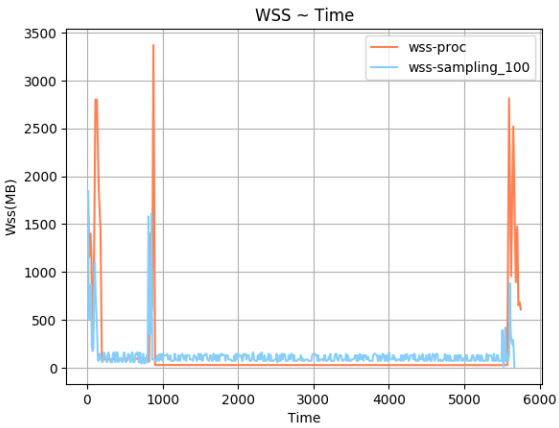
Σχήμα 5.168: 625.x264_s-ref-3[WSS, samples=100, interval=10s, user level]



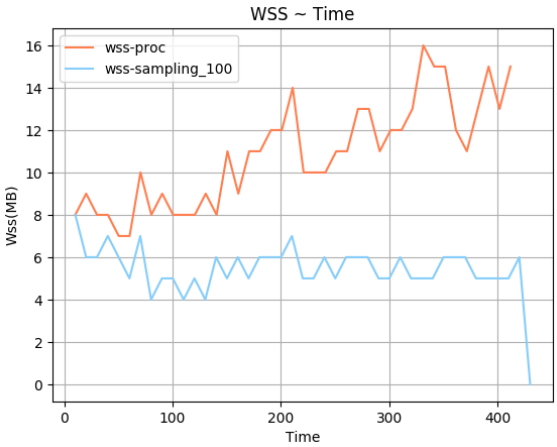
Σχήμα 5.169: 628.pop2_s-ref-1[WSS, samples=100, interval=10s, user level]



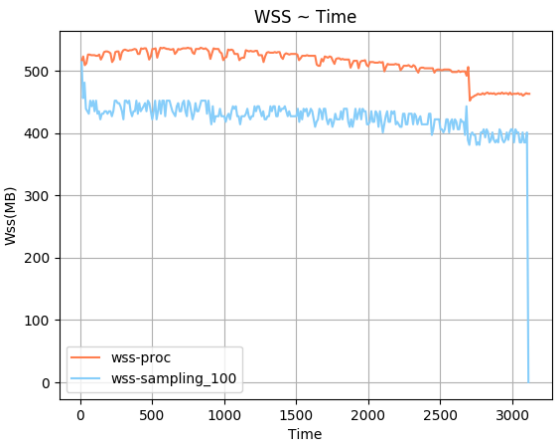
Σχήμα 5.170: 631.deepsjeng_s-ref-1[WSS, samples=100, interval=10s, user level]



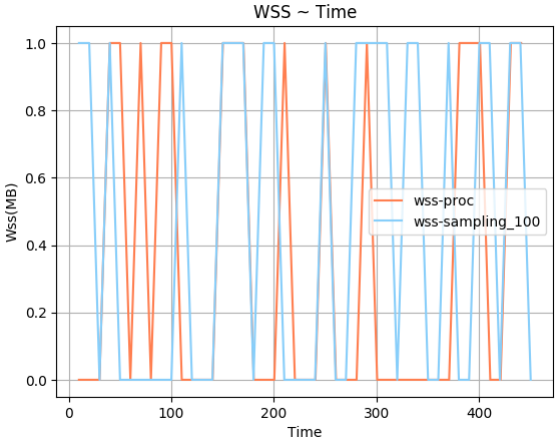
Σχήμα 5.171: 638.imagick_s-ref-1[WSS, samples=100, interval=10s, user level]



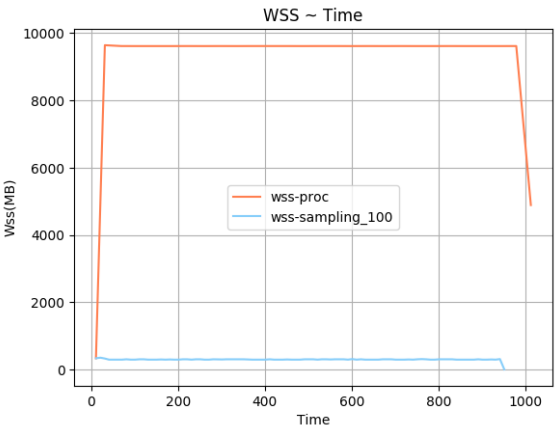
Σχήμα 5.172: 641.leela_s-ref-1[WSS, samples=100, interval=10s, user level]



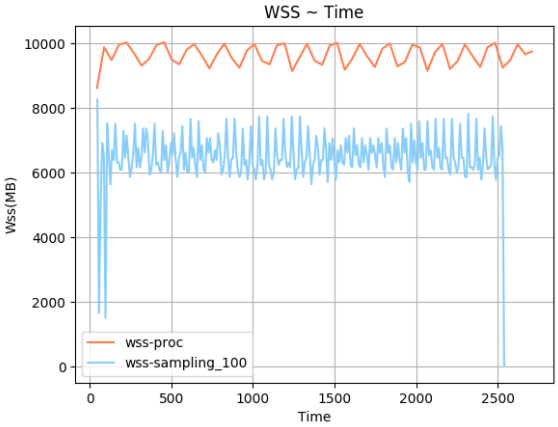
Σχήμα 5.173: 644.nab_s-ref-1[WSS, samples=100, interval=10s, user level]



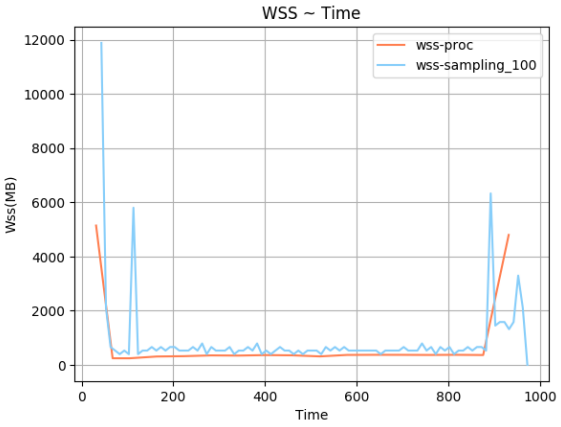
Σχήμα 5.174: 648.exchange2_s-ref-1[WSS, samples=100, interval=10s, user level]



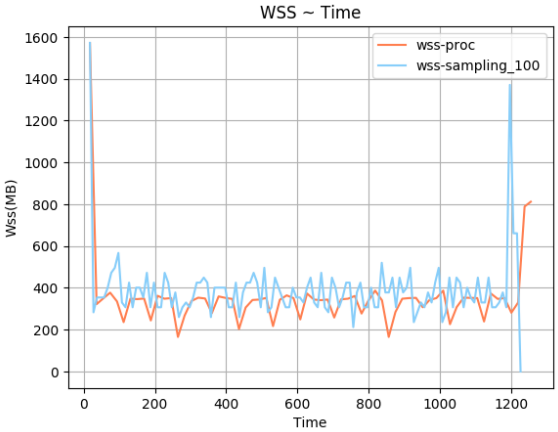
Σχήμα 5.175: 649.fotonik3d_s-ref-1[WSS, samples=100, interval=10s, user level]



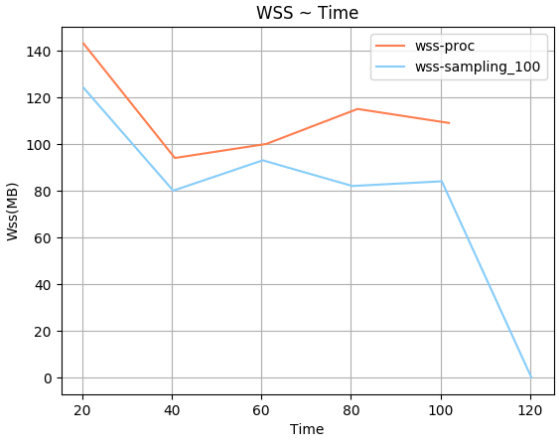
Σχήμα 5.176: 654.roms_s-ref-1[WSS, samples=100, interval=10s, user level]



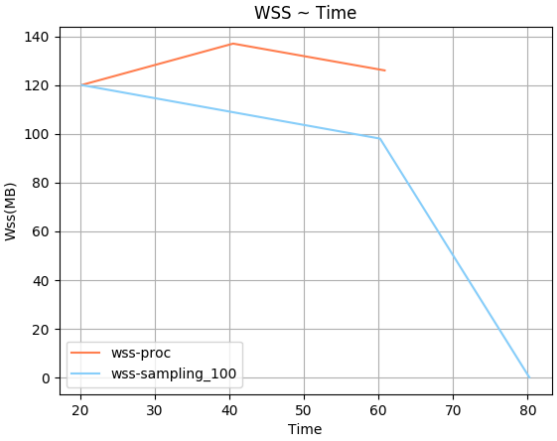
Σχήμα 5.177: 657.xz_s-ref-1[WSS, samples=100, interval=10s, user level]



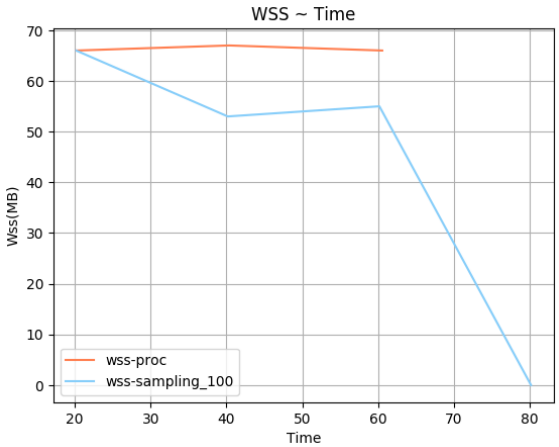
Σχήμα 5.178: 657.xz_s-ref-2[WSS, samples=100, interval=10s, user level]



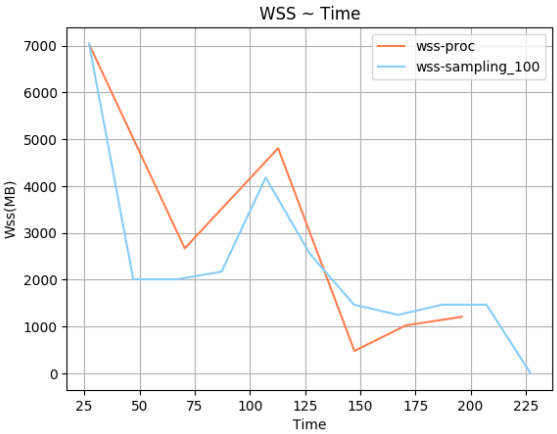
Σχήμα 5.179: 600.perlbench_s-ref-1[WSS, samples=100, interval=20s, user level]



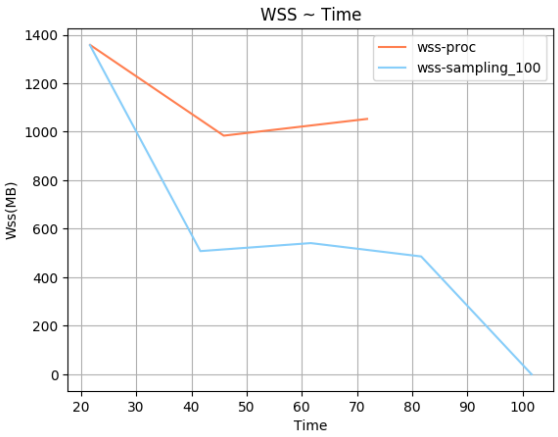
Σχήμα 5.180: 600.perlbench_s-ref-2[WSS, samples=100, interval=20s, user level]



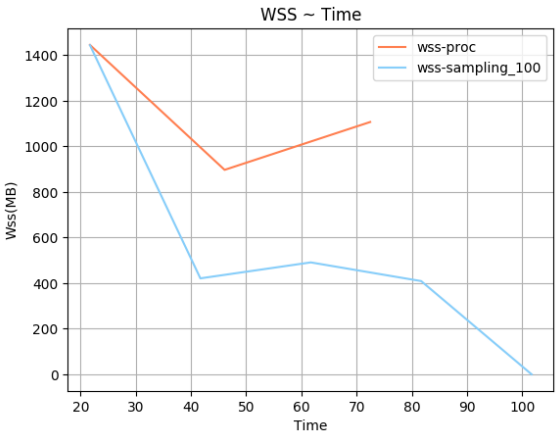
Σχήμα 5.181: 600.perlbench_s-ref-3[WSS, samples=100, interval=20s, user level]



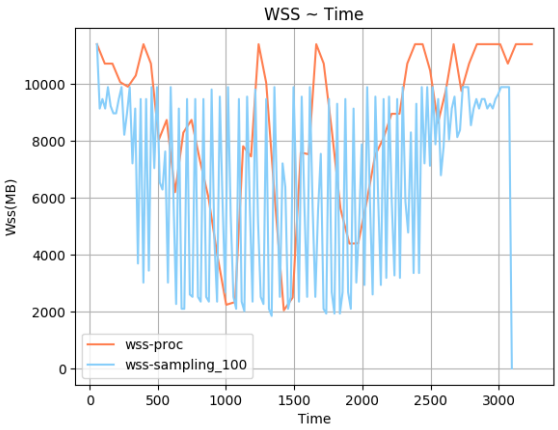
Σχήμα 5.182: 602.gcc_s-ref-1[WSS, samples=100, interval=20s, user level]



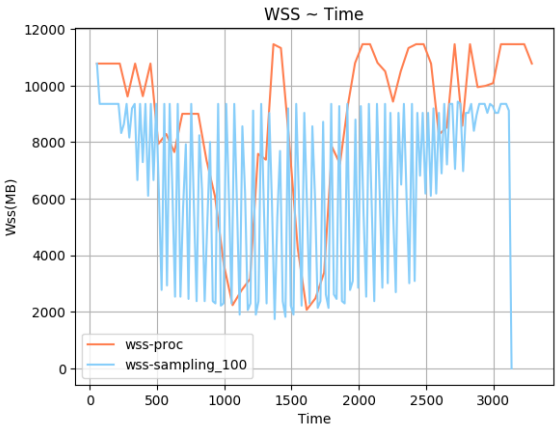
Σχήμα 5.183: 602.gcc_s-ref-2[WSS, samples=100, interval=20s, user level]



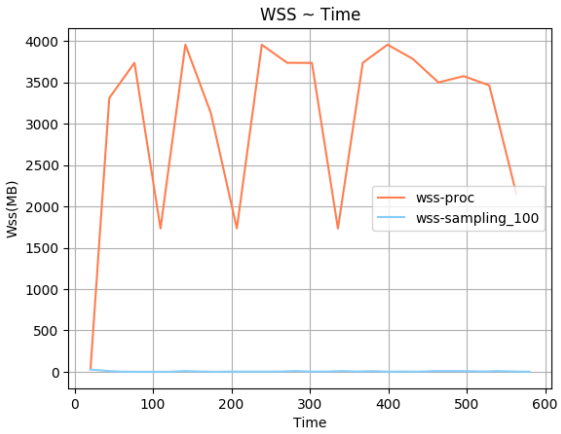
Σχήμα 5.184: 602.gcc_s-ref-3[WSS, samples=100, interval=20s, user level]



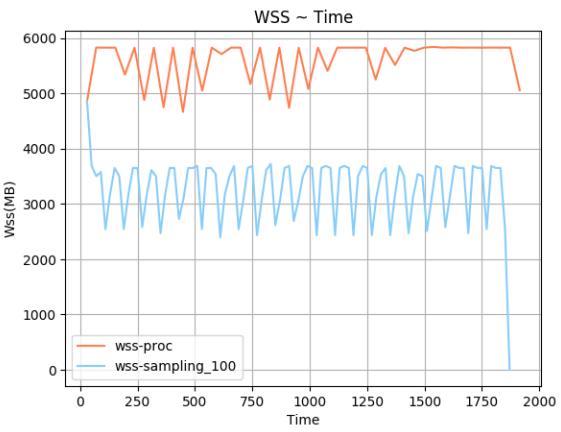
Σχήμα 5.185: 603.bwaves_s-ref-1[WSS, samples=100, interval=20s, user level]



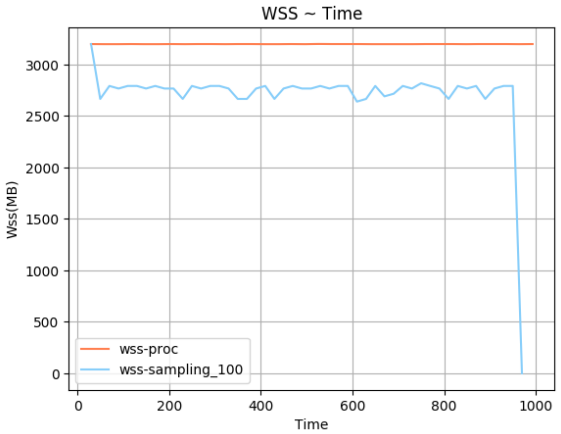
Σχήμα 5.186: 603.bwaves_s-ref-2[WSS, samples=100, interval=20s, user level]



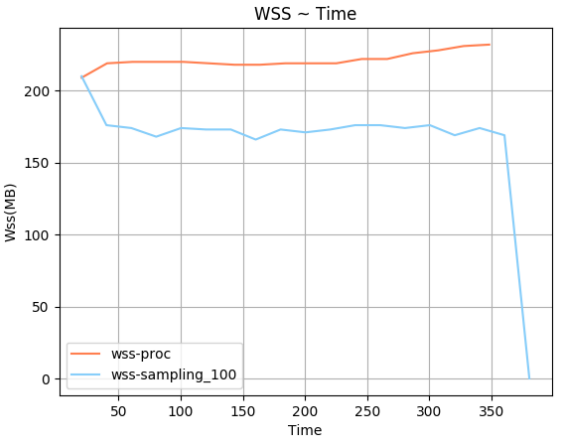
Σχήμα 5.187: 605.mcf_s-ref-1[WSS, samples=100, interval=20s, user level]



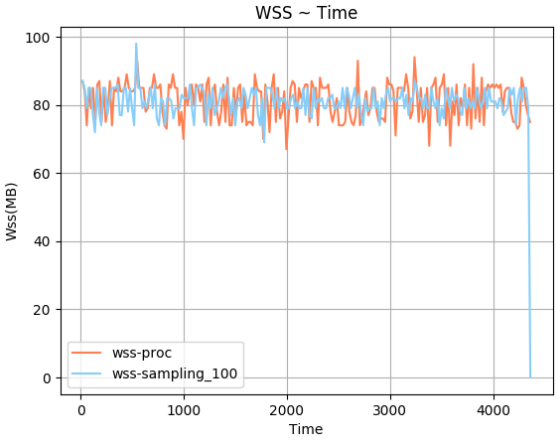
Σχήμα 5.188: 607.cactuBSSN_s-ref-1[WSS, samples=100, interval=20s, user level]



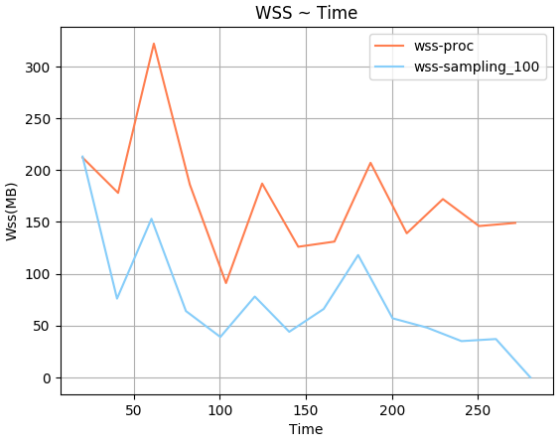
Σχήμα 5.189: 619.lbm_s-ref-1[WSS, samples=100, interval=20s, user level]



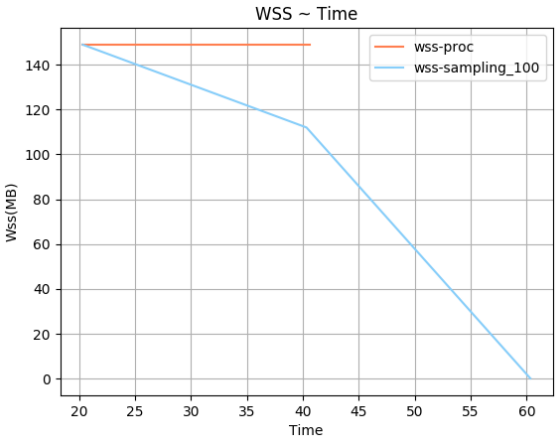
Σχήμα 5.190: 620.omnetpp_s-ref-1[WSS, samples=100, interval=20s, user level]



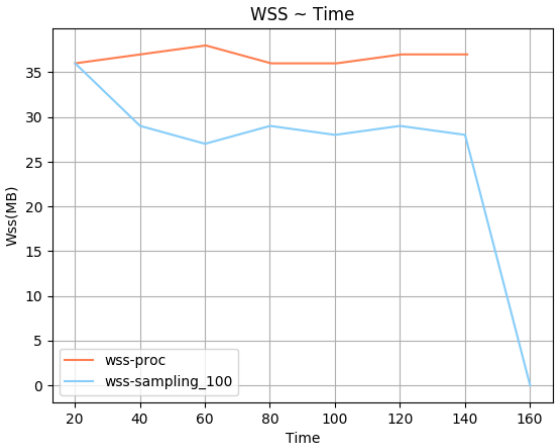
Σχήμα 5.191: 621.wrf_s-ref-1[WSS, samples=100, interval=20s, user level]



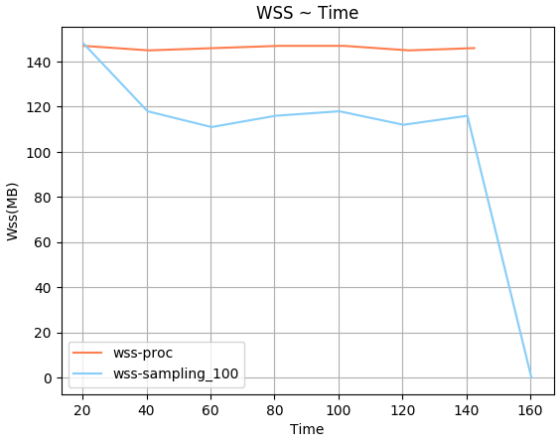
Σχήμα 5.192: 623.xalancbmk_s-ref-1[WSS, samples=100, interval=20s, user level]



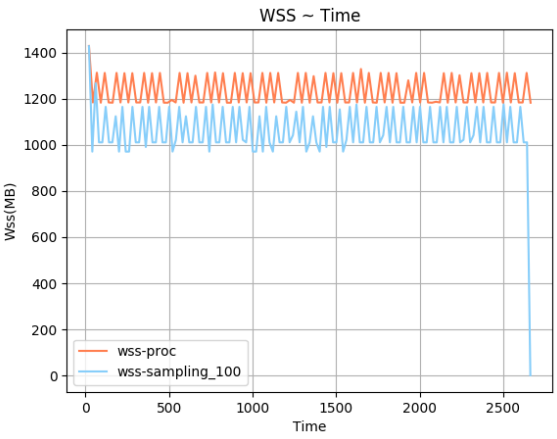
Σχήμα 5.193: 625.x264_s-ref-1[WSS, samples=100, interval=20s, user level]



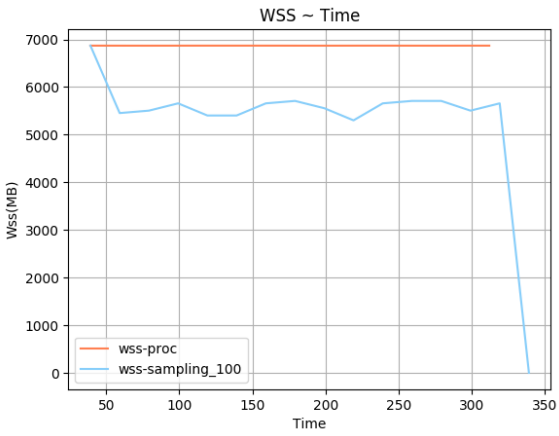
Σχήμα 5.194: 625.x264_s-ref-2[WSS, samples=100, interval=20s, user level]



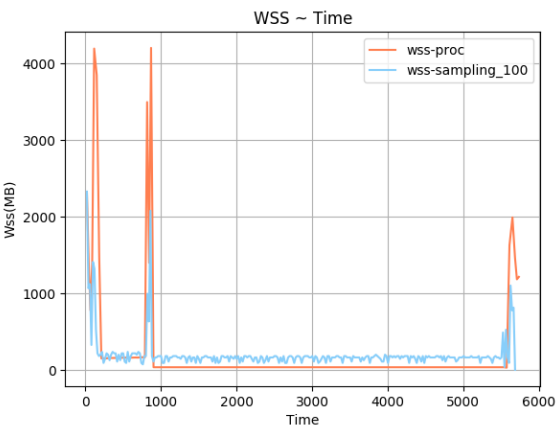
Σχήμα 5.195: 625.x264_s-ref-3[WSS, samples=100, interval=20s, user level]



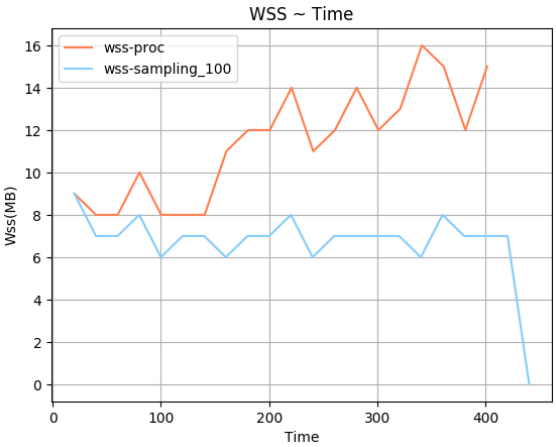
Σχήμα 5.196: 628.pop2_s-ref-1[WSS, samples=100, interval=20s, user level]



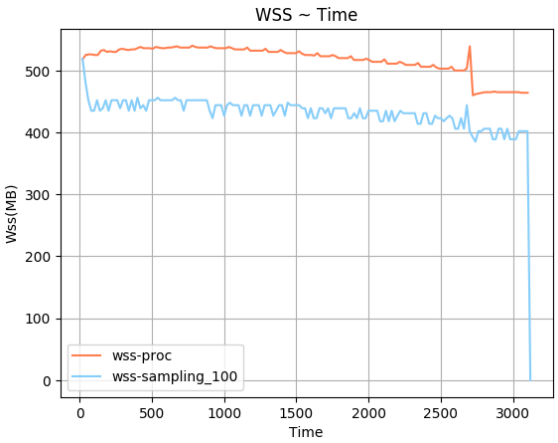
Σχήμα 5.197: 631.deepsjeng_s-ref-1[WSS, samples=100, interval=20s, user level]



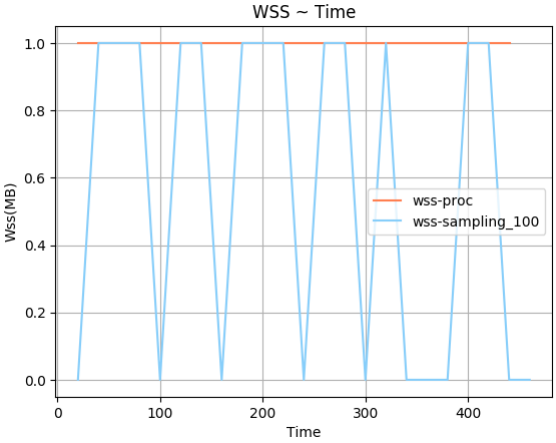
Σχήμα 5.198: 638.imagick_s-ref-1[WSS, samples=100, interval=20s, user level]



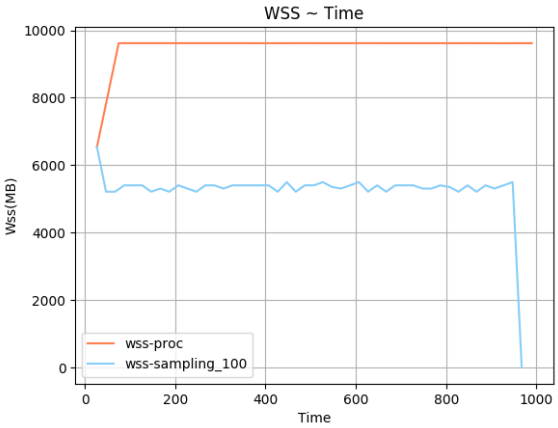
Σχήμα 5.199: 641.leela_s-ref-1[WSS, samples=100, interval=20s, user level]



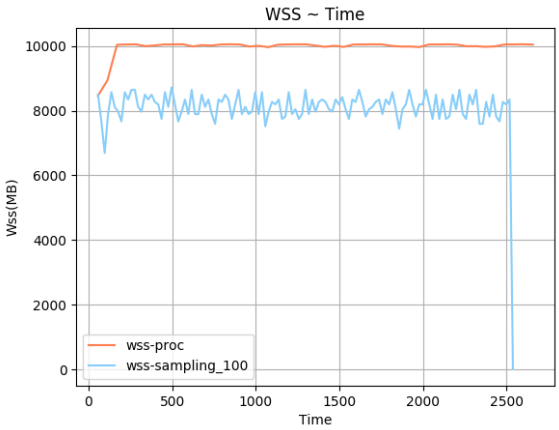
Σχήμα 5.200: 644.nab_s-ref-1[WSS, samples=100, interval=20s, user level]



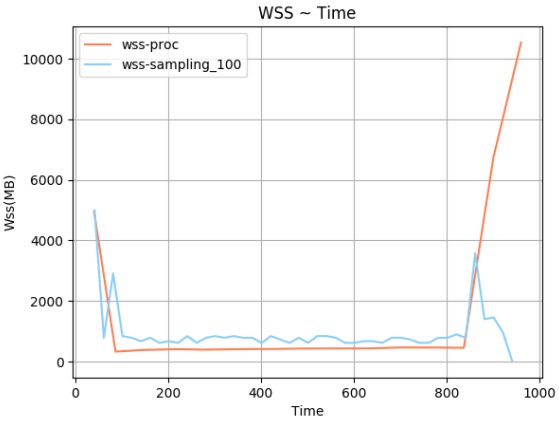
Σχήμα 5.201: 648.exchange2_s-ref-1[WSS, samples=100, interval=20s, user level]



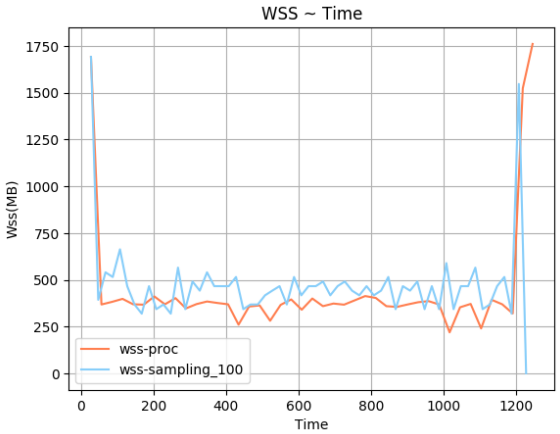
Σχήμα 5.202: 649.fotonik3d_s-ref-1[WSS, samples=100, interval=20s, user level]



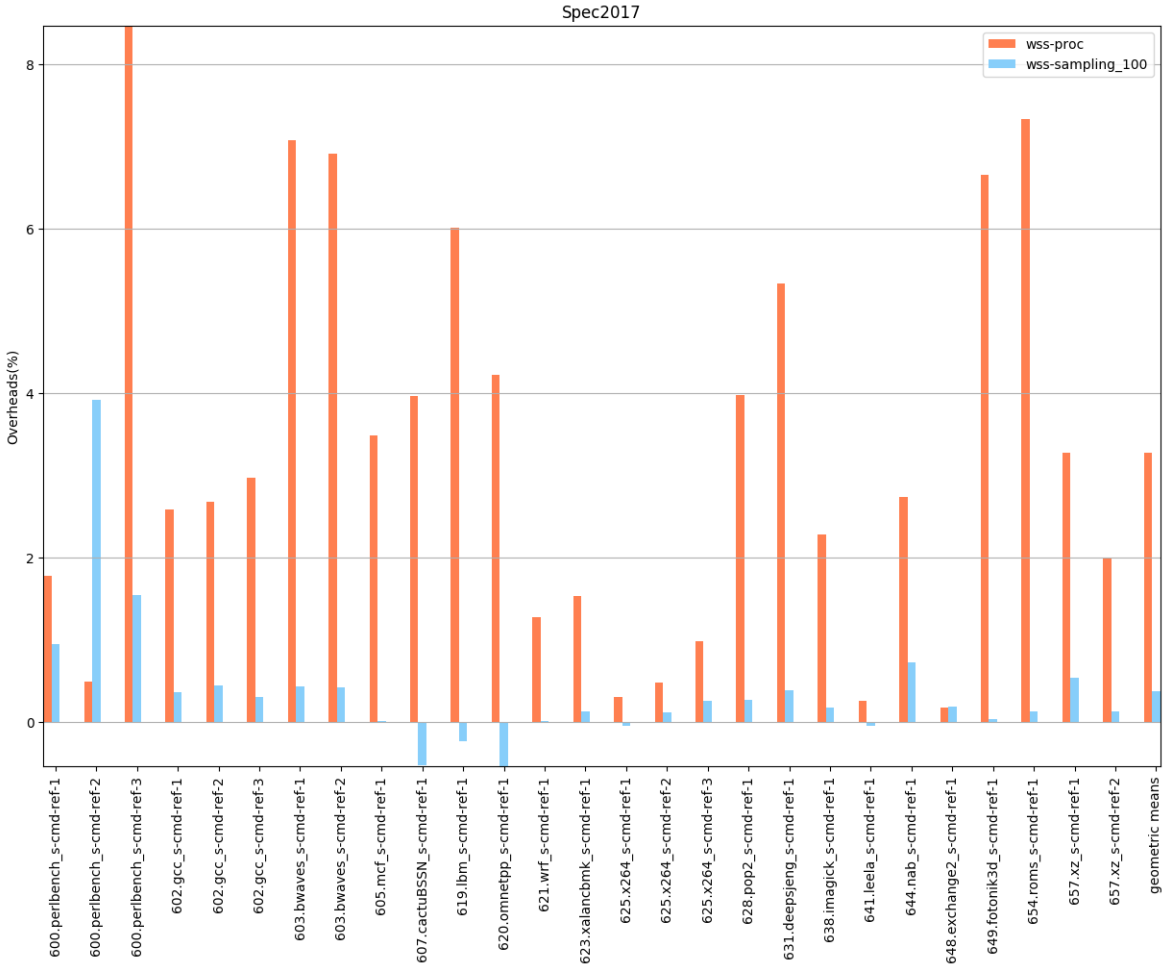
Σχήμα 5.203: 654.roms_s-ref-1[WSS, samples=100, interval=20s, user level]



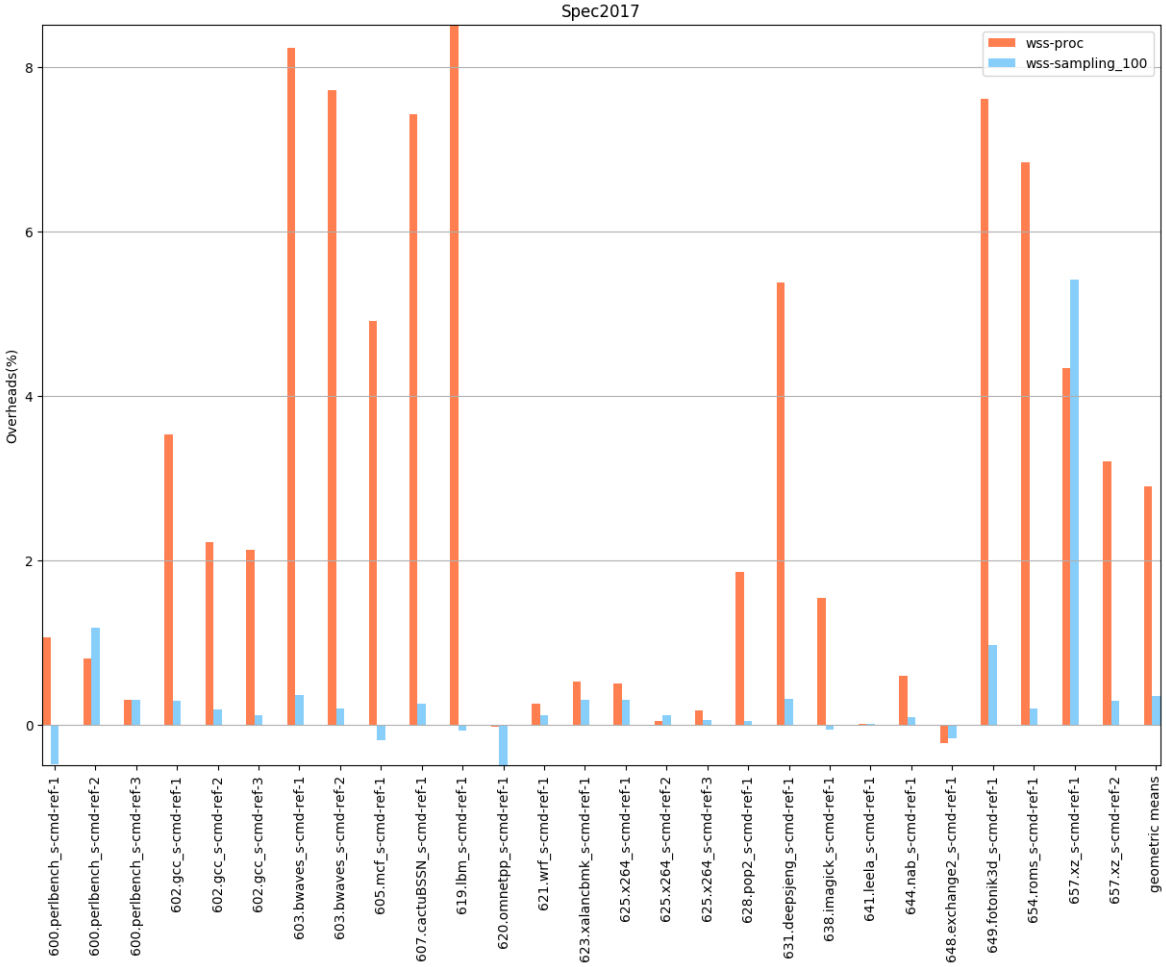
Σχήμα 5.204: 657.xz_s-ref-1[WSS, samples=100, interval=20s, user level]



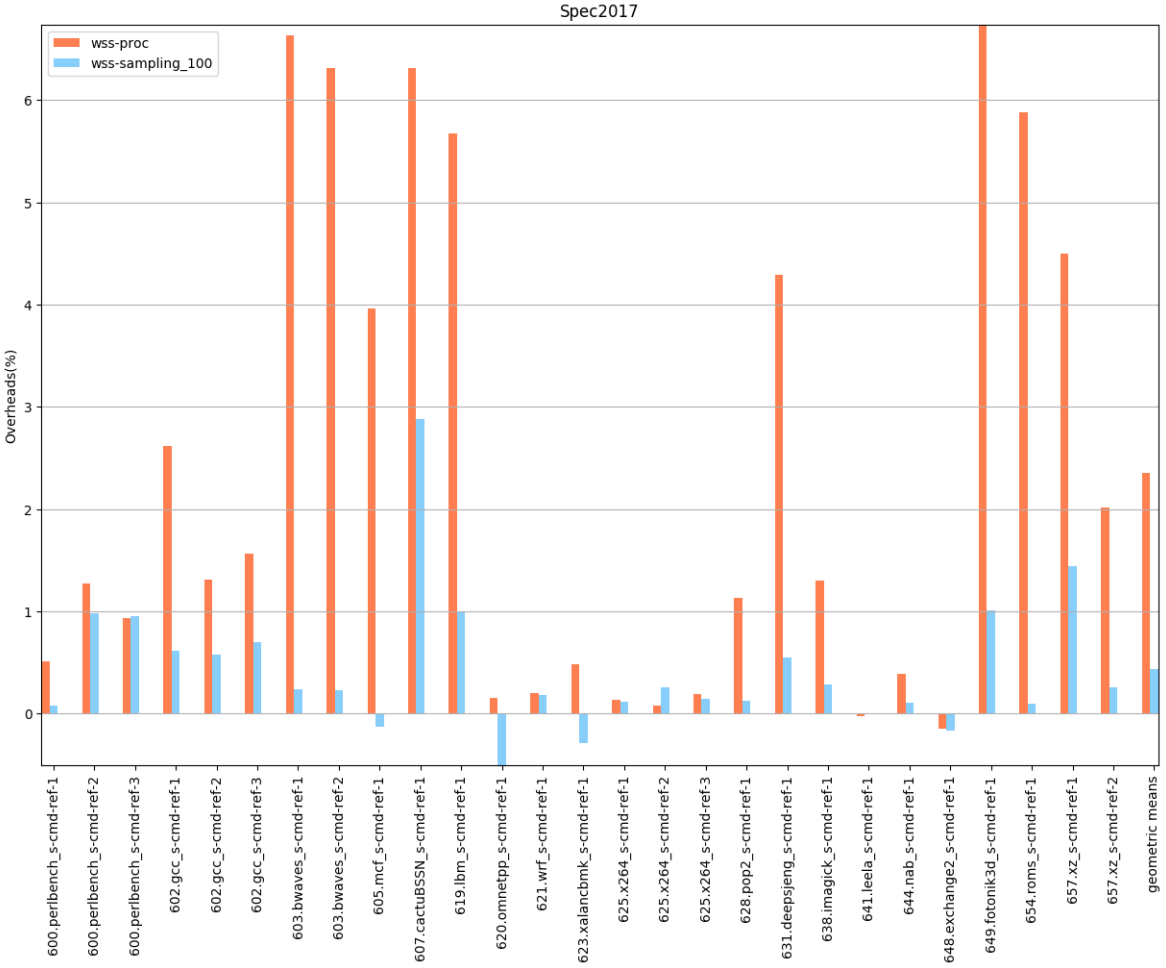
Σχήμα 5.205: 657.xz_s-ref-2[WSS, samples=100, interval=20s, user level]



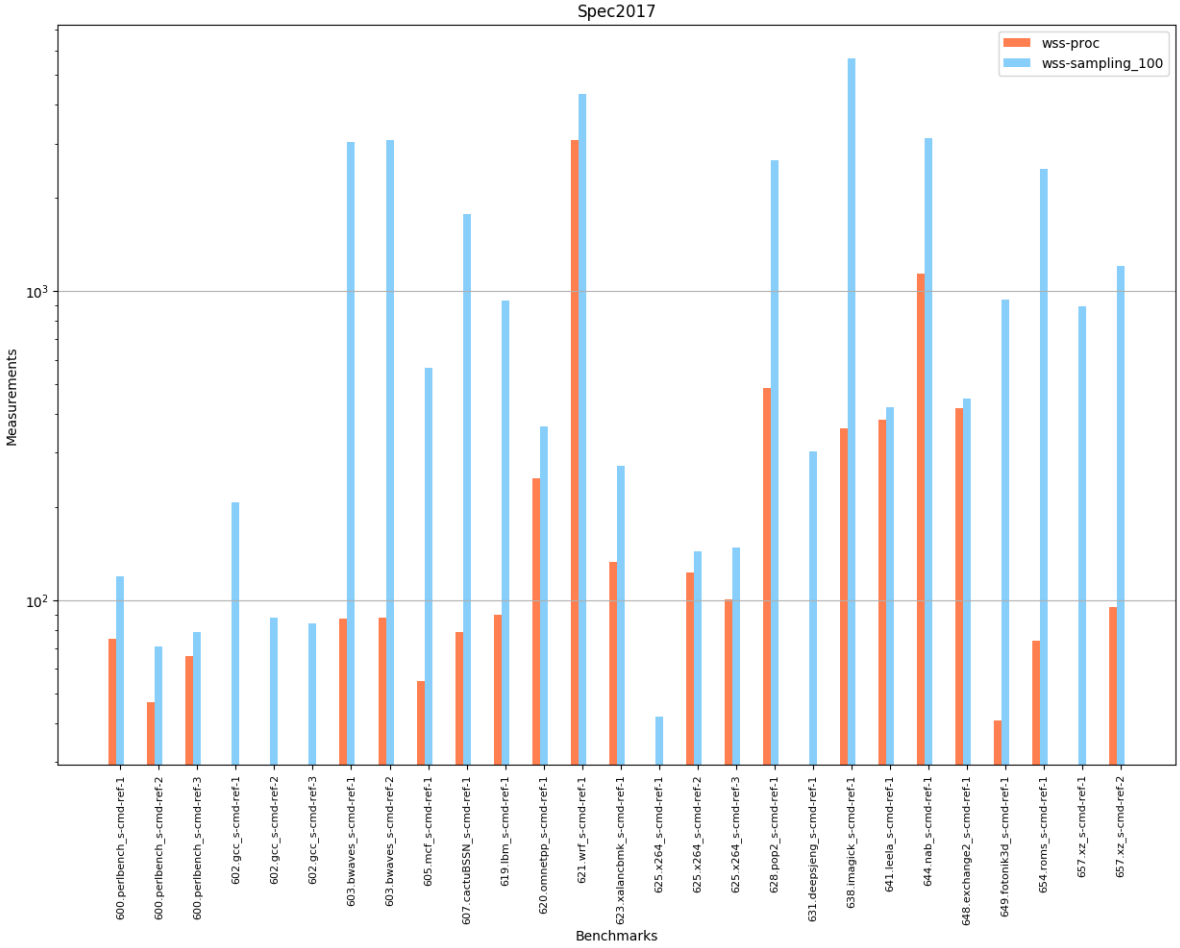
Σχήμα 5.206: 6xx benchmarks[Overheads, samples=100, interval=1s, user level]



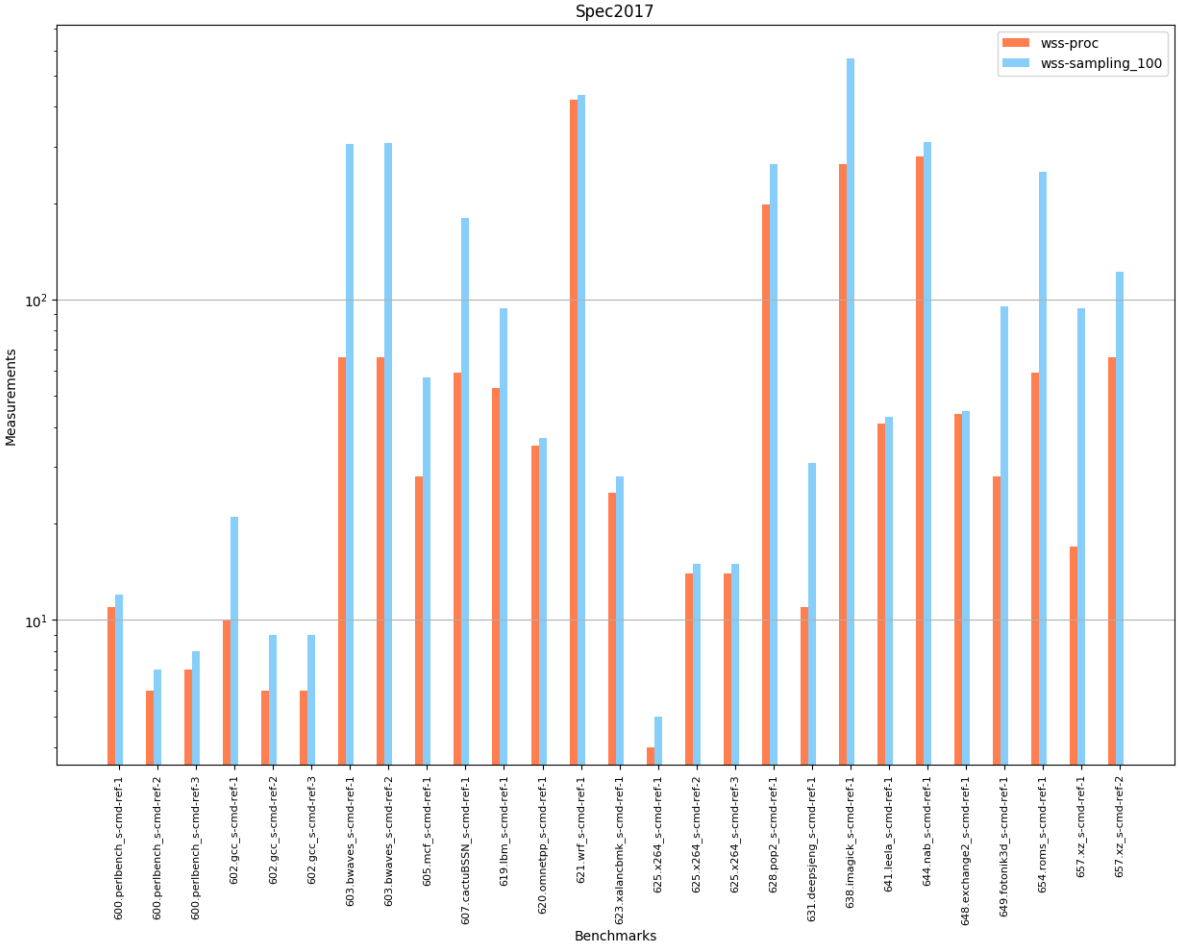
Σχήμα 5.207: 6xx benchmarks[Overheads, samples=100, interval=10s, user level]



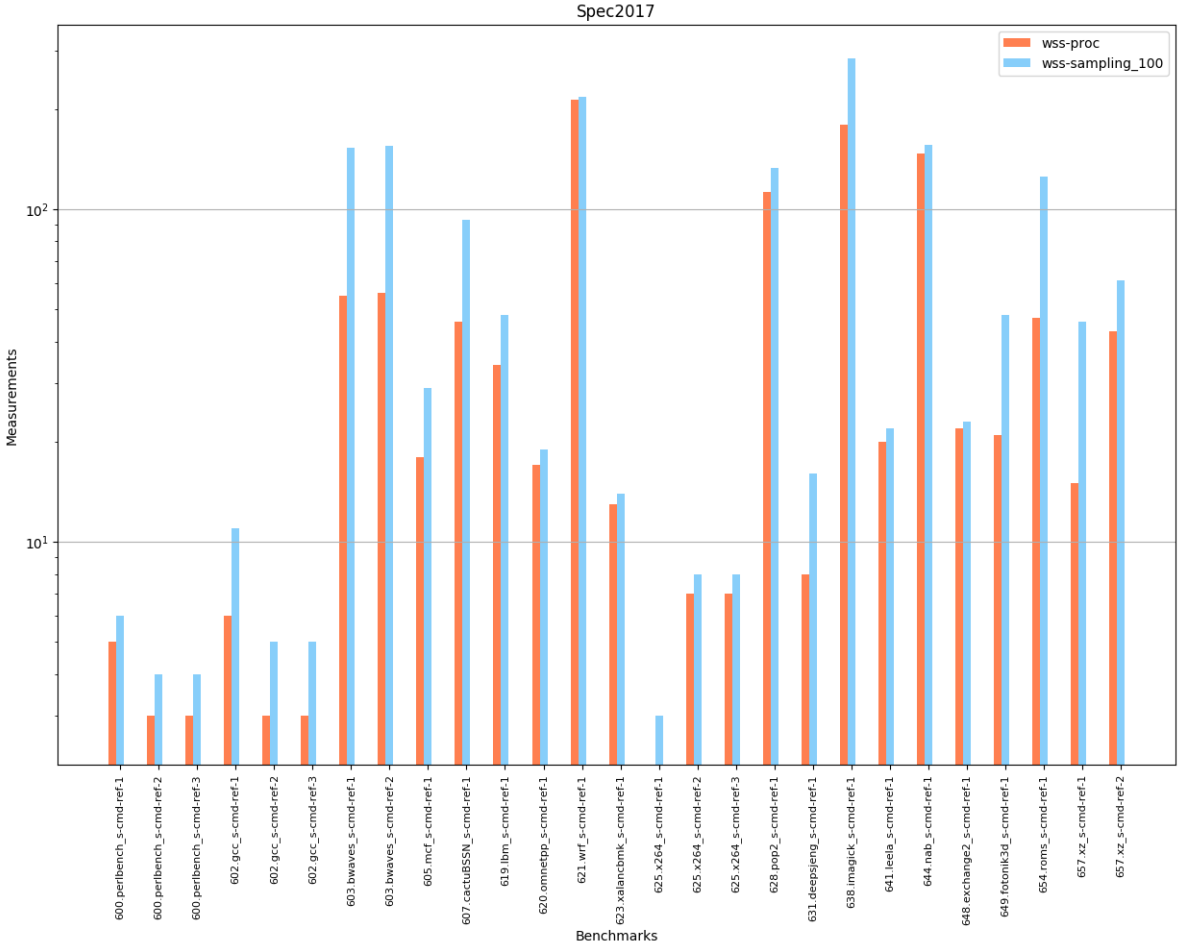
Σχήμα 5.208: 6xx benchmarks[Overheads, samples=100, interval=20s, user level]



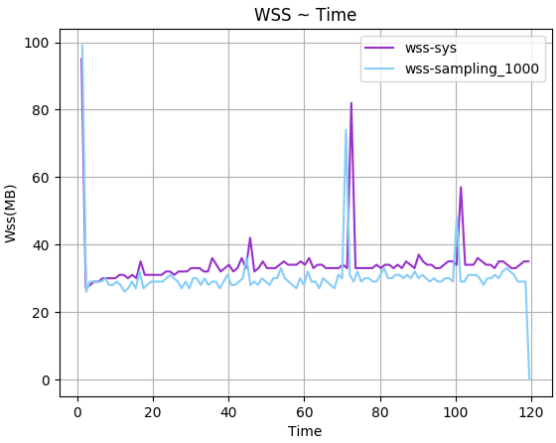
Σχήμα 5.209: 6xx benchmarks[Measurements, samples=100, interval=1s, user level]



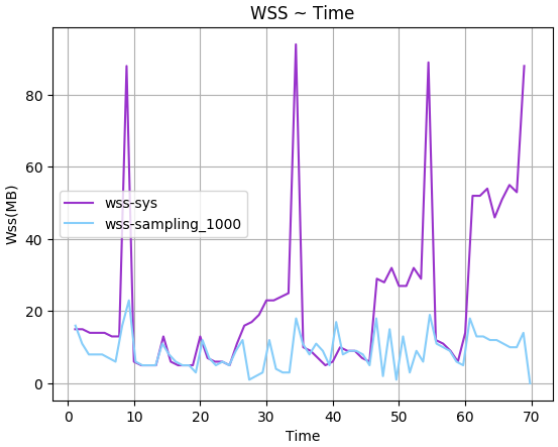
Σχήμα 5.210: 6xx benchmarks[Measurements, samples=100, interval=10s, user level]



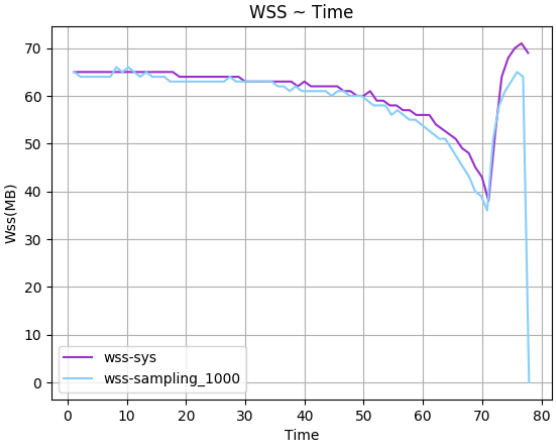
Σχήμα 5.211: 6xx benchmarks[Measurements, samples=100, interval=20s, user level]



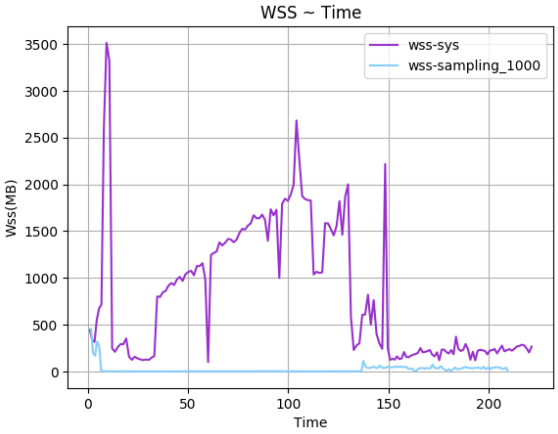
Σχήμα 5.212: 600.perlbench_s-ref-1[WSS, samples=1000, interval=1s, user level]



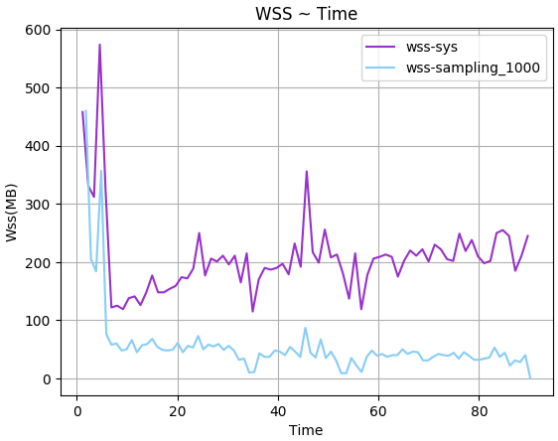
Σχήμα 5.213: 600.perlbench_s-ref-2[WSS, samples=1000, interval=1s, user level]



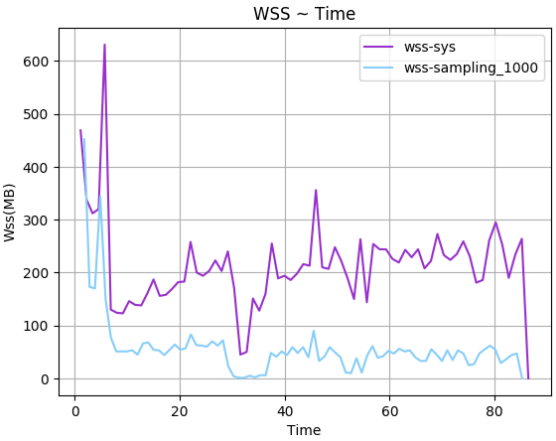
Σχήμα 5.214: 600.perlbench_s-ref-3[WSS, samples=1000, interval=1s, user level]



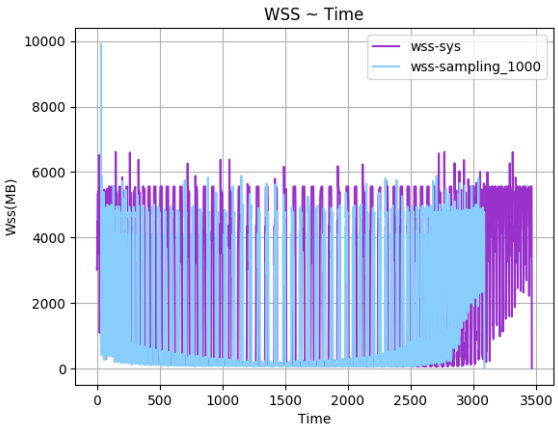
Σχήμα 5.215: 602.gcc_s-ref-1[WSS, samples=1000, interval=1s, user level]



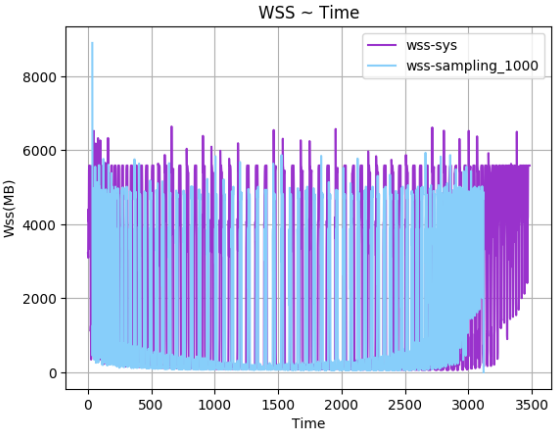
Σχήμα 5.216: 602.gcc_s-ref-2[WSS, samples=1000, interval=1s, user level]



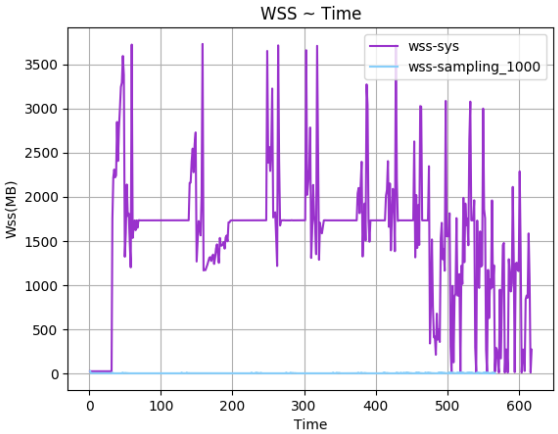
Σχήμα 5.217: 602.gcc_s-ref-3[WSS, samples=1000, interval=1s, user level]



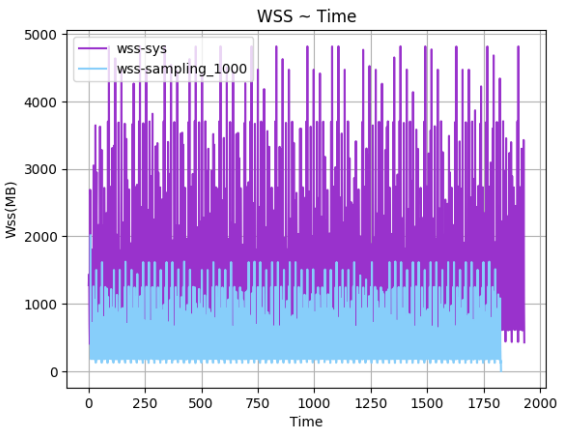
Σχήμα 5.218: 603.bwaves_s-ref-1[WSS, samples=1000, interval=1s, user level]



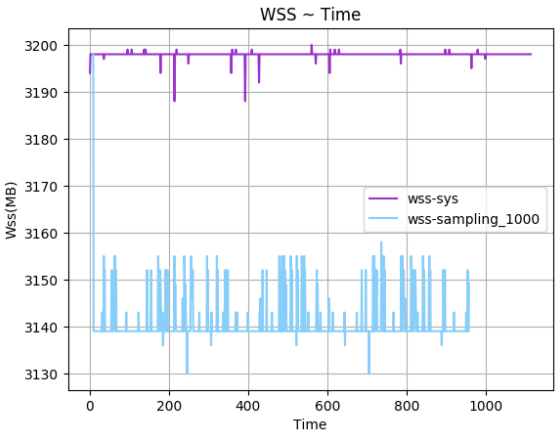
Σχήμα 5.219: 603.bwaves_s-ref-2[WSS, samples=1000, interval=1s, user level]



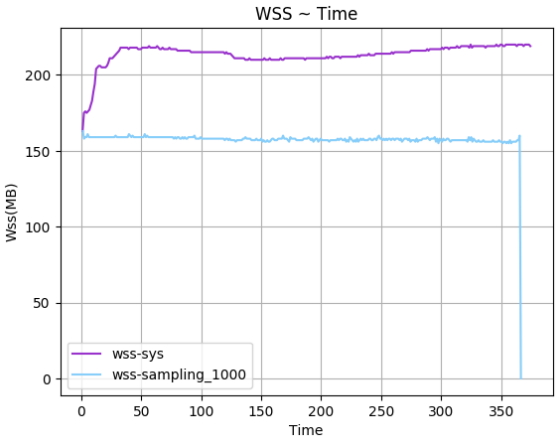
Σχήμα 5.220: 605.mcf_s-ref-1[WSS, samples=1000, interval=1s, user level]



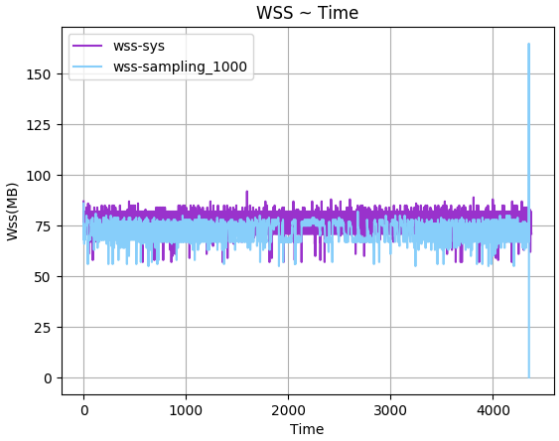
Σχήμα 5.221: 607.cactuBSSN_s-ref-1[WSS, samples=1000, interval=1s, user level]



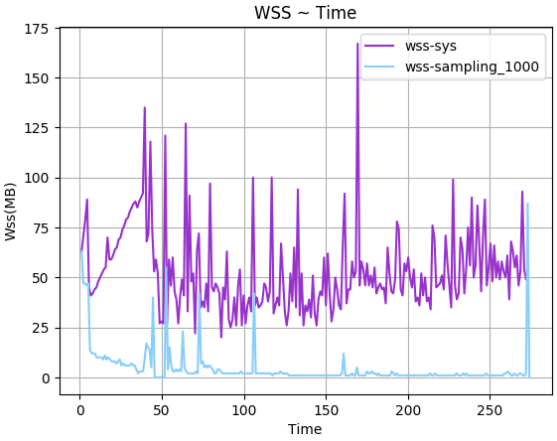
Σχήμα 5.222: 619.lbm_s-ref-1[WSS, samples=1000, interval=1s, user level]



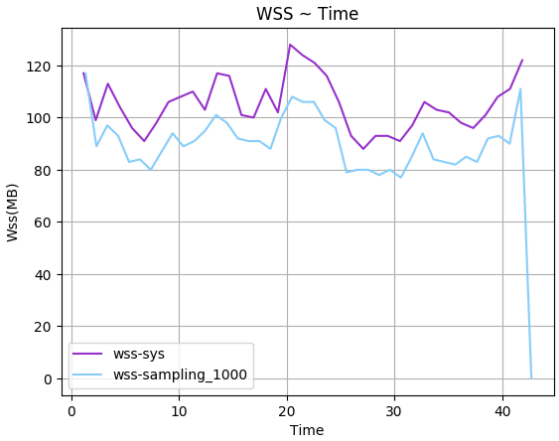
Σχήμα 5.223: 620.omnetpp_s-ref-1[WSS, samples=1000, interval=1s, user level]



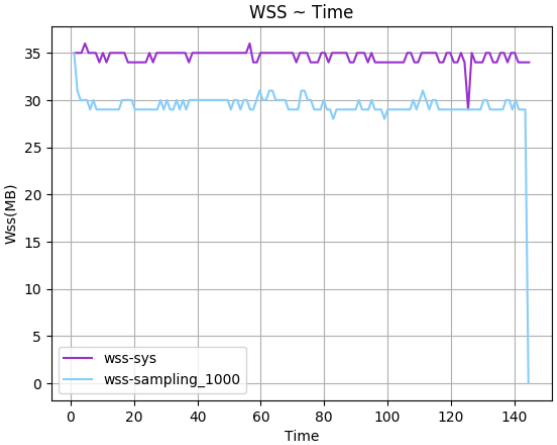
Σχήμα 5.224: 621.wrf_s-ref-1[WSS, samples=1000, interval=1s, user level]



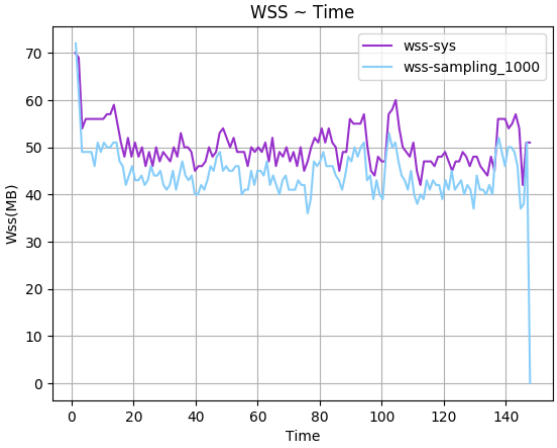
Σχήμα 5.225: 623.xalancbmk_s-ref-1[WSS, samples=1000, interval=1s, user level]



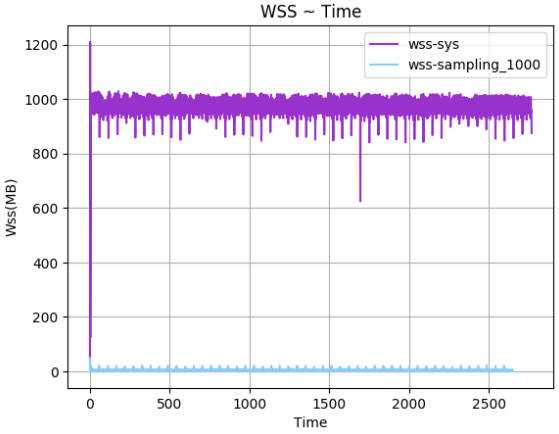
Σχήμα 5.226: 625.x264_s-ref-1[WSS, samples=1000, interval=1s, user level]



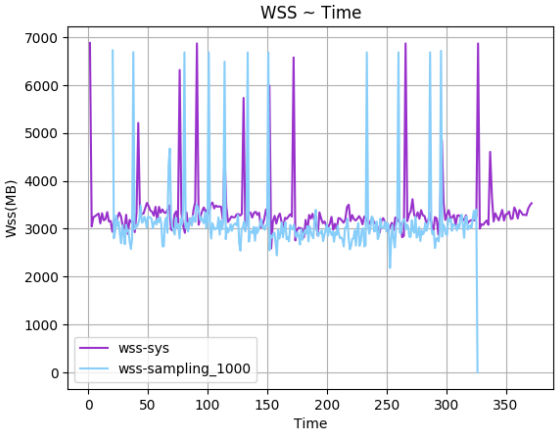
Σχήμα 5.227: 625.x264_s-ref-2[WSS, samples=1000, interval=1s, user level]



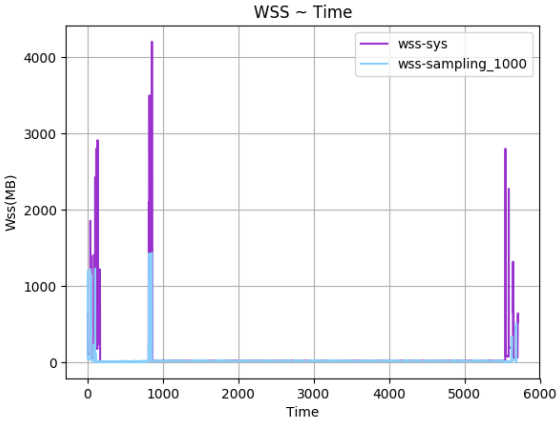
Σχήμα 5.228: 625.x264_s-ref-3[WSS, samples=1000, interval=1s, user level]



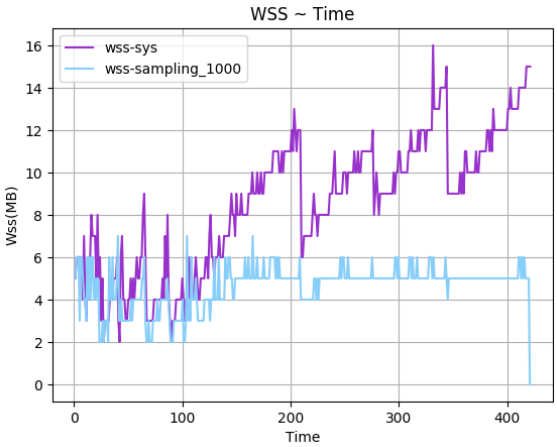
Σχήμα 5.229: 628.pop2_s-ref-1[WSS, samples=1000, interval=1s, user level]



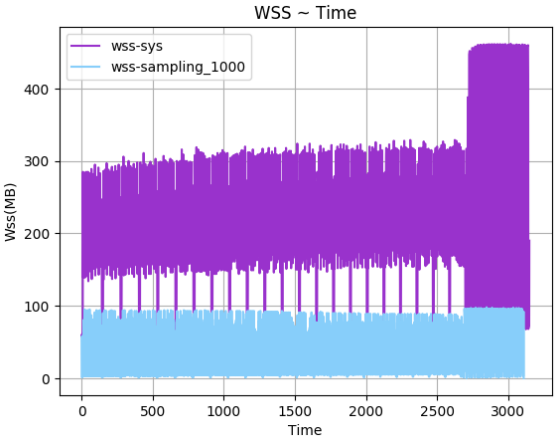
Σχήμα 5.230: 631.deepsjeng_s-ref-1[WSS, samples=1000, interval=1s, user level]



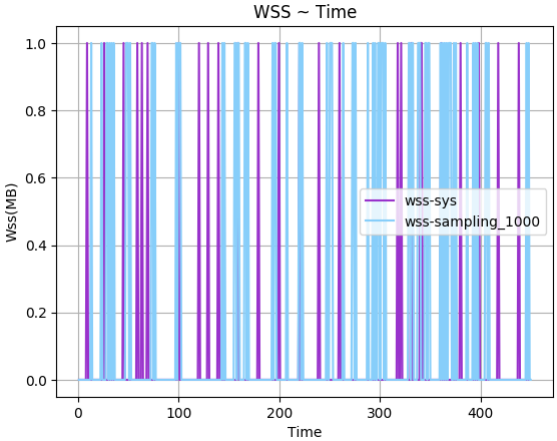
Σχήμα 5.231: 638.imagick_s-ref-1[WSS, samples=1000, interval=1s, user level]



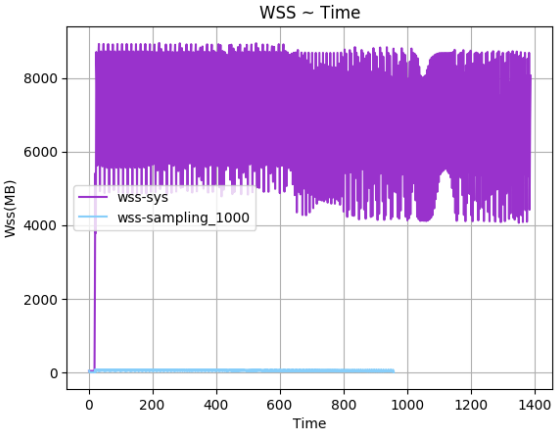
Σχήμα 5.232: 641.leela_s-ref-1[WSS, samples=1000, interval=1s, user level]



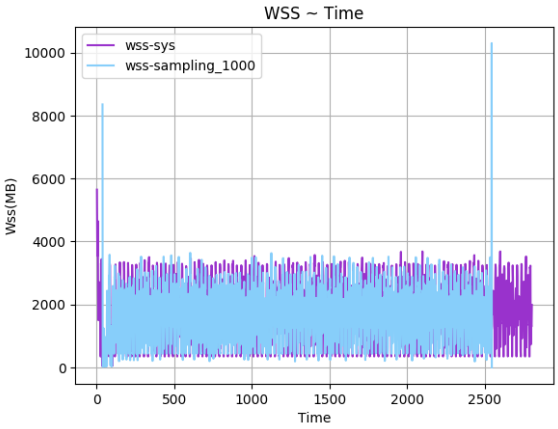
Σχήμα 5.233: 644.nab_s-ref-1[WSS, samples=1000, interval=1s, user level]



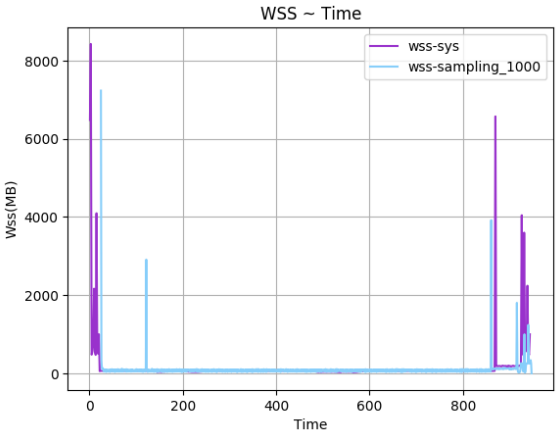
Σχήμα 5.234: 648.exchange2_s-ref-1[WSS, samples=1000, interval=1s, user level]



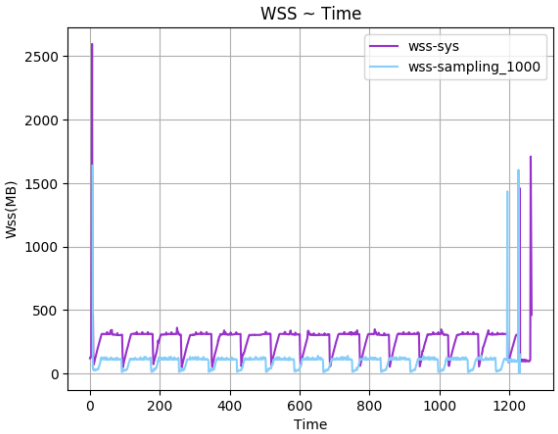
Σχήμα 5.235: 649.fotonik3d_s-ref-1[WSS, samples=1000, interval=1s, user level]



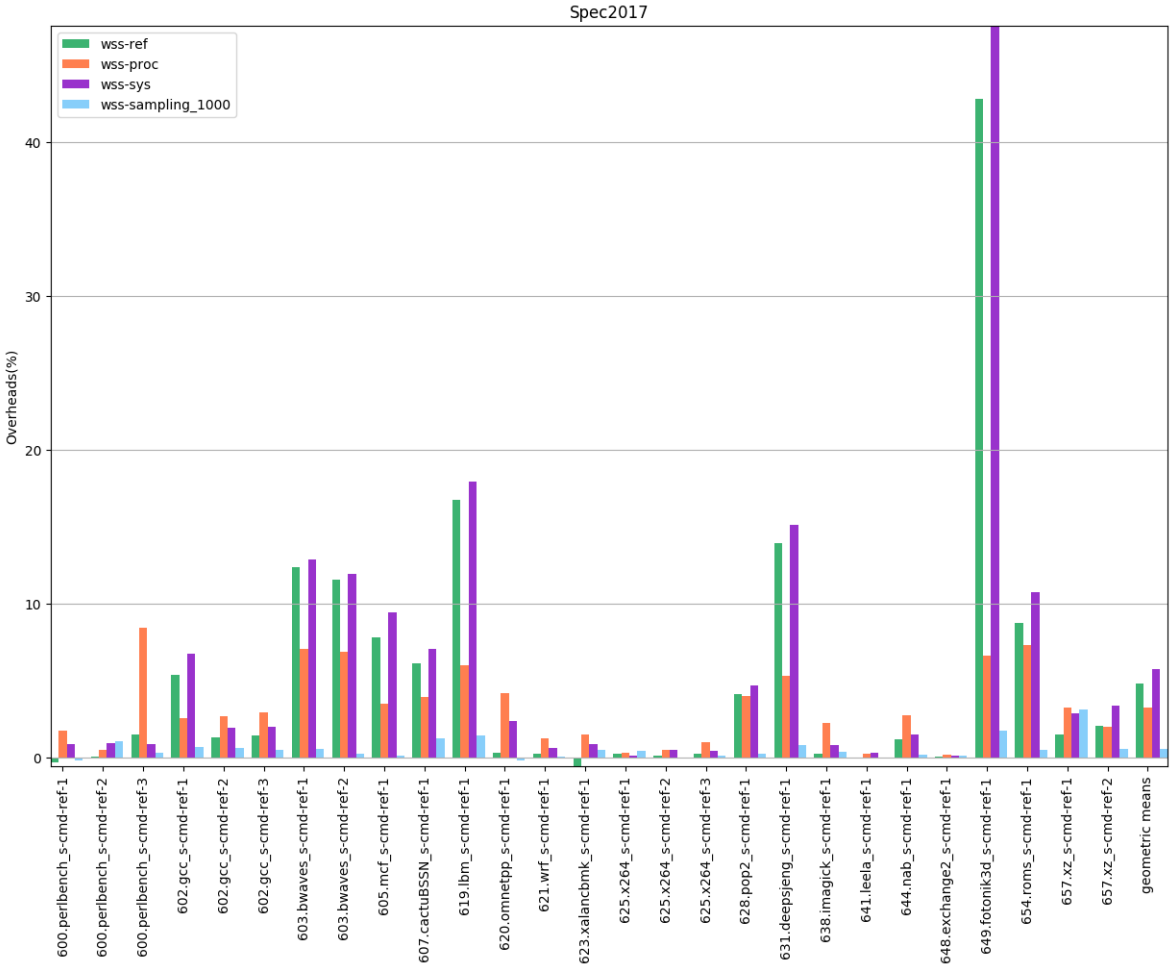
Σχήμα 5.236: 654.roms_s-ref-1[WSS, samples=1000, interval=1s, user level]



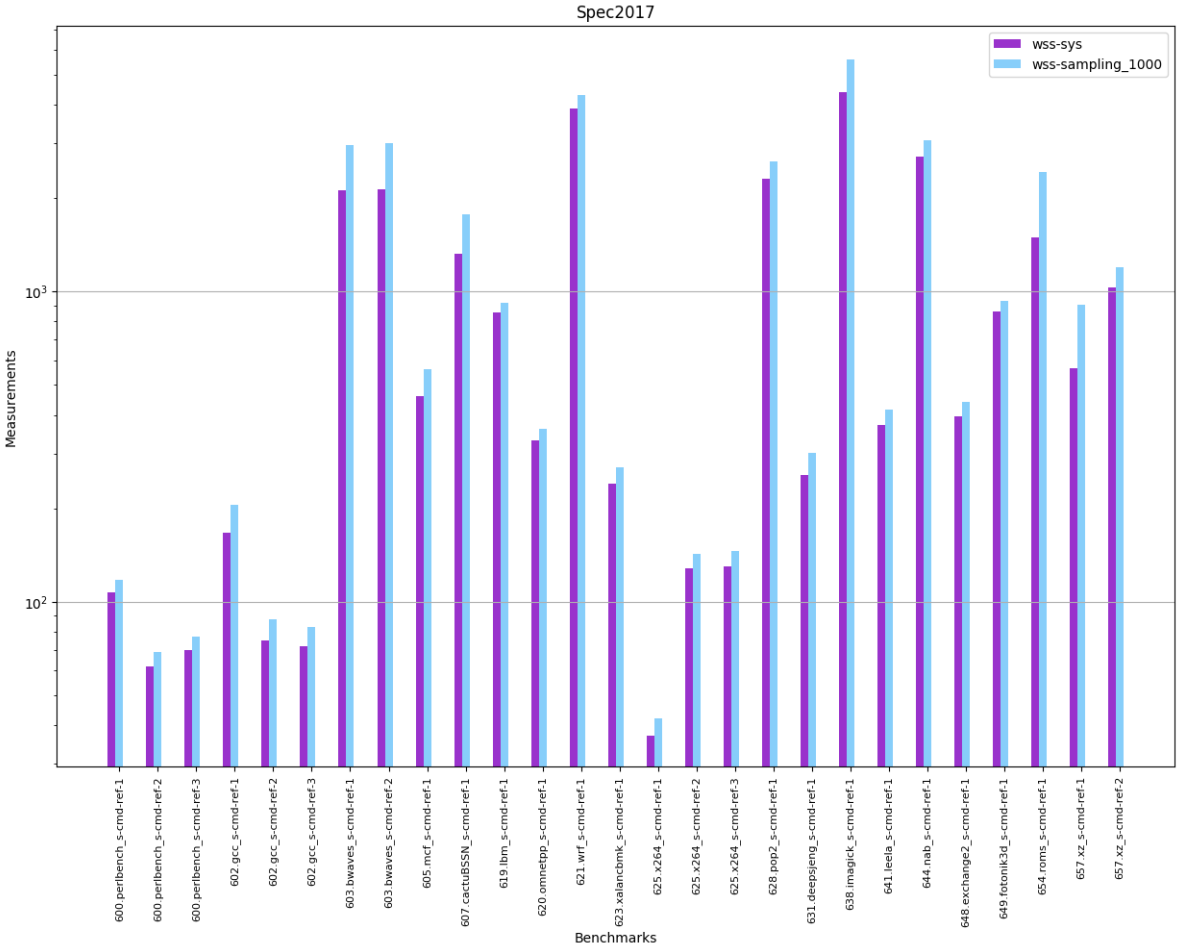
Σχήμα 5.237: 657.xz_s-ref-1[WSS, samples=1000, interval=1s, user level]



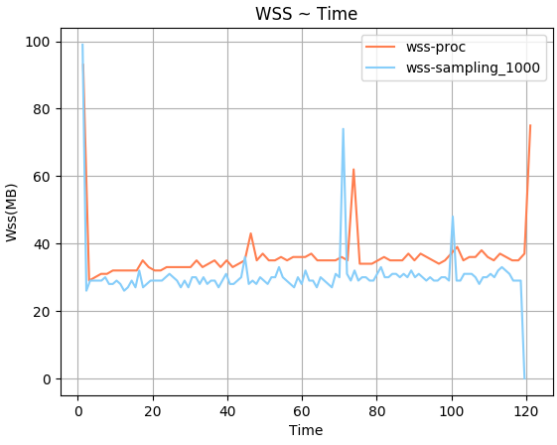
Σχήμα 5.238: 657.xz_s-ref-2[WSS, samples=1000, interval=1s, user level]



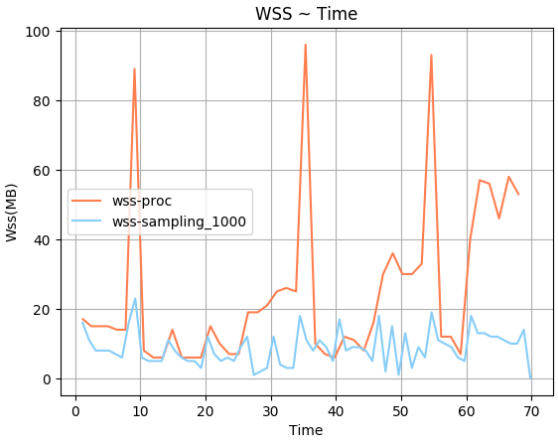
Σχήμα 5.239: 6xx benchmarks[Overheads, samples=1000, interval=1s, user level]



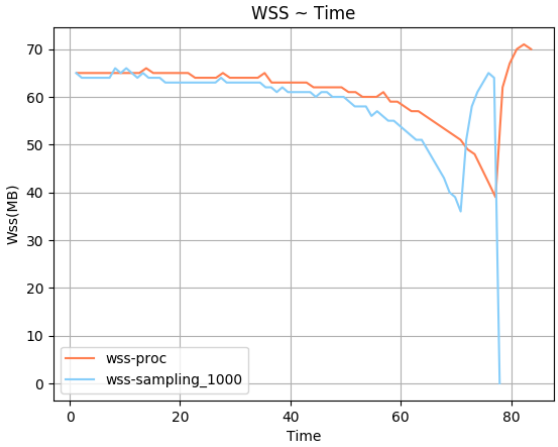
Σχήμα 5.240: 6xx benchmarks[Measurements, samples=1000, interval=1s, user level]



Σχήμα 5.241: 600.perlbench_s-ref-1[WSS, samples=1000, interval=1s, user level]

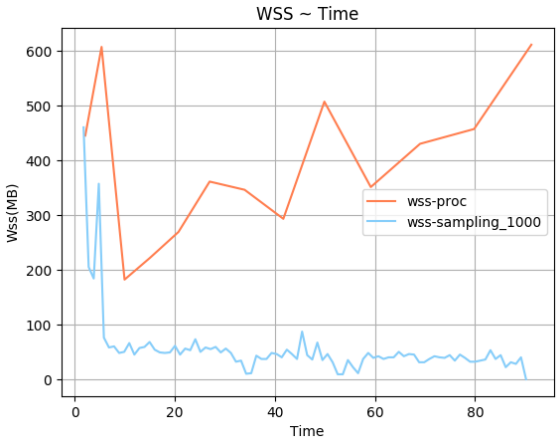


Σχήμα 5.242: 600.perlbench_s-ref-2[WSS, samples=1000, interval=1s, user level]

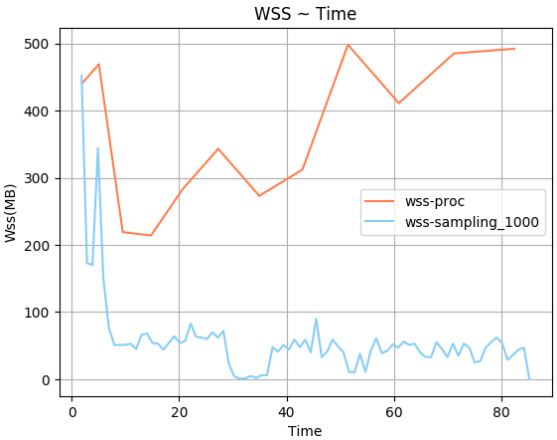


Σχήμα 5.243: 600.perlbench_s-ref-3[WSS, samples=1000, interval=1s, user level]

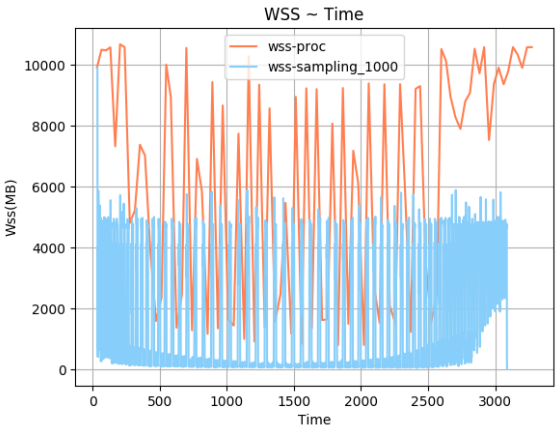
Σχήμα 5.244: 602.gcc_s-ref-1[WSS, samples=1000, interval=1s, user level]



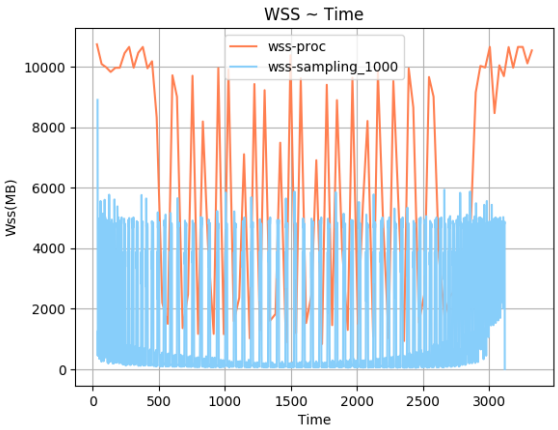
Σχήμα 5.245: 602.gcc_s-ref-2[WSS, samples=1000, interval=1s, user level]



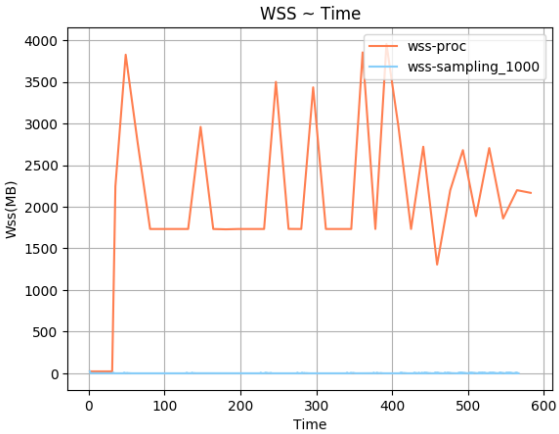
Σχήμα 5.246: 602.gcc_s-ref-3[WSS, samples=1000, interval=1s, user level]



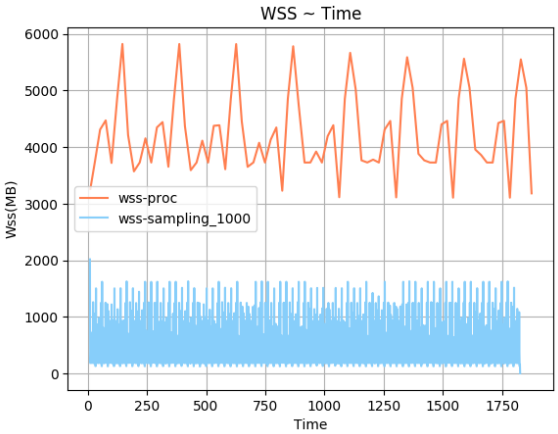
Σχήμα 5.247: 603.bwaves_s-ref-1[WSS, samples=1000, interval=1s, user level]



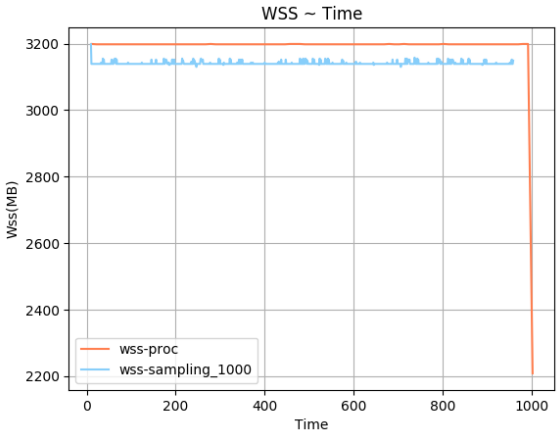
Σχήμα 5.248: 603.bwaves_s-ref-2[WSS, samples=1000, interval=1s, user level]



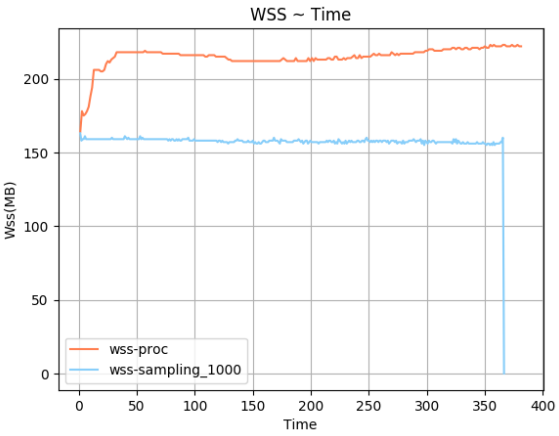
Σχήμα 5.249: 605.mcf_s-ref-1[WSS, samples=1000, interval=1s, user level]



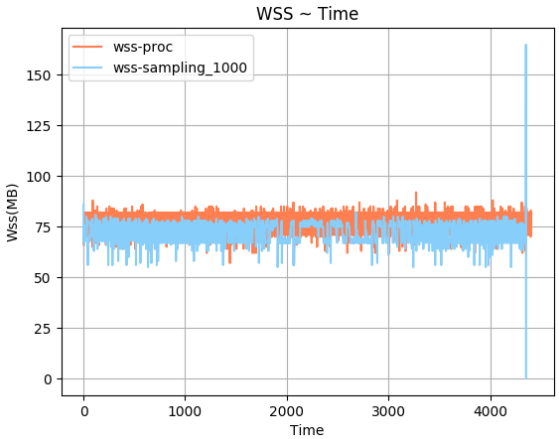
Σχήμα 5.250: 607.cactuBSSN_s-ref-1[WSS, samples=1000, interval=1s, user level]



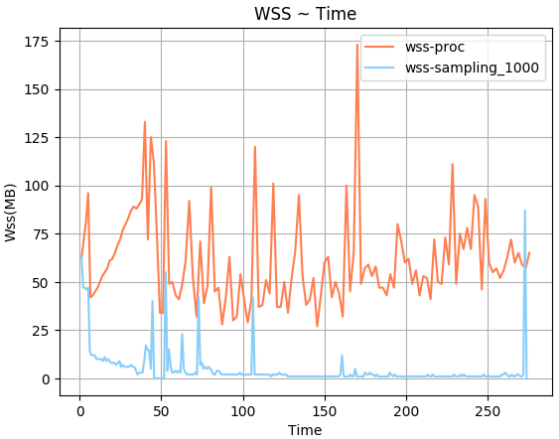
Σχήμα 5.251: 619.lbm_s-ref-1[WSS, samples=1000, interval=1s, user level]



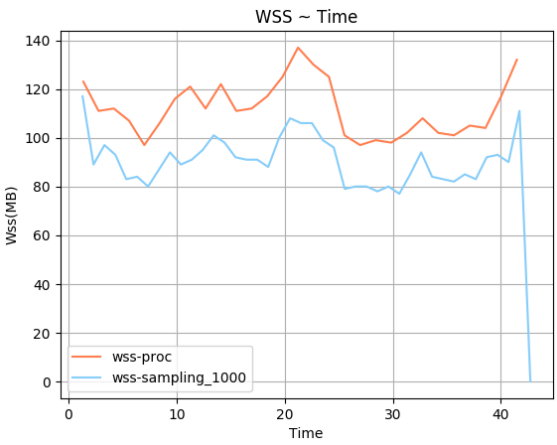
Σχήμα 5.252: 620.omnetpp_s-ref-1[WSS, samples=1000, interval=1s, user level]



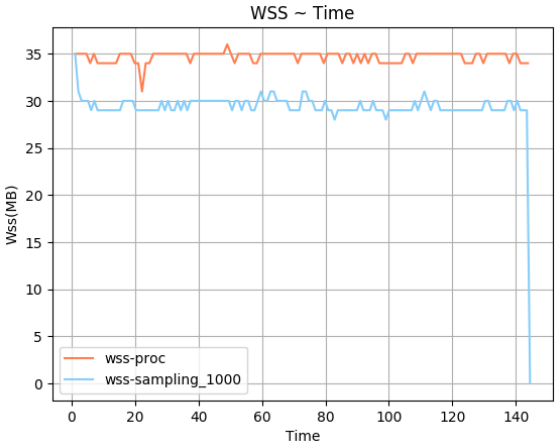
Σχήμα 5.253: 621.wrf_s-ref-1[WSS, samples=1000, interval=1s, user level]



Σχήμα 5.254: 623.xalancbmk_s-ref-1[WSS, samples=1000, interval=1s, user level]

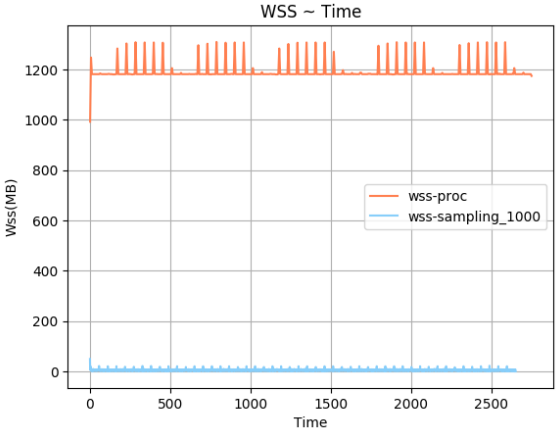


Σχήμα 5.255: 625.x264_s-ref-1[WSS, samples=1000, interval=1s, user level]

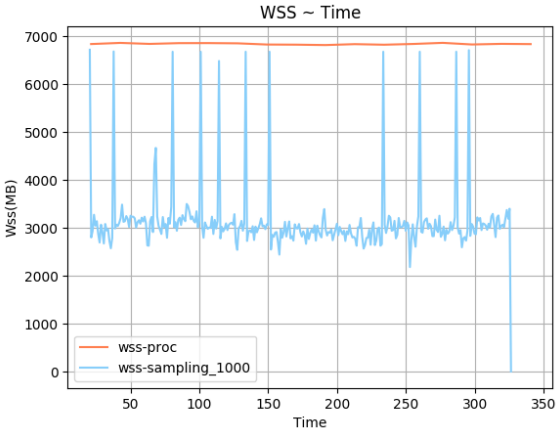


Σχήμα 5.256: 625.x264_s-ref-2[WSS, samples=1000, interval=1s, user level]

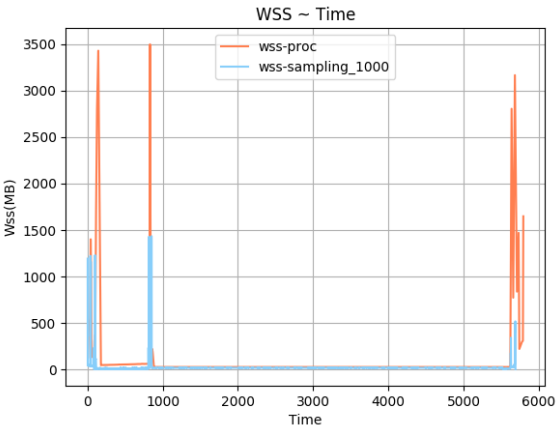
Σχήμα 5.257: 625.x264_s-ref-3[WSS, samples=1000, interval=1s, user level]



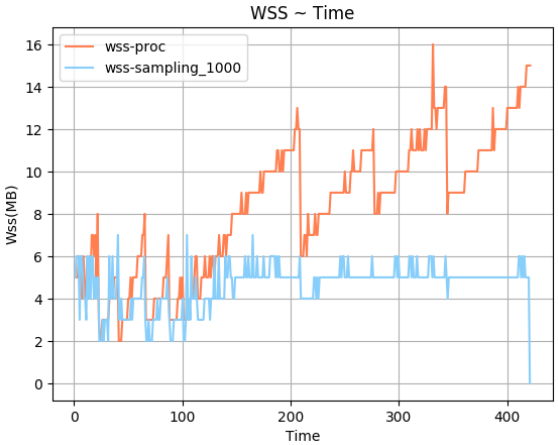
Σχήμα 5.258: 628.pop2_s-ref-1[WSS, samples=1000, interval=1s, user level]



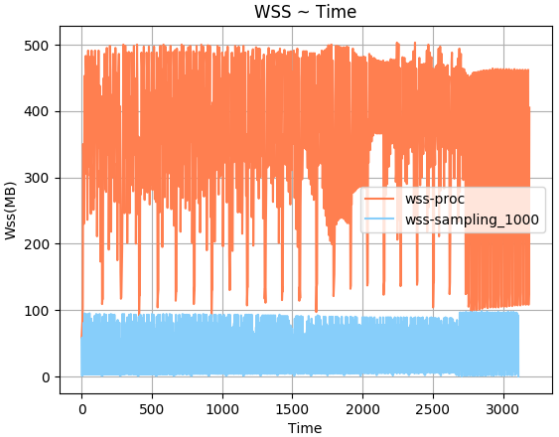
Σχήμα 5.259: 631.deepsjeng_s-ref-1[WSS, samples=1000, interval=1s, user level]



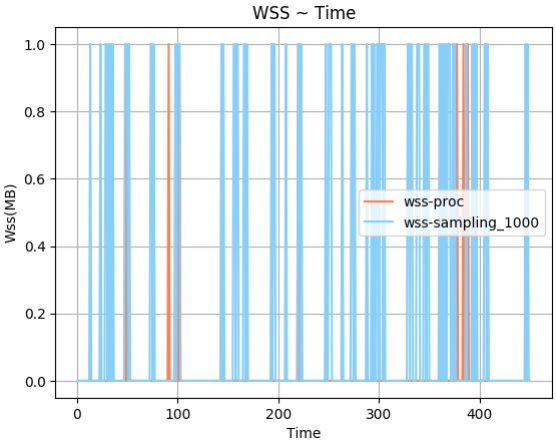
Σχήμα 5.260: 638.imagick_s-ref-1[WSS, samples=1000, interval=1s, user level]



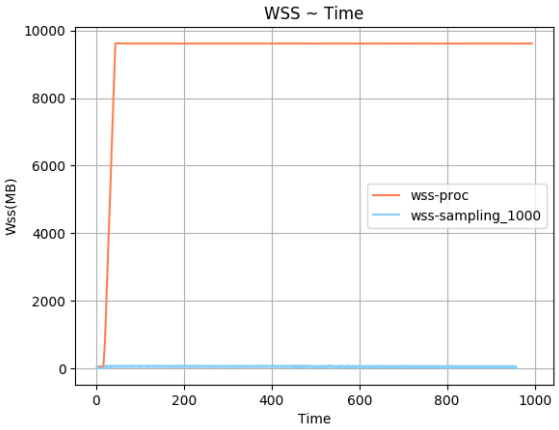
Σχήμα 5.261: 641.leela_s-ref-1[WSS, samples=1000, interval=1s, user level]



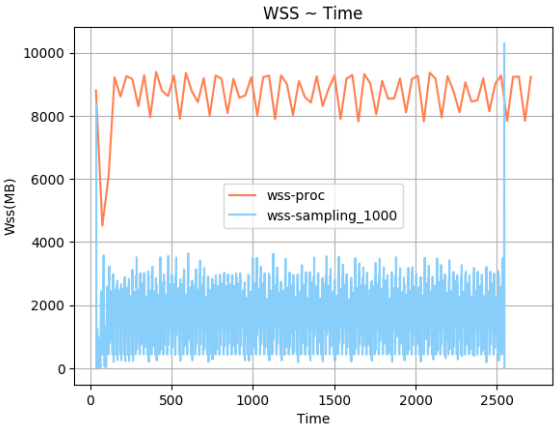
Σχήμα 5.262: 644.nab_s-ref-1[WSS, samples=1000, interval=1s, user level]



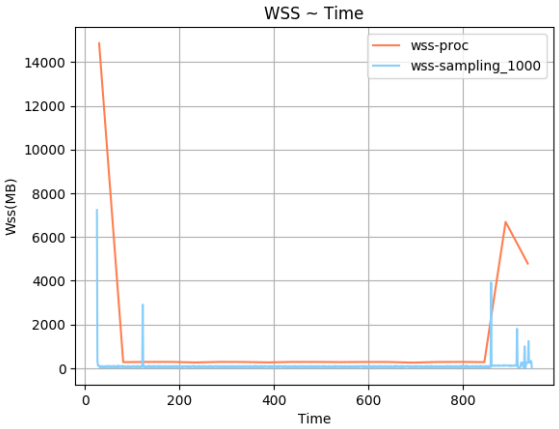
Σχήμα 5.263: 648.exchange2_s-ref-1[WSS, samples=1000, interval=1s, user level]



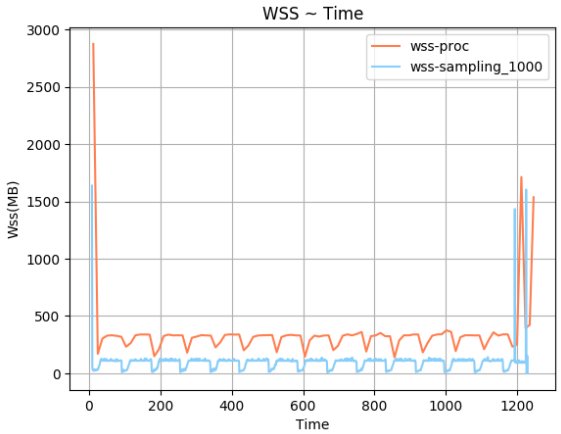
Σχήμα 5.264: 649.fotonik3d_s-ref-1[WSS, samples=1000, interval=1s, user level]



Σχήμα 5.265: 654.roms_s-ref-1[WSS, samples=1000, interval=1s, user level]

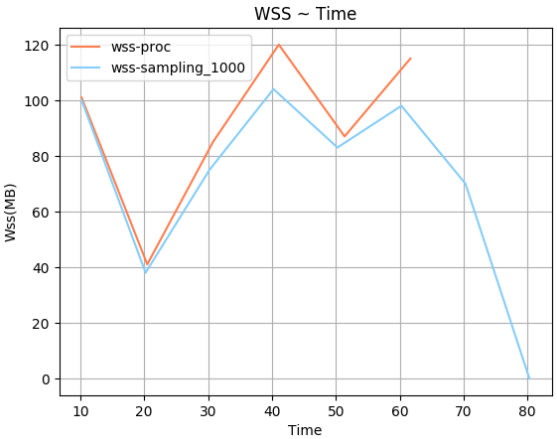


Σχήμα 5.266: 657.xz_s-ref-1[WSS, samples=1000, interval=1s, user level]

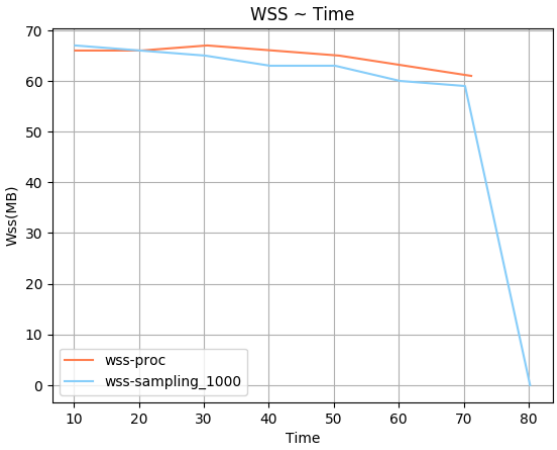


Σχήμα 5.267: 657.xz_s-ref-2[WSS, samples=1000, interval=1s, user level]

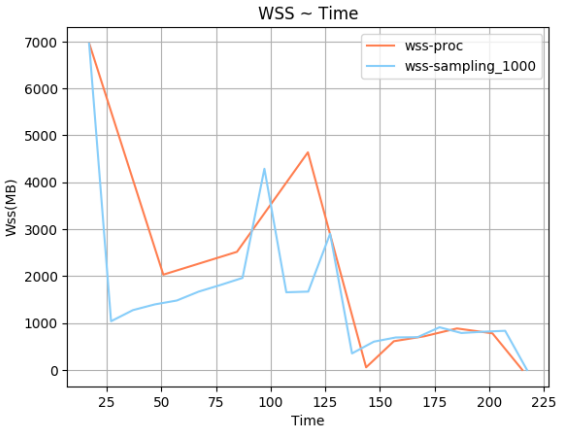
Σχήμα 5.268: 600.perlbench_s-ref-1[WSS, samples=1000, interval=10s, user level]



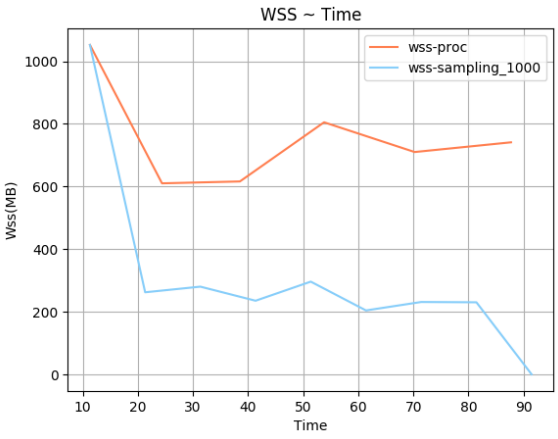
Σχήμα 5.269: 600.perlbench_s-ref-2[WSS, samples=1000, interval=10s, user level]



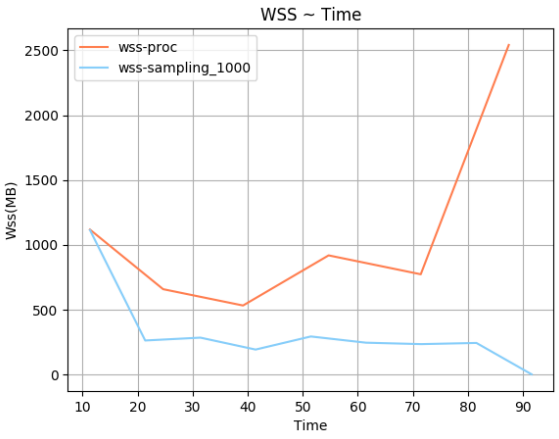
Σχήμα 5.270: 600.perlbench_s-ref-3[WSS, samples=1000, interval=10s, user level]



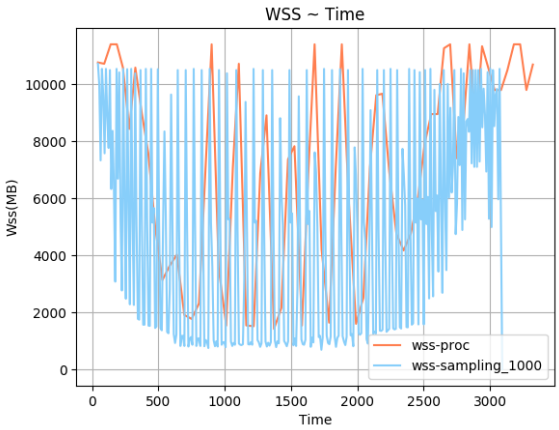
Σχήμα 5.271: 602.gcc_s-ref-1[WSS, samples=1000, interval=10s, user level]



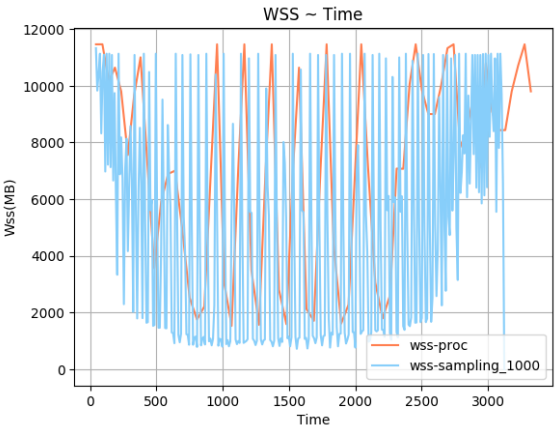
Σχήμα 5.272: 602.gcc_s-ref-2[WSS, samples=1000, interval=10s, user level]



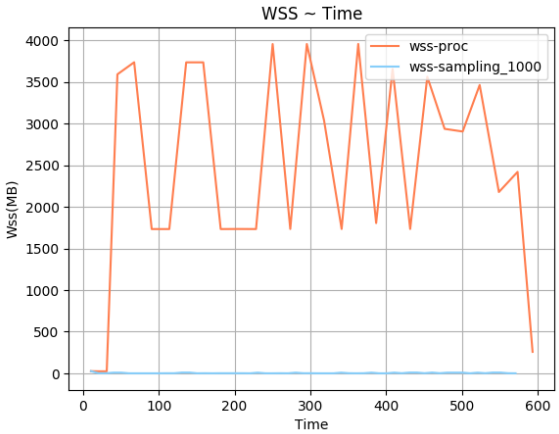
Σχήμα 5.273: 602.gcc_s-ref-3[WSS, samples=1000, interval=10s, user level]



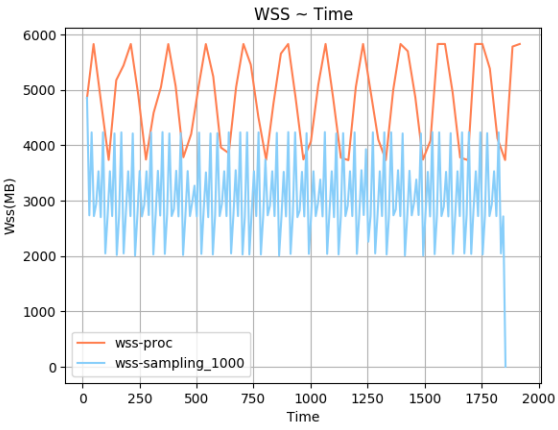
Σχήμα 5.274: 603.bwaves_s-ref-1[WSS, samples=1000, interval=10s, user level]



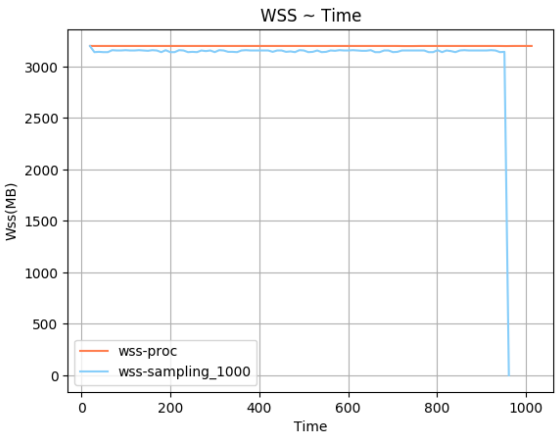
Σχήμα 5.275: 603.bwaves_s-ref-2[WSS, samples=1000, interval=10s, user level]



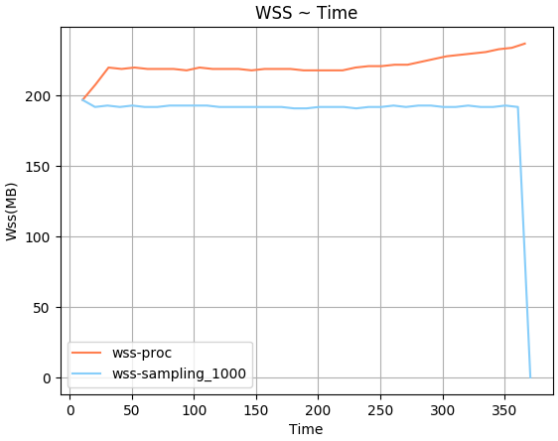
Σχήμα 5.276: 605.mcf_s-ref-1[WSS, samples=1000, interval=10s, user level]



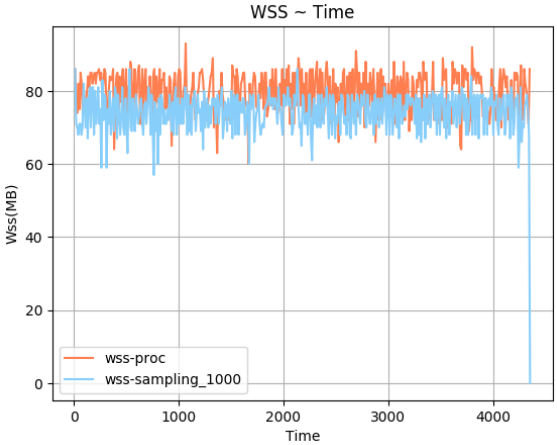
Σχήμα 5.277: 607.cactuBSSN_s-ref-1[WSS, samples=1000, interval=10s, user level]



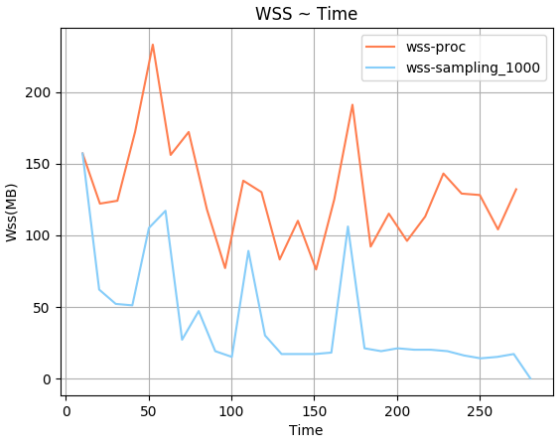
Σχήμα 5.278: 619.lbm_s-ref-1[WSS, samples=1000, interval=10s, user level]



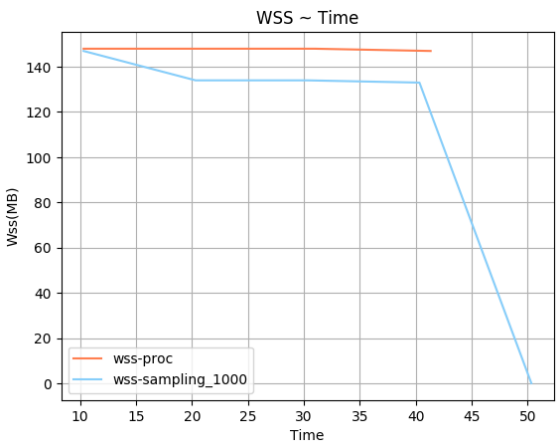
Σχήμα 5.279: 620.omnetpp_s-ref-1[WSS, samples=1000, interval=10s, user level]



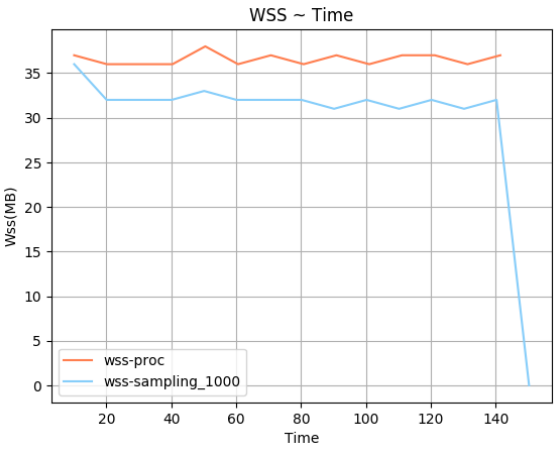
Σχήμα 5.280: 621.wrf_s-ref-1[WSS, samples=1000, interval=10s, user level]



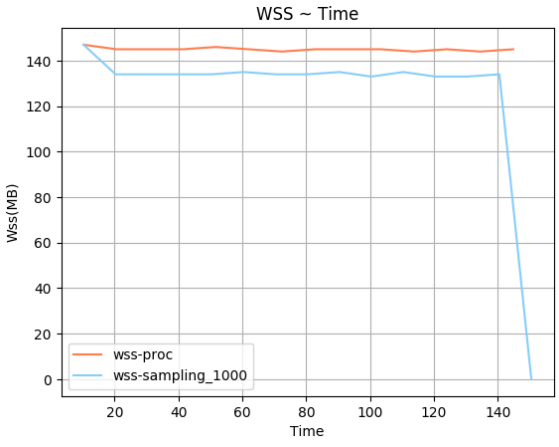
Σχήμα 5.281: 623.xalancbmk_s-ref-1[WSS, samples=1000, interval=10s, user level]



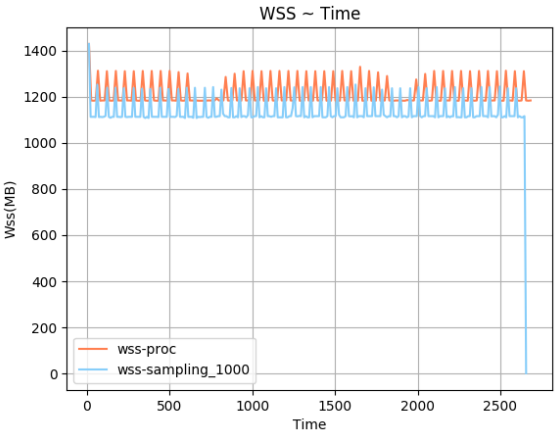
Σχήμα 5.282: 625.x264_s-ref-1[WSS, samples=1000, interval=10s, user level]



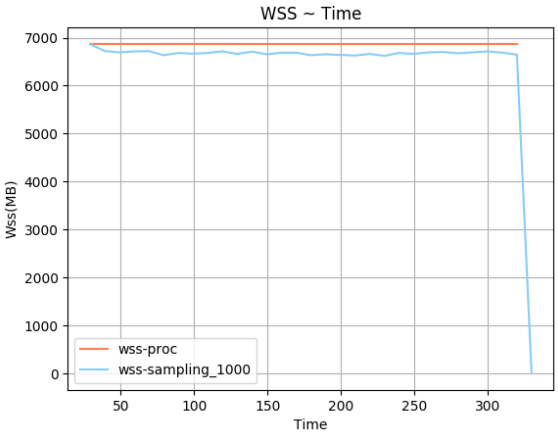
Σχήμα 5.283: 625.x264_s-ref-2[WSS, samples=1000, interval=10s, user level]



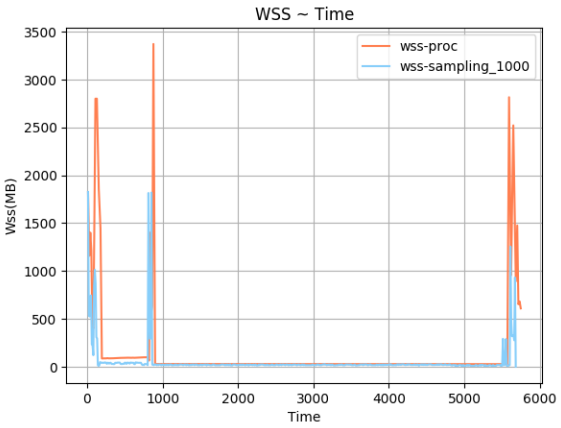
Σχήμα 5.284: 625.x264_s-ref-3[WSS, samples=1000, interval=10s, user level]



Σχήμα 5.285: 628.pop2_s-ref-1[WSS, samples=1000, interval=10s, user level]

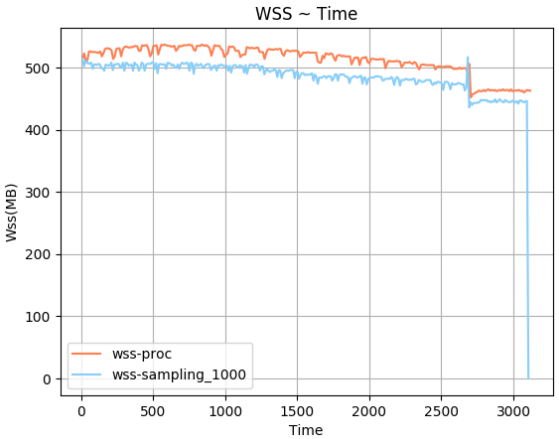


Σχήμα 5.286: 631.deepsjeng_s-ref-1[WSS, samples=1000, interval=10s, user level]

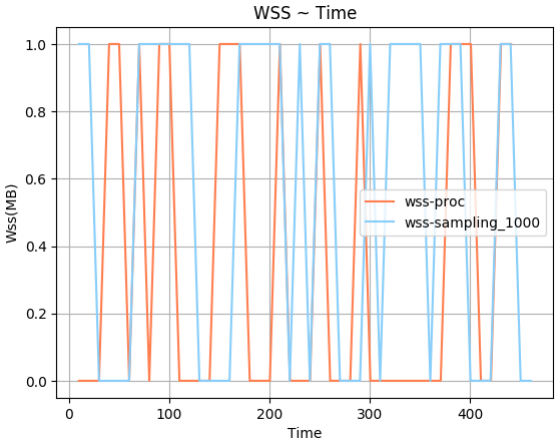


Σχήμα 5.287: 638.imagick_s-ref-1[WSS, samples=1000, interval=10s, user level]

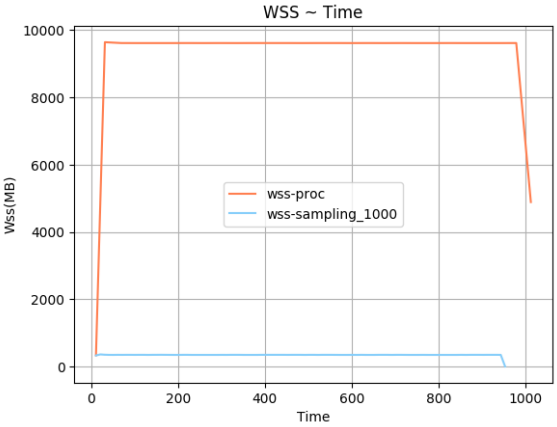
Σχήμα 5.288: 641.leela_s-ref-1[WSS, samples=1000, interval=10s, user level]



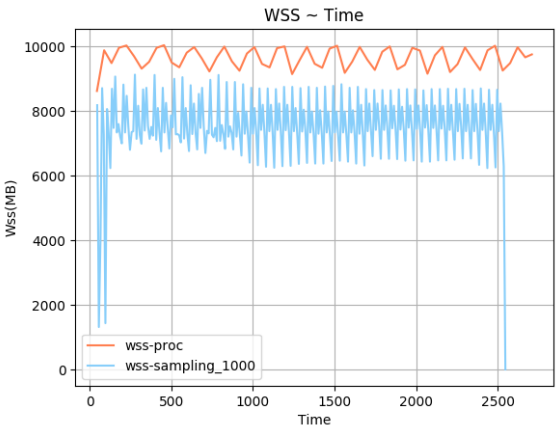
Σχήμα 5.289: 644.nab_s-ref-1[WSS, samples=1000, interval=10s, user level]



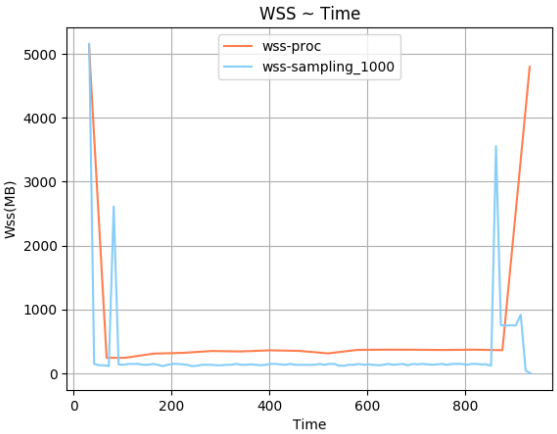
Σχήμα 5.290: 648.exchange2_s-ref-1[WSS, samples=1000, interval=10s, user level]



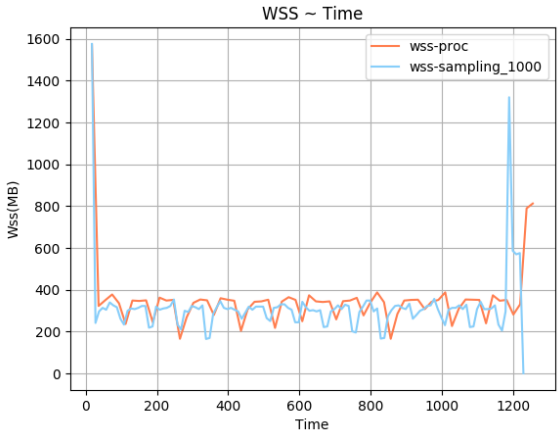
Σχήμα 5.291: 649.fotonik3d_s-ref-1[WSS, samples=1000, interval=10s, user level]



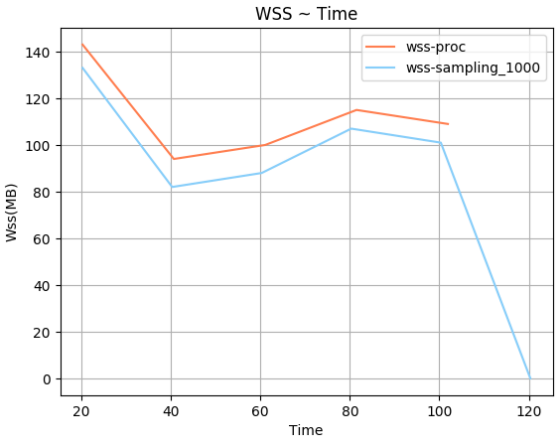
Σχήμα 5.292: 654.roms_s-ref-1[WSS, samples=1000, interval=10s, user level]



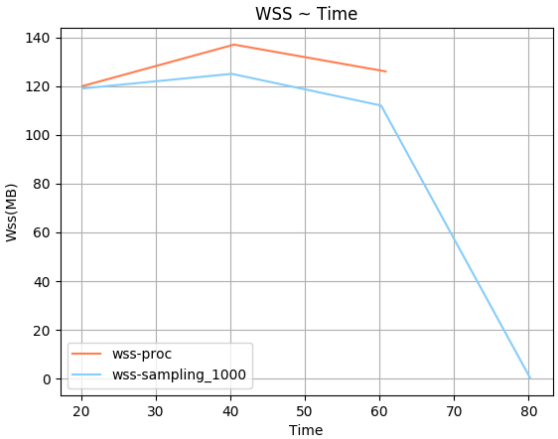
Σχήμα 5.293: 657.xz_s-ref-1[WSS, samples=1000, interval=10s, user level]



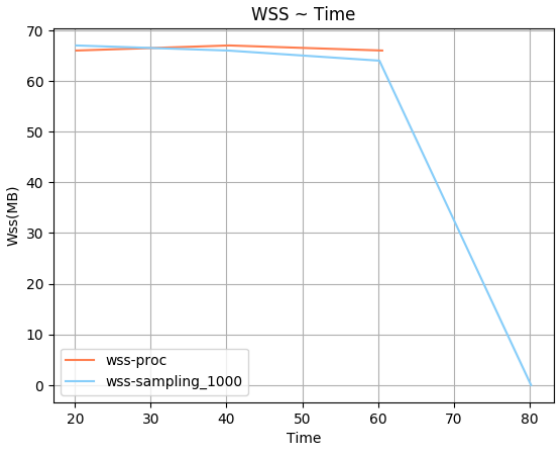
Σχήμα 5.294: 657.xz_s-ref-2[WSS, samples=1000, interval=10s, user level]



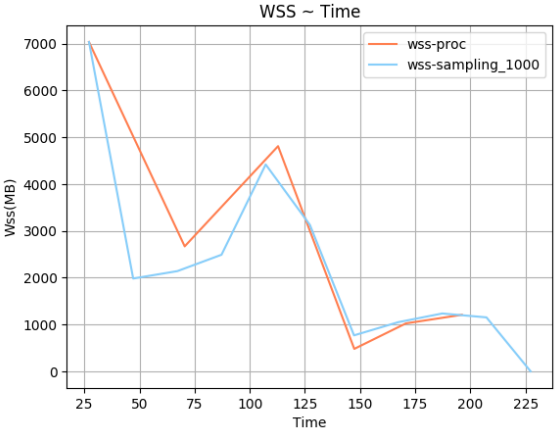
Σχήμα 5.295: 600.perlbench_s-ref-1[WSS, samples=1000, interval=20s, user level]



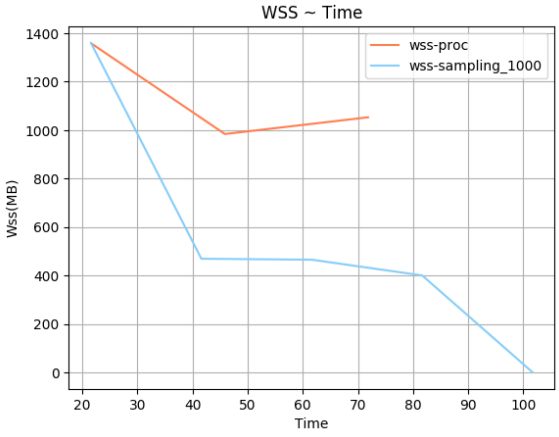
Σχήμα 5.296: 600.perlbench_s-ref-2[WSS, samples=1000, interval=20s, user level]



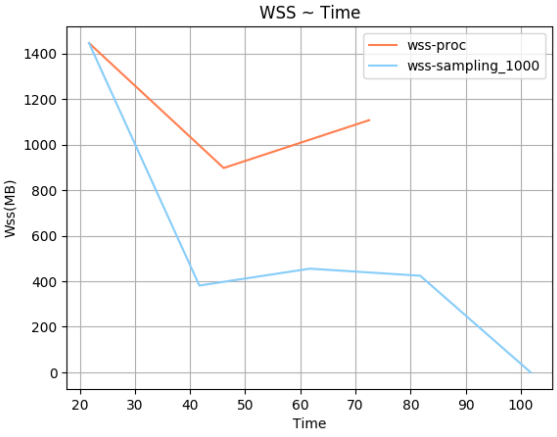
Σχήμα 5.297: 600.perlbench_s-ref-3[WSS, samples=1000, interval=20s, user level]



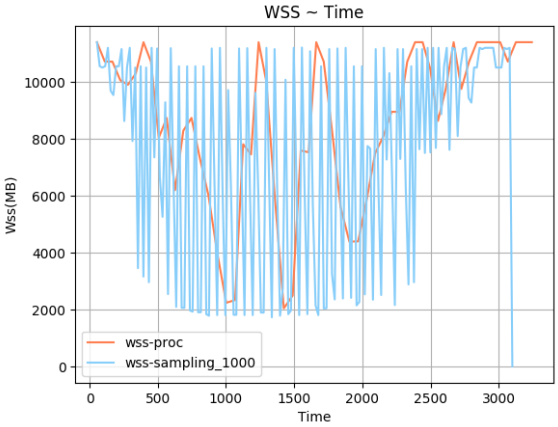
Σχήμα 5.298: 602.gcc_s-ref-1[WSS, samples=1000, interval=20s, user level]



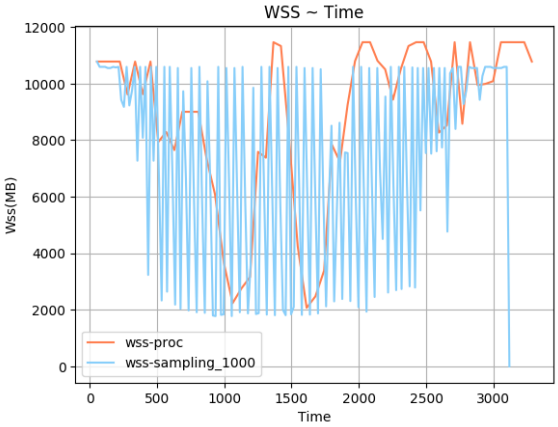
Σχήμα 5.299: 602.gcc_s-ref-2[WSS, samples=1000, interval=20s, user level]



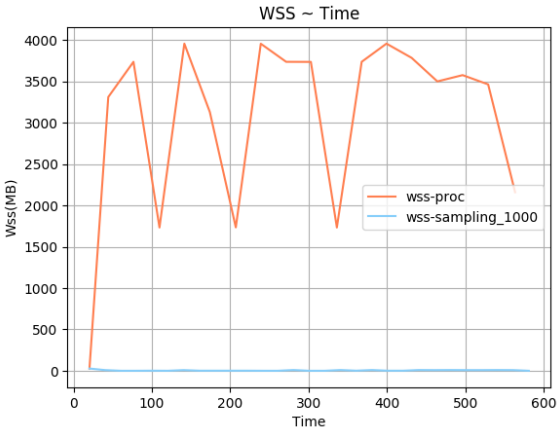
Σχήμα 5.300: 602.gcc_s-ref-3[WSS, samples=1000, interval=20s, user level]



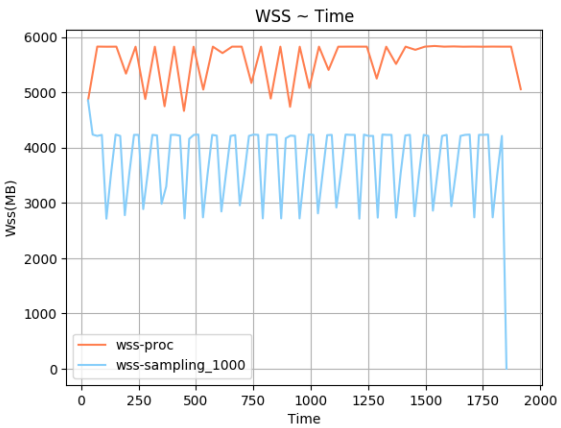
Σχήμα 5.301: 603.bwaves_s-ref-1[WSS, samples=1000, interval=20s, user level]



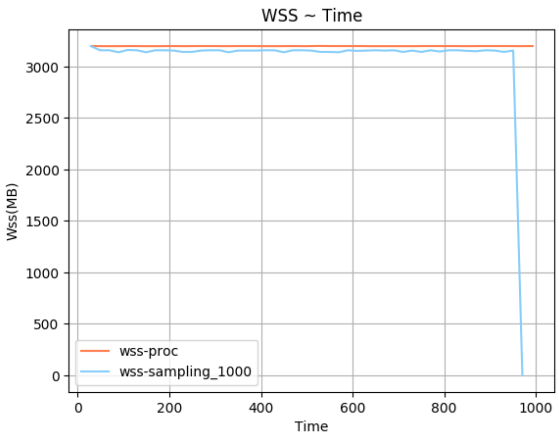
Σχήμα 5.302: 603.bwaves_s-ref-2[WSS, samples=1000, interval=20s, user level]



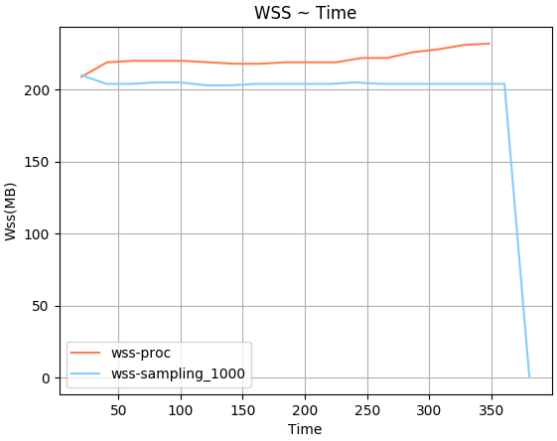
Σχήμα 5.303: 605.mcf_s-ref-1[WSS, samples=1000, interval=20s, user level]



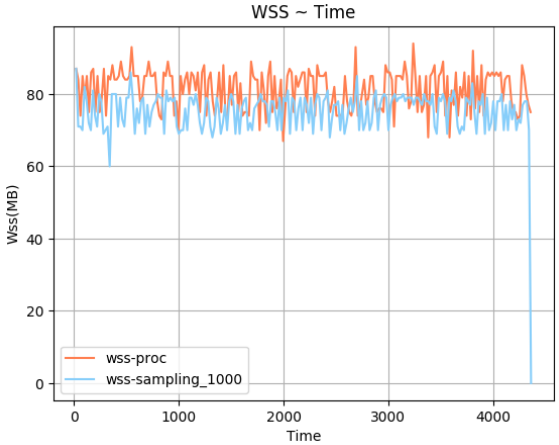
Σχήμα 5.304: 607.cactuBSSN_s-ref-1[WSS, samples=1000, interval=20s, user level]



Σχήμα 5.305: 619.lbm_s-ref-1[WSS, samples=1000, interval=20s, user level]

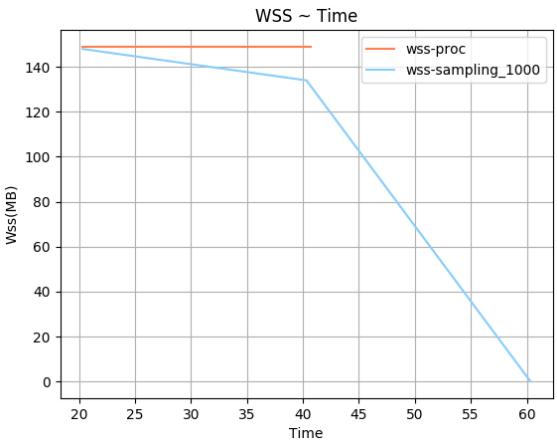


Σχήμα 5.306: 620.omnetpp_s-ref-1[WSS, samples=1000, interval=20s, user level]

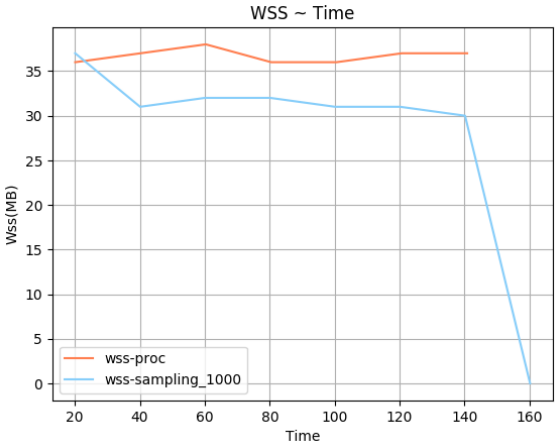


Σχήμα 5.307: 621.wrf_s-ref-1[WSS, samples=1000, interval=20s, user level]

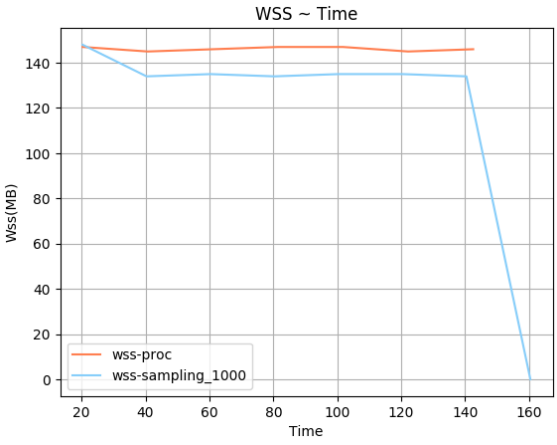
Σχήμα 5.308: 623.xalancbmk_s-ref-1[WSS, samples=1000, interval=20s, user level]



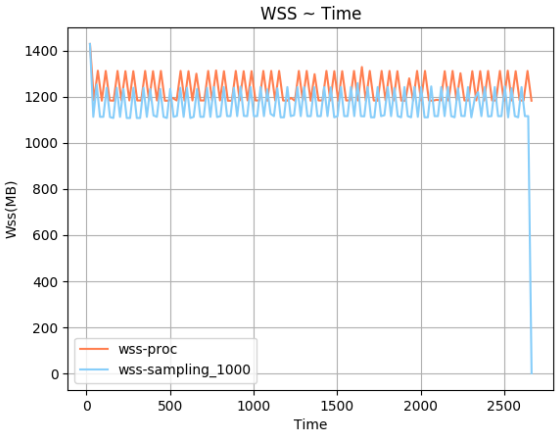
Σχήμα 5.309: 625.x264_s-ref-1[WSS, samples=1000, interval=20s, user level]



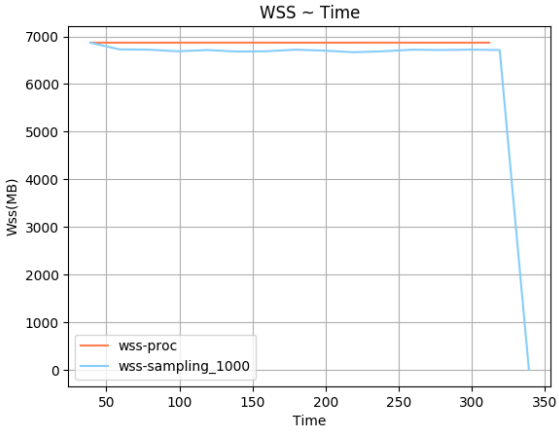
Σχήμα 5.310: 625.x264_s-ref-2[WSS, samples=1000, interval=20s, user level]



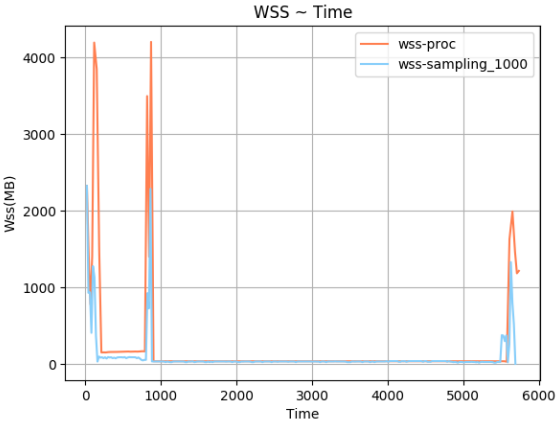
Σχήμα 5.311: 625.x264_s-ref-3[WSS, samples=1000, interval=20s, user level]



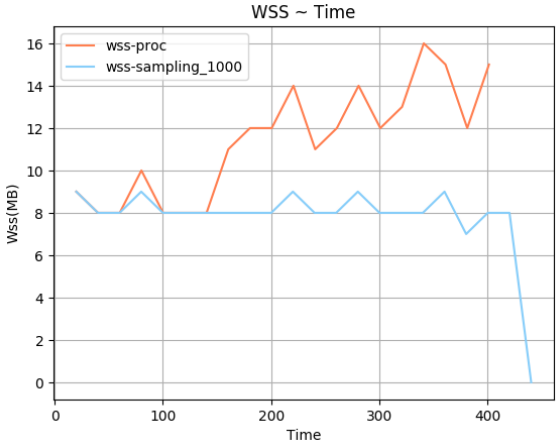
Σχήμα 5.312: 628.pop2_s-ref-1[WSS, samples=1000, interval=20s, user level]



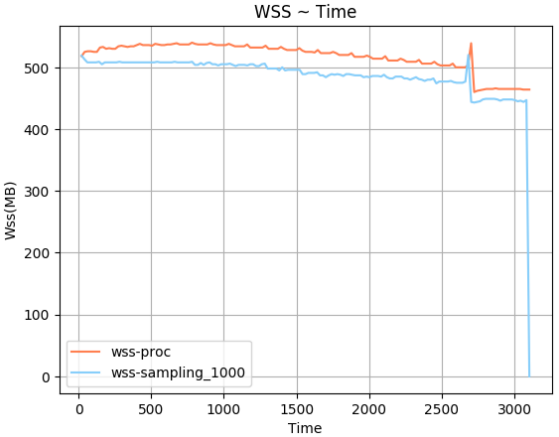
Σχήμα 5.313: 631.deepsjeng_s-ref-1[WSS, samples=1000, interval=20s, user level]



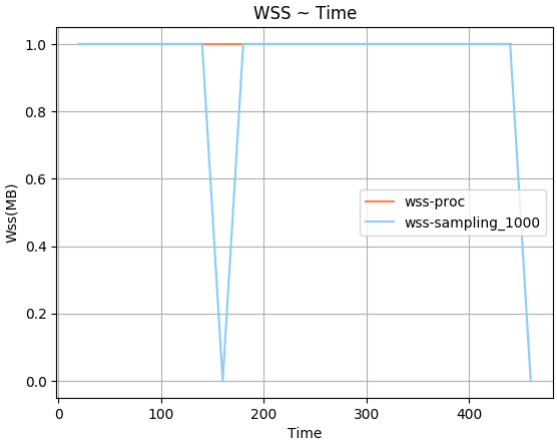
Σχήμα 5.314: 638.imagick_s-ref-1[WSS, samples=1000, interval=20s, user level]



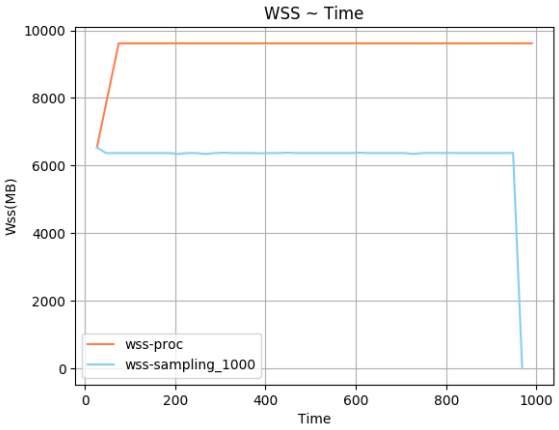
Σχήμα 5.315: 641.leela_s-ref-1[WSS, samples=1000, interval=20s, user level]



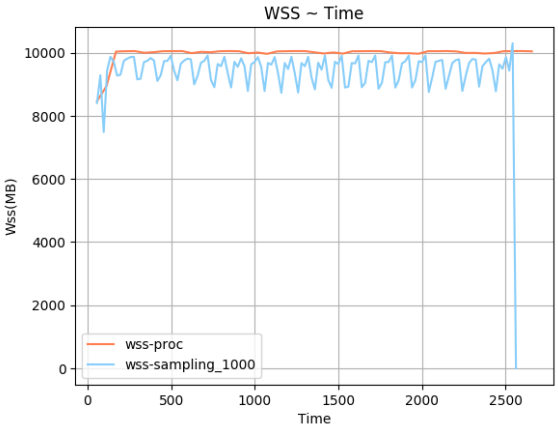
Σχήμα 5.316: 644.nab_s-ref-1[WSS, samples=1000, interval=20s, user level]



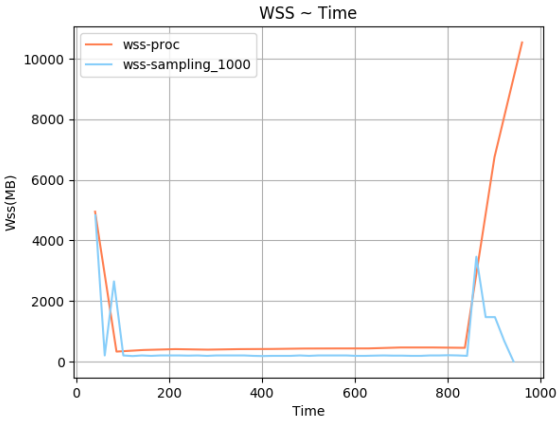
Σχήμα 5.317: 648.exchange2_s-ref-1[WSS, samples=1000, interval=20s, user level]



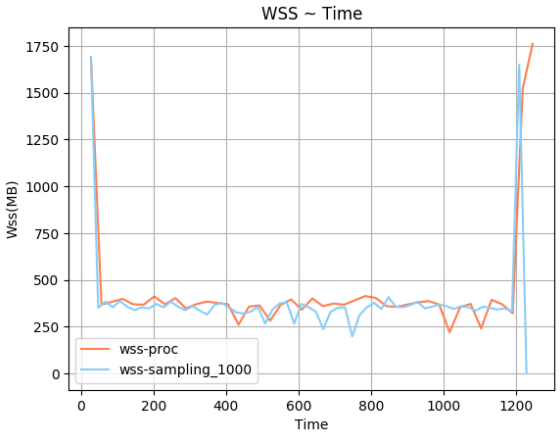
Σχήμα 5.318: 649.fotonik3d_s-ref-1[WSS, samples=1000, interval=20s, user level]



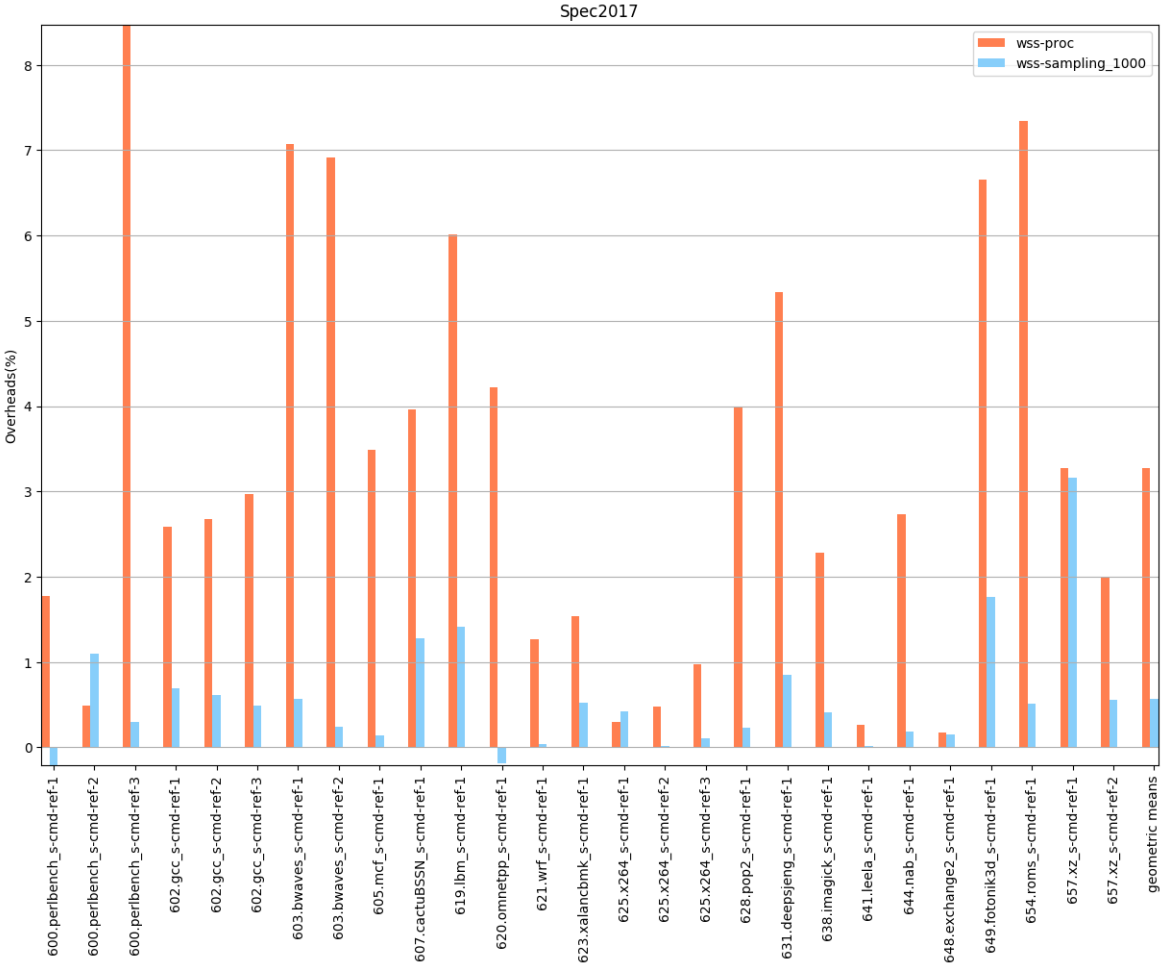
Σχήμα 5.319: 654.roms_s-ref-1[WSS, samples=1000, interval=20s, user level]



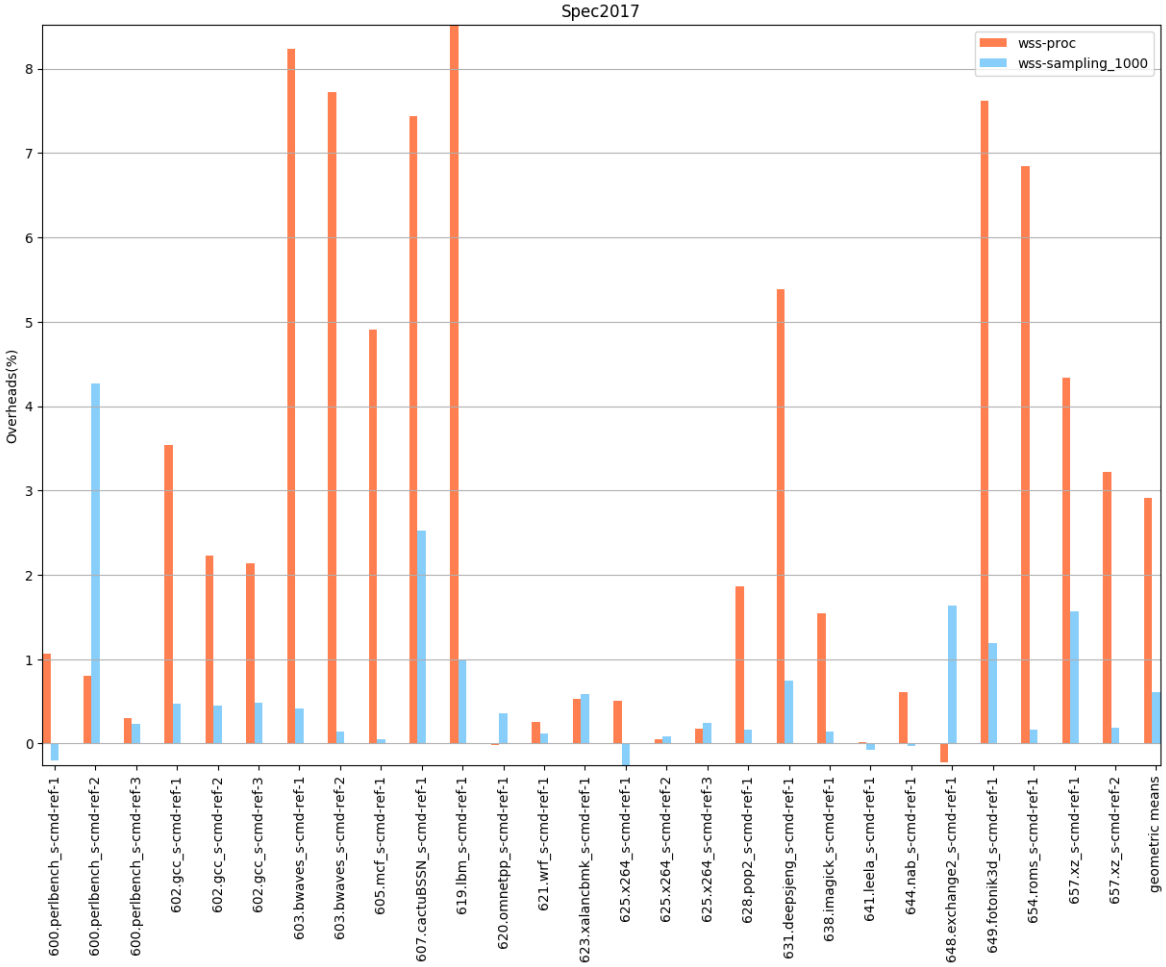
Σχήμα 5.320: 657.xz_s-ref-1[WSS, samples=1000, interval=20s, user level]



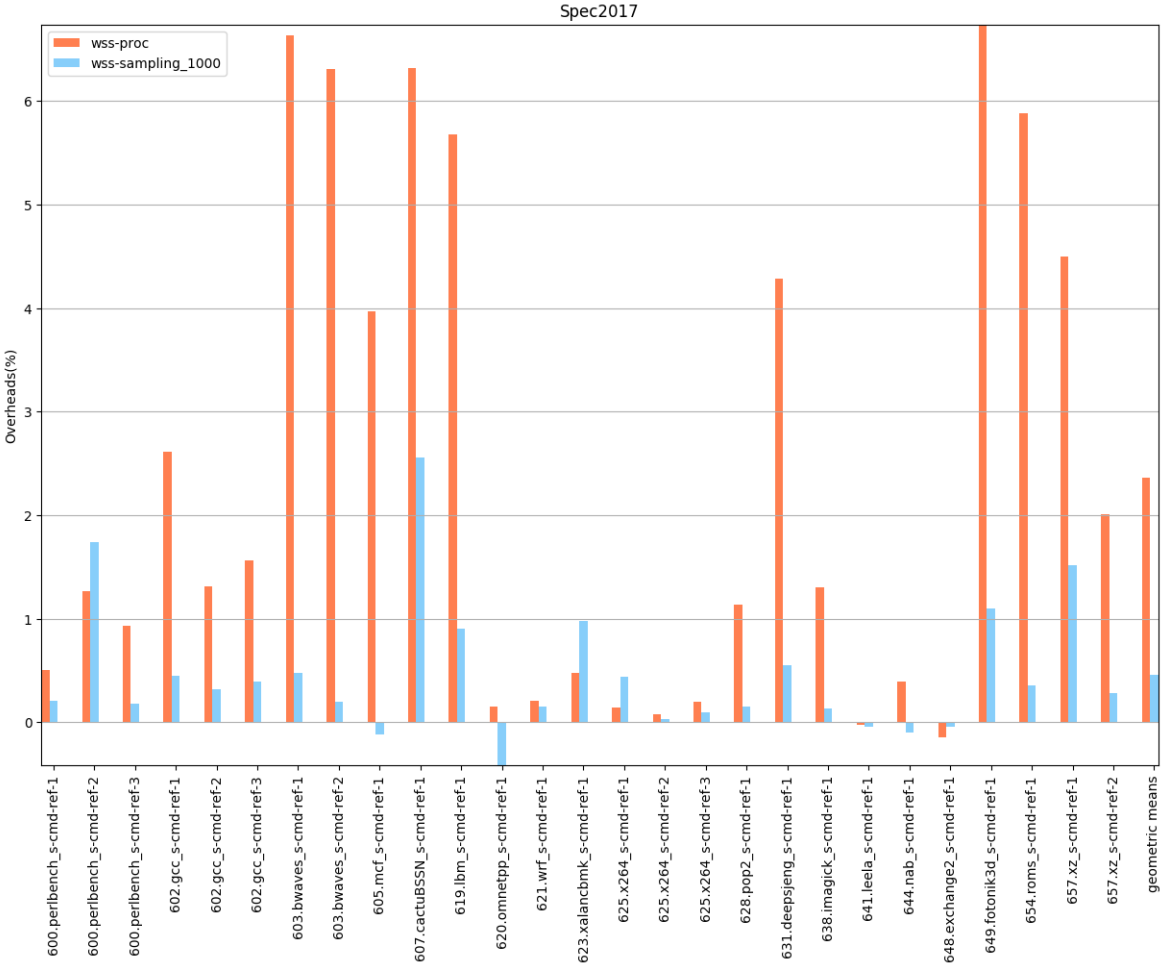
Σχήμα 5.321: 657.xz_s-ref-2[WSS, samples=1000, interval=20s, user level]



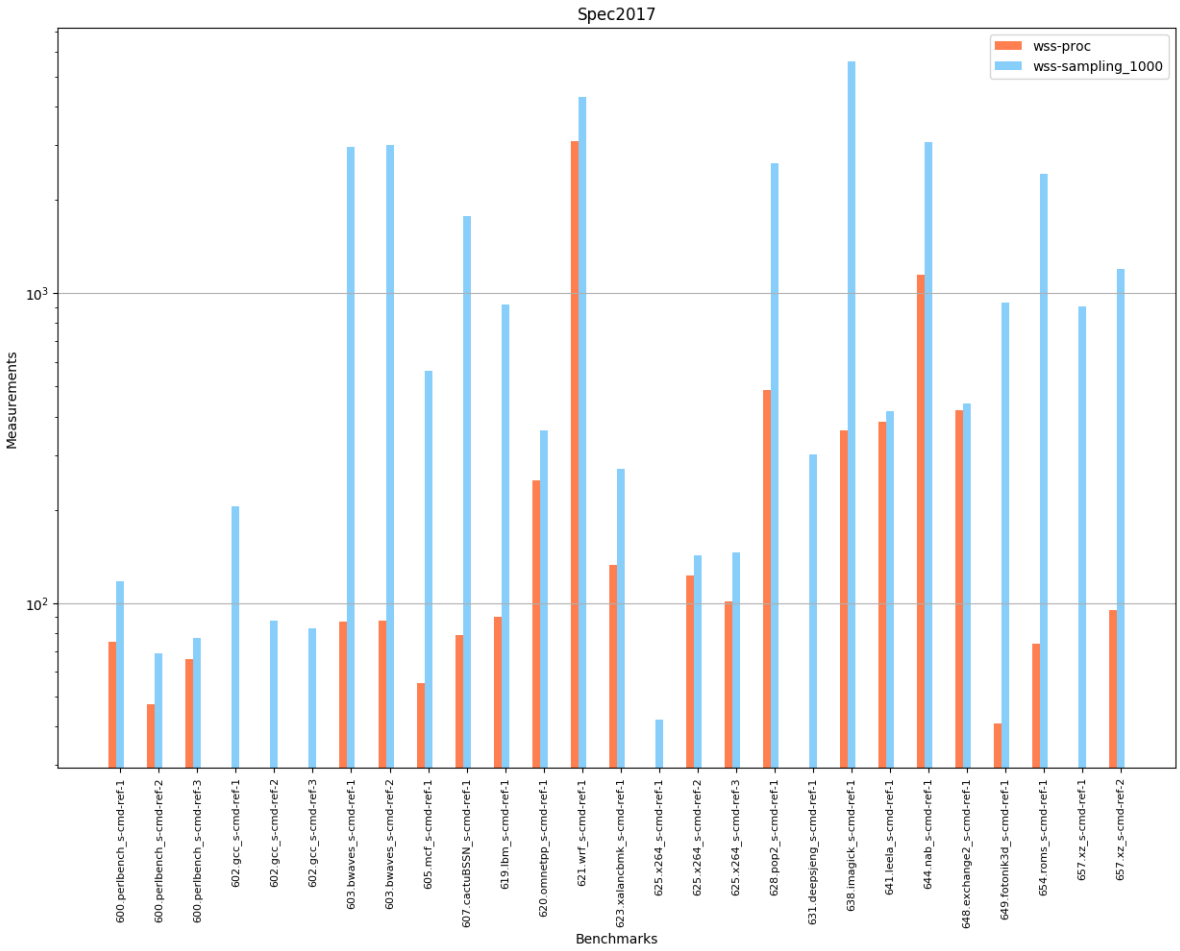
Σχήμα 5.322: 6xx benchmarks[Overheads, samples=1000, interval=1s, user level]



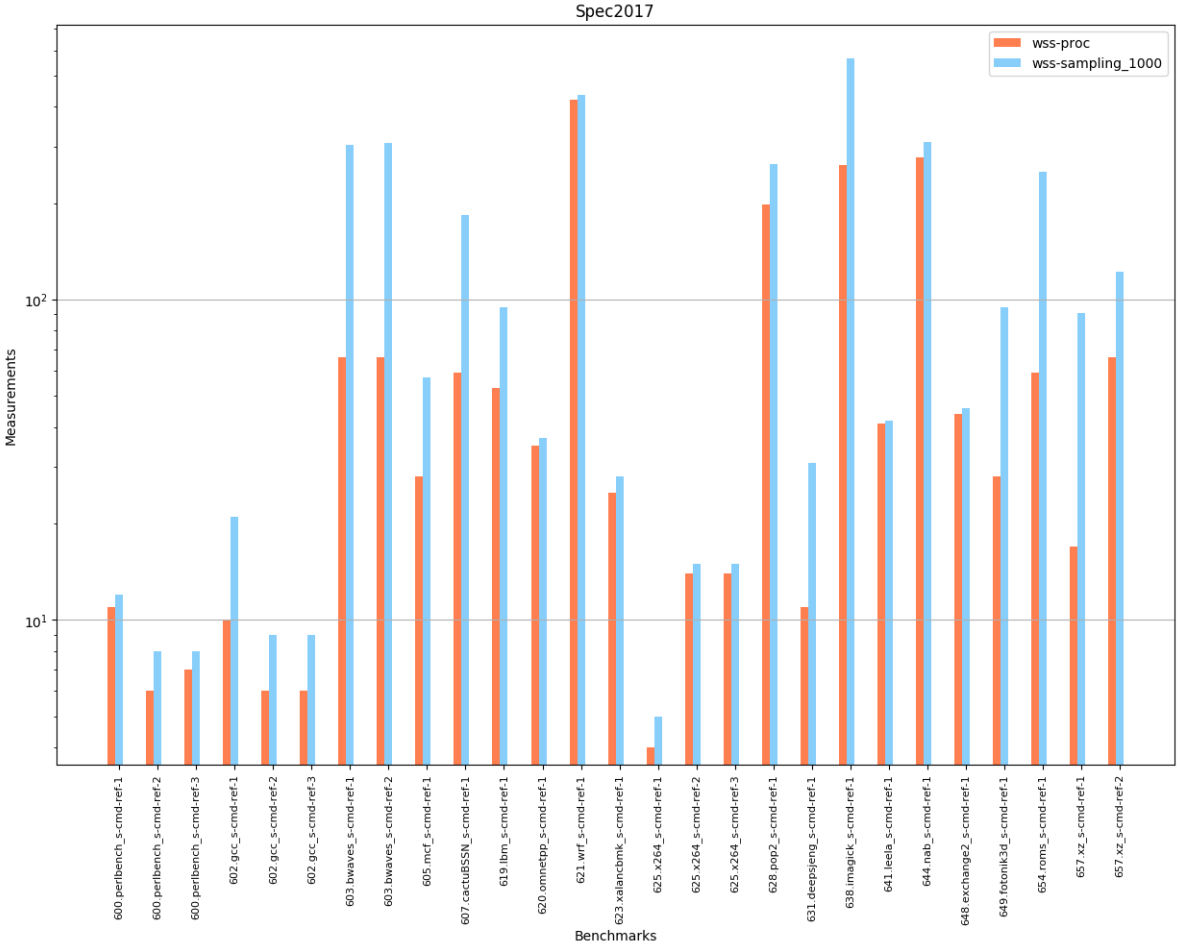
Σχήμα 5.323: 6xx benchmarks[Overheads, samples=1000, interval=10s, user level]



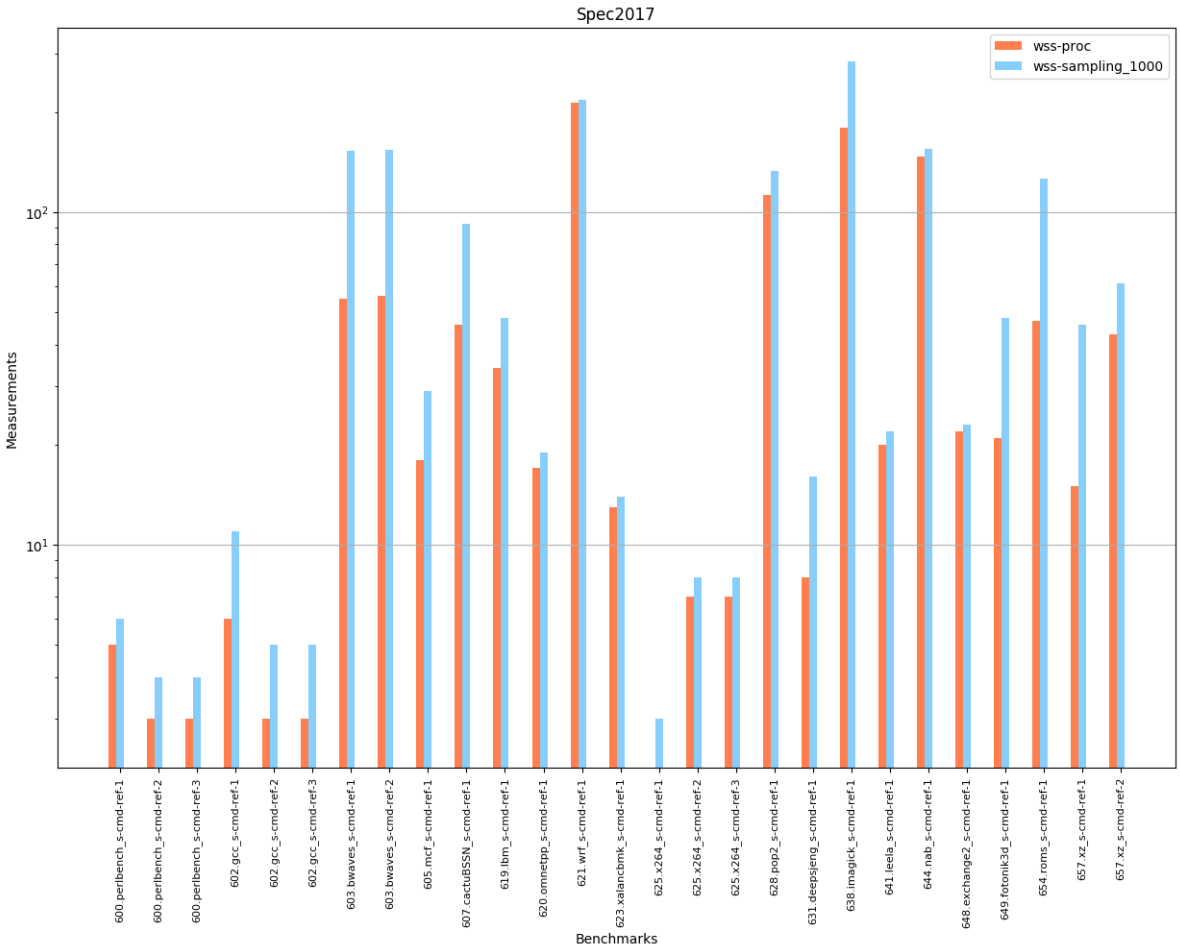
Σχήμα 5.324: 6xx benchmarks[Overheads, samples=1000, interval=20s, user level]



Σχήμα 5.325: 6xx benchmarks[Measurements, samples=1000, interval=1s, user level]



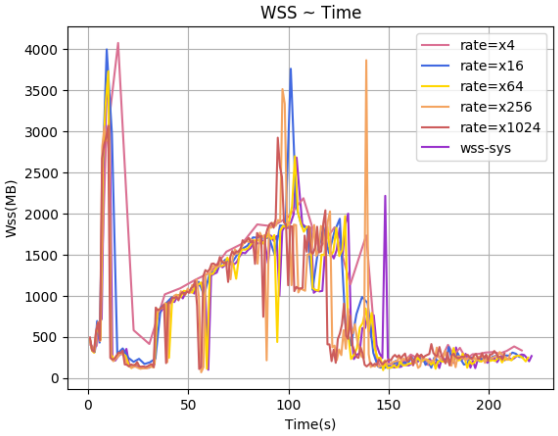
Σχήμα 5.326: 6xx benchmarks[Measurements, samples=1000, interval=10s, user level]



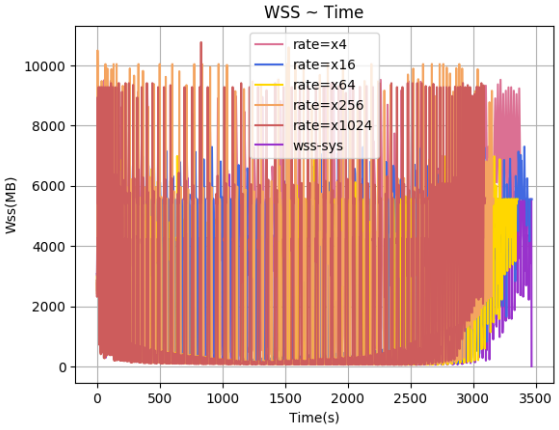
Σχήμα 5.327: 6xx benchmarks[Measurements, samples=1000, interval=20s, user level]

Sampling με ρυθμό δειγματοληψίας & Idle Page Tracking

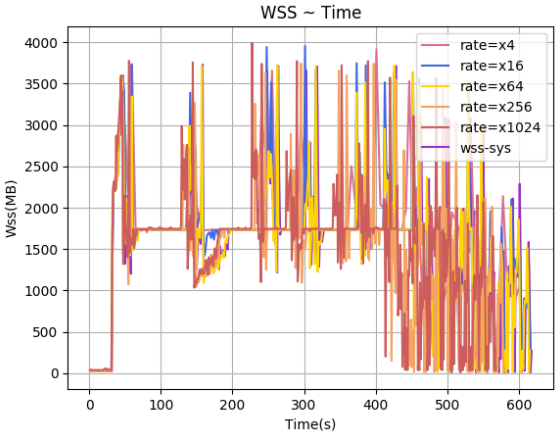
Στη συγκεκριμένη μέθοδο δειγματοληψίας ο αριθμός των δείγματων που λαμβάνουμε είναι αναλογικός με το συνολικό πλήθος των σελίδων της διεργασίας και λαμβάνεται με ρυθμό 4, 16, 64, 256 και 1024. Στα γραφήματα 5.328 - 5.335 απεικονίζονται οι εκτιμήσεις που προέκυψαν για το WSS των benchmarks 602.gcc_s, 603.bwaves_s, 605.mcf_s, 619.lbm_s, 623.xalancbmk_s, 628.pop2_s, 631.deepsjeng_s, 649.fotonik3d_s, ενώ στο 5.336 παρουσιάζονται τα overheads τους.



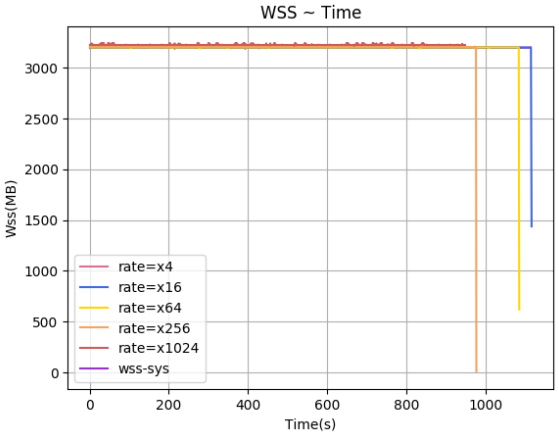
Σχήμα 5.328: 602.gcc_s-ref-1[WSS, rate=4,16,64,256,1024, interval=1s, user level]



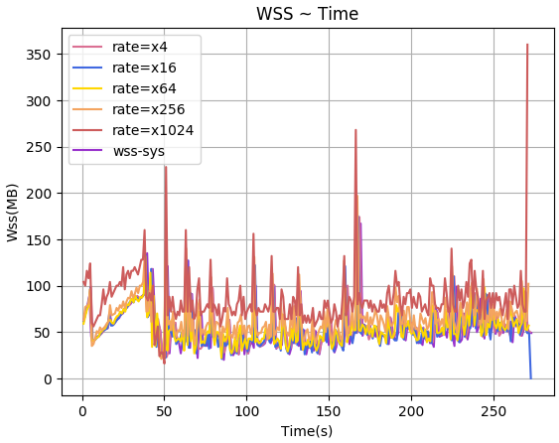
Σχήμα 5.329: 603.bwaves_s-ref-1[WSS, rate=4,16,64,256,1024, interval=1s, user level]



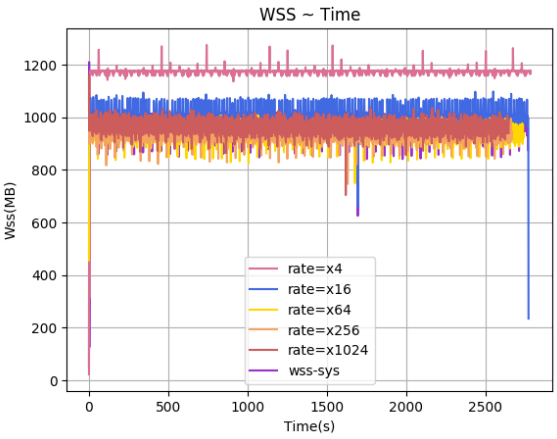
Σχήμα 5.330: 605.mcf_s-ref-1[WSS, rate=4,16,64,256,1024, interval=1s, user level]



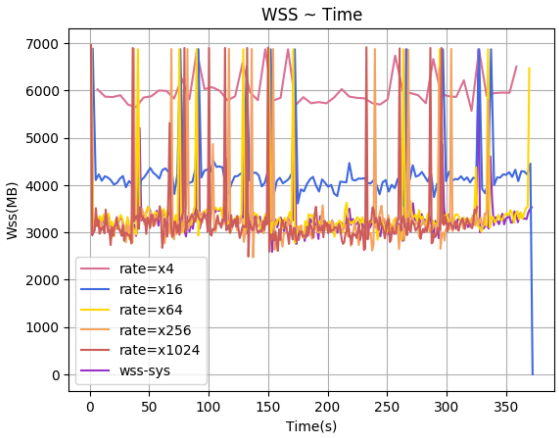
Σχήμα 5.331: 619.lbm_s-ref-1[WSS, rate=4,16,64,256,1024, interval=1s, user level]



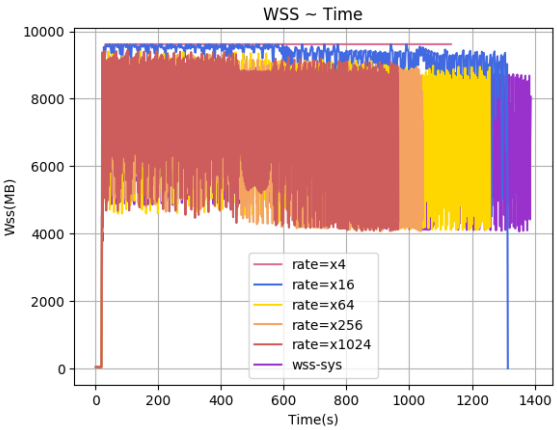
Σχήμα 5.332: 623.xalancbmk_s-ref-1[WSS, rate=4,16,64,256,1024, interval=1s, user level]



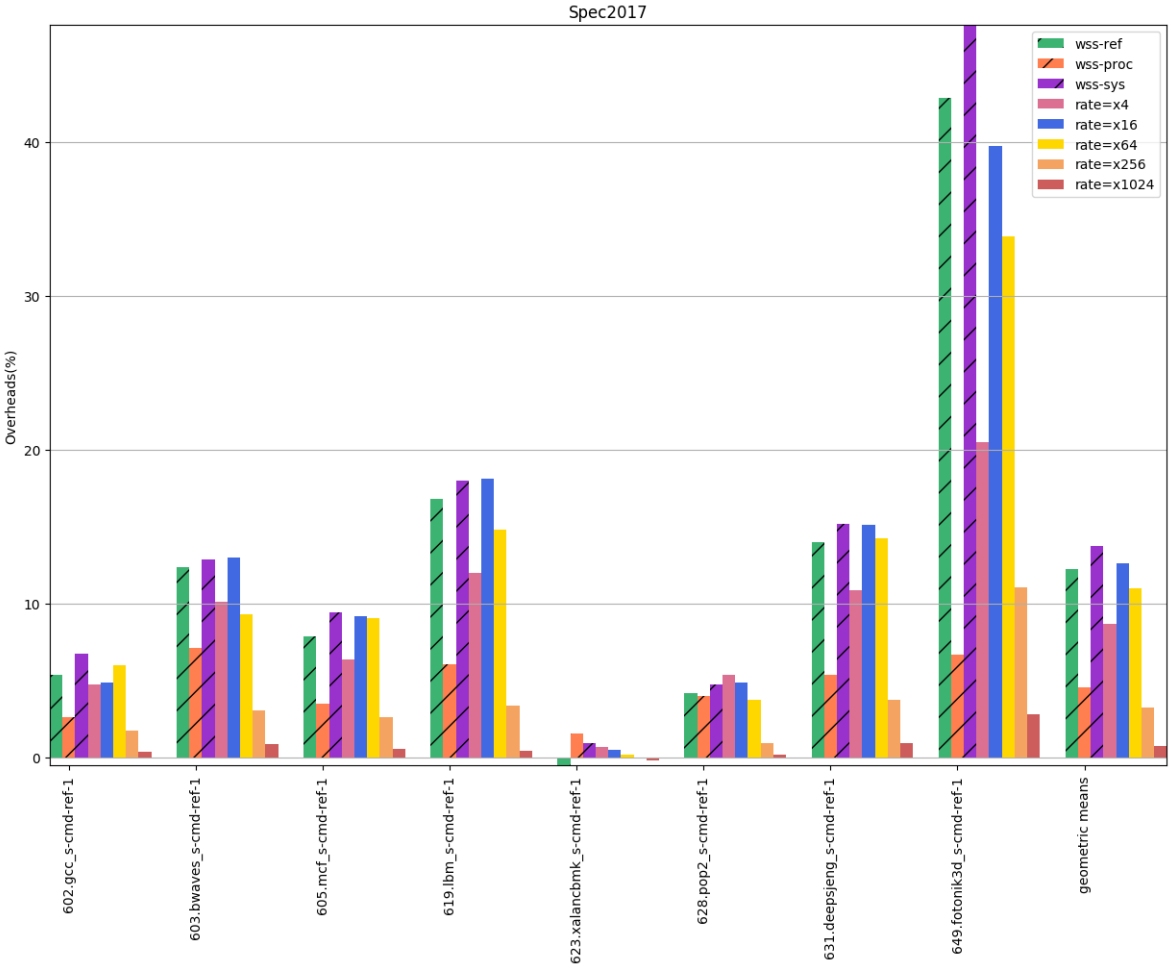
Σχήμα 5.333: 628.pop2_s-ref-1[WSS, rate=4,16,64,256,1024, interval=1s, user level]



Σχήμα 5.334: 631.deepsjeng_s-ref-1[WSS, rate=4,16,64,256,1024, interval=1s, user level]



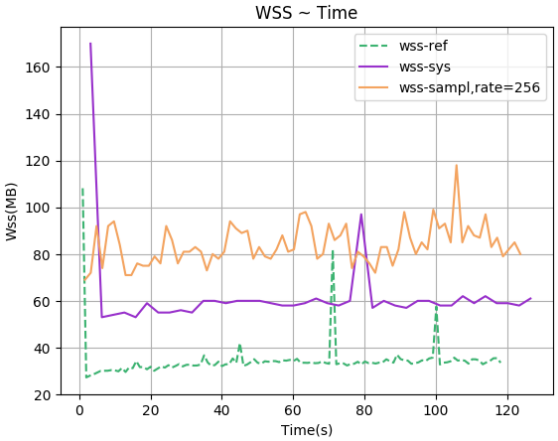
Σχήμα 5.335: 649.fotonik3d_s-ref-1[WSS, rate=4,16,64,256,1024, interval=1s, user level]



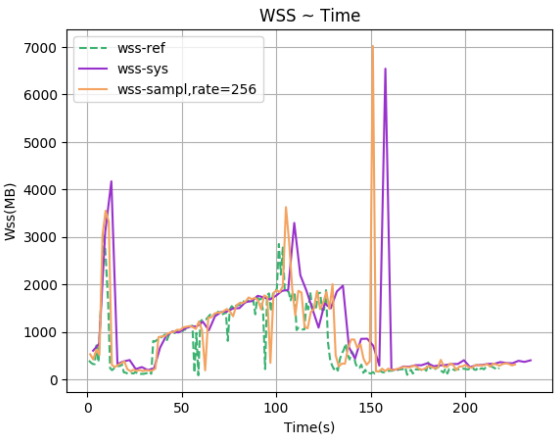
Σχήμα 5.336: 6xx benchmarks[Overheads, rate=4,16,64,256,1024, interval=1s, user level]

Sampling με ρυθμό δειγματοληψίας & Idle Page Tracking σε εικονικό περιβάλλον Linux

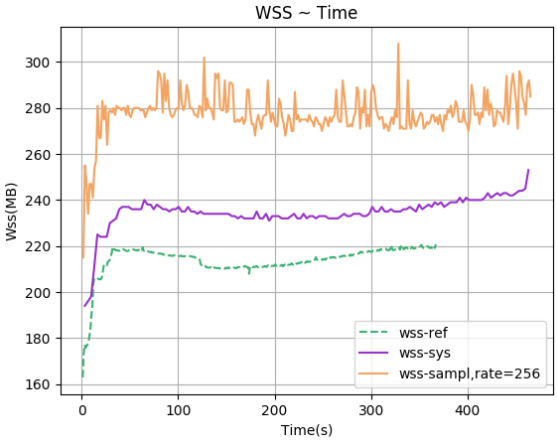
Στα παρακάτω πειράματα εφαρμόσαμε αρχικά τη μέθοδο idle page tracking εκτελώντας το εργαλείο wss-sys με διάστημα παρακολούθησης 1s στον host για να λάβουμε μετρήσεις για το working set ορισμένων εκ των 6xx benchmarks που έτρεχαν στον guest. Έπειτα, εφαρμόσαμε για τον ίδιο σκοπό τη μέθοδο δειγματοληψίας με ρυθμό 256 στον host για εφαρμογές του guest. Στα γραφήματα 5.337 - 5.344 απεικονίζονται οι εκτιμήσεις που προέκυψαν για το WSS των benchmarks 600.perlbench_s, 602.gcc_s, 620.omnetpp_s, 623.xalancbmk_s, 625.x264_s, 631.deepsjeng_s, 641.leela_s, 648.exchange_s, ενώ στα 5.345 και 5.346 παρουσιάζονται τα overheads και το πλήθος μετρήσεών τους. Η μπάρα των μετρήσεων του wss-ref αναφέρεται στα αποτελέσματα του συγκεκριμένου εργαλείου όταν αυτό εκτελέστηκε σε περιβάλλον guest ώστε να συγκρίνουμε τη διαφορά εκτέλεσης σε host.



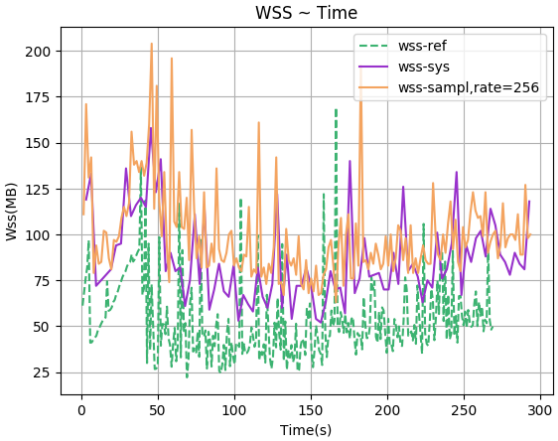
Σχήμα 5.337: 600.perlbench_s-ref-1[WSS, rate=256, interval=1s, user level, virtual]



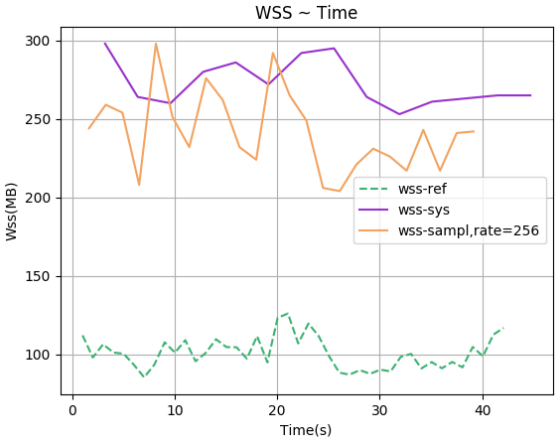
Σχήμα 5.338: 602.gcc_s-ref-1[WSS, rate=256, interval=1s, user level, virtual]



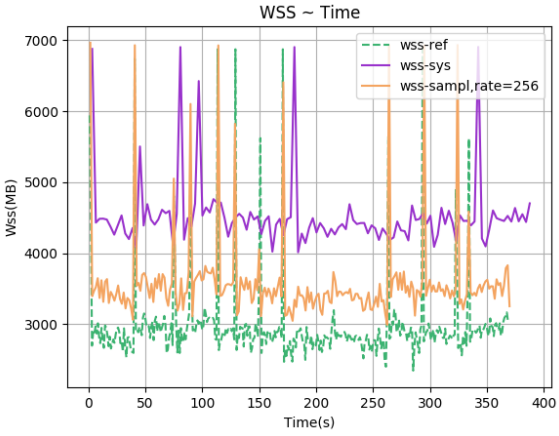
Σχήμα 5.339: 620.omnetpp_s-ref-1[WSS, rate=256, interval=1s, user level, virtual]



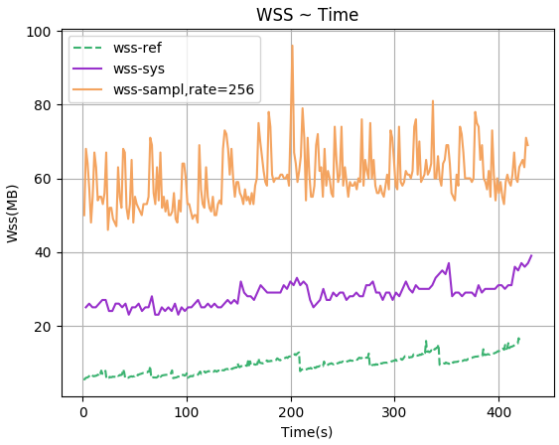
Σχήμα 5.340: 623.xalancbmk_s-ref-1[WSS, rate=256, interval=1s, user level, virtual]



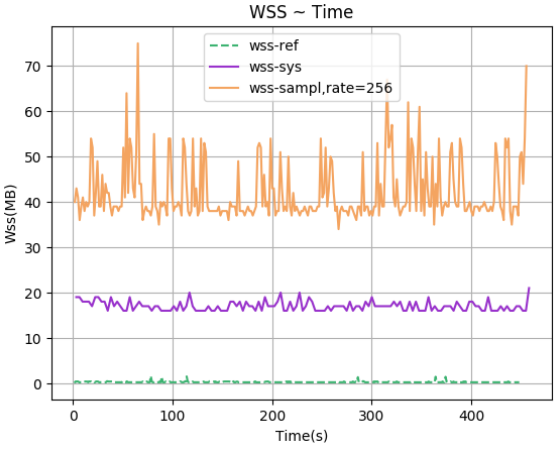
Σχήμα 5.341: 625.x264_s-ref-1[WSS, rate=256, interval=1s, user level, virtual]



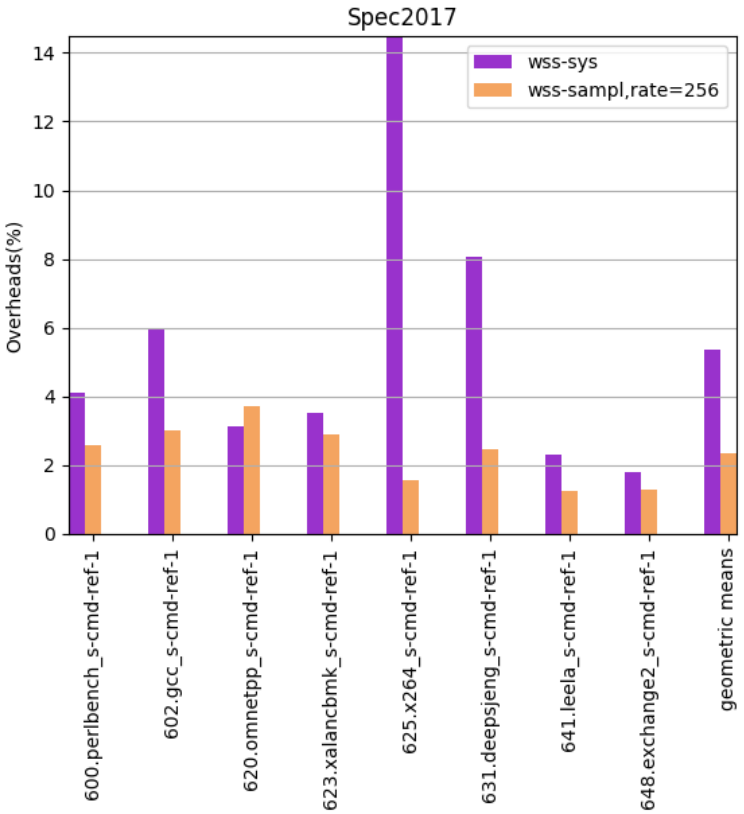
Σχήμα 5.342: 631.deepsjeng_s-ref-1[WSS, rate=256, interval=1s, user level, virtual]



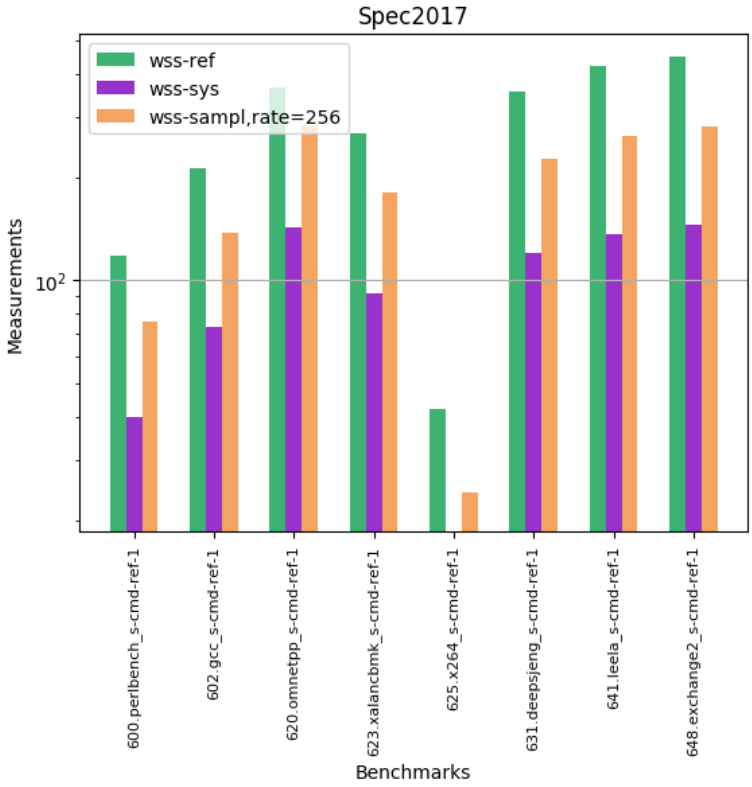
Σχήμα 5.343: 641.leela_s-ref-1[WSS, rate=256, interval=1s, user level, virtual]



Σχήμα 5.344: 648.exchange2_s-ref-1[WSS, rate=256, interval=1s, user level, virtual]



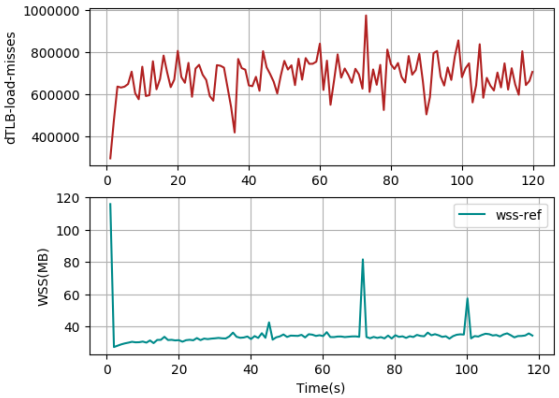
Σχήμα 5.345: 6xx benchmarks[Overheads, rate=256, interval=1s, user level, virtual]



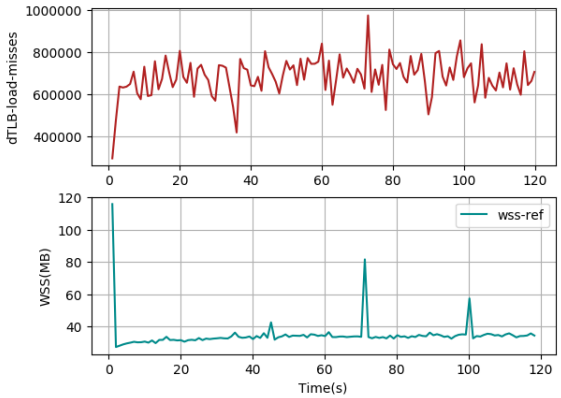
Σχήμα 5.346: 6xx benchmarks[Measurements, rate=256, interval=1s, user level, virtual]

5.1.3 Intermittent Page Tracking

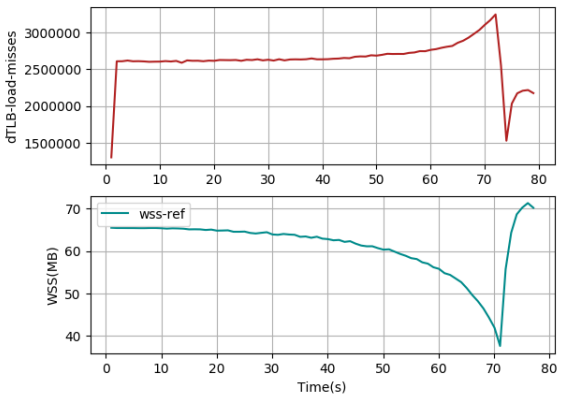
Τα παρακάτω πειράματα έγιναν με την τεχνική διακοπτόμενης παρακολούθησης του working set και με τη χρήση του εργαλείου wss-ref που ελέγχει τα referenced flags. Αρχικά εκτελέσαμε τα benchmarks μόνο με το εργαλείο perf stat για να επιβεβαιώσουμε ότι οι αλλαγές στα dTLB-load-misses σηματοδοτούν αλλαγές και στο working set των εφαρμογών. Η επιλογή του wss-ref έγινε λόγω του μειωμένου overhead που εισάγει στα προγράμματα σε σχέση με τα άλλα εργαλεία παρακολούθησης. Όπως αναλύσαμε στο Κεφάλαιο 4 θεωρούμε πως το πρόγραμμα βρίσκεται σε σταθερή φάση όταν το σφάλμα err_r είναι εντός κάποιου επιθυμητού εύρους. Το ίδιο εύρος σφάλματος θεωρούμε και για τις μετρήσεις wss, ώστε να κρίνουμε πάλι το είδος της φάσης. Τα πειράματα που εκτελέσαμε έγιναν με εύρη ±10%,20%,30%. Στα γραφήματα 5.347 - 5.373 απεικονίζονται τα wss όπως μετρήθηκαν με το wss-ref παράλληλα με το perf stat για την παρακολούθηση των dTLB-load-misses, ενώ στα 5.374 - 5.400 παρουσιάζονται οι εκτιμήσεις για το wss των εφαρμογών με χρήση Intermittent Page Tracking για τα διάφορα προαναφερθέντα εύρη σφάλματος. Τέλος, στα 5.401 και 5.402 παρουσιάζονται αντίστοιχα τα overheads για τα διάφορα εύρη σφάλματος και τα ποσοστά μείωσης των μετρήσεων που λαμβάνουμε με καθένα από αυτά.



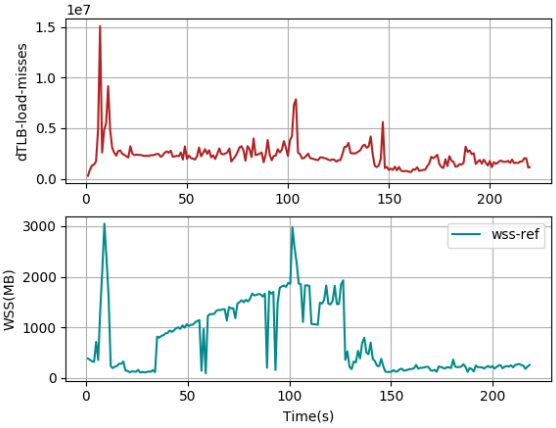
Σχήμα 5.347: 600.perlbench_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



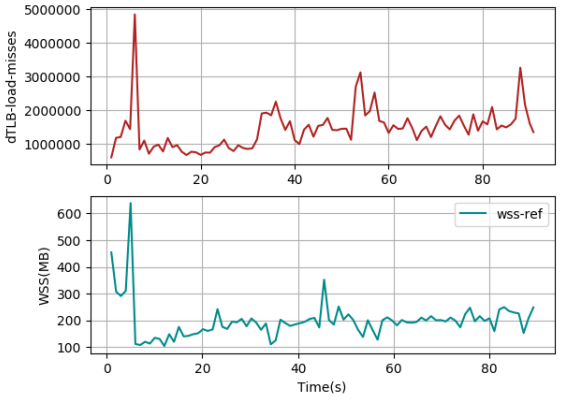
Σχήμα 5.348: 600.perlbench_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



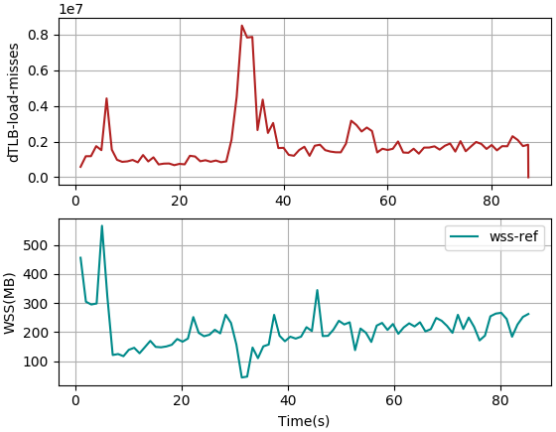
Σχήμα 5.349: 600.perlbench_s-ref-3[WSS & dTLB-load-misses, interval=1s, user level]



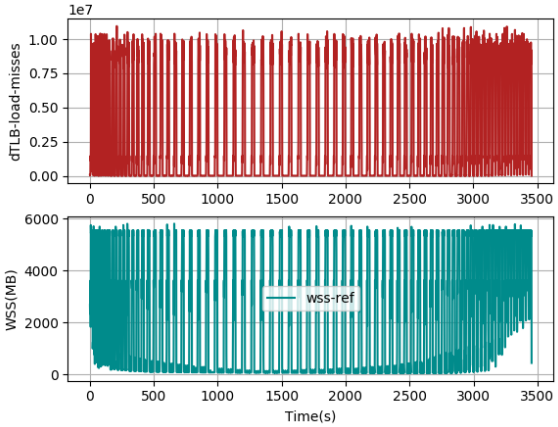
Σχήμα 5.350: 602.gcc_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



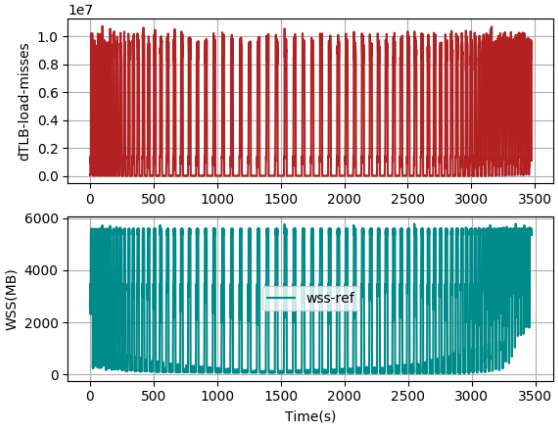
Σχήμα 5.351: 602.gcc_s-ref-2[WSS & dTLB-load-misses, interval=1s, user level]



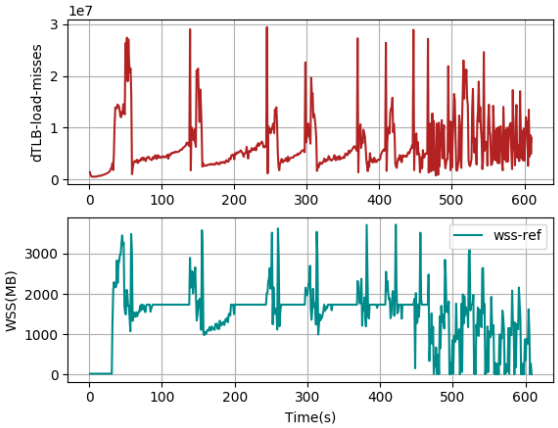
Σχήμα 5.352: 602.gcc_s-ref-3[WSS & dTLB-load-misses, interval=1s, user level]



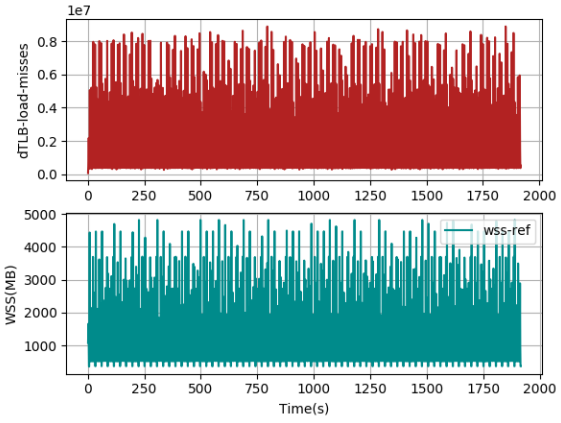
Σχήμα 5.353: 603.bwaves_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



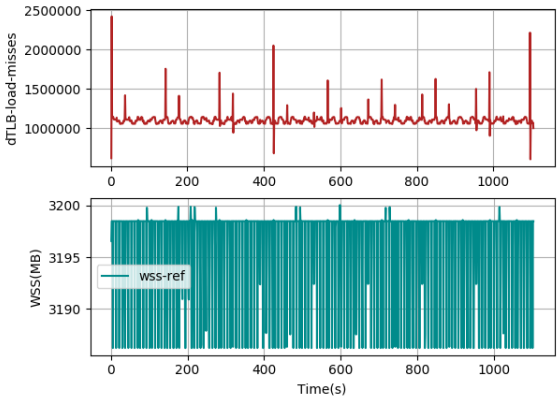
Σχήμα 5.354: 603.bwaves_s-ref-2[WSS & dTLB-load-misses, interval=1s, user level]



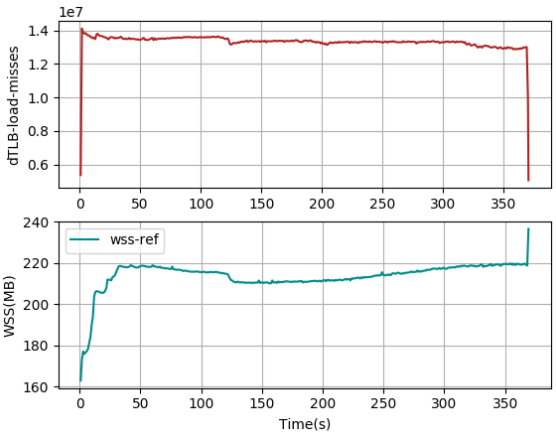
Σχήμα 5.355: 605.mcf_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



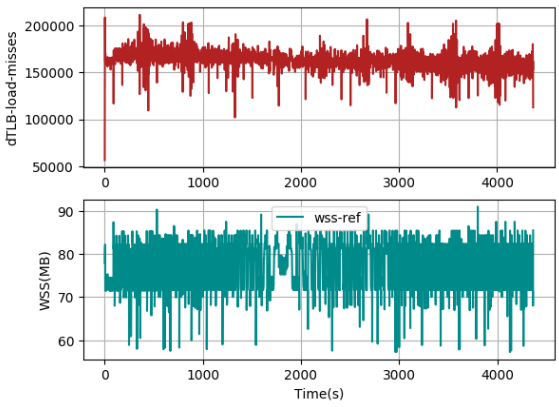
Σχήμα 5.356: 607.cactuBSSN_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



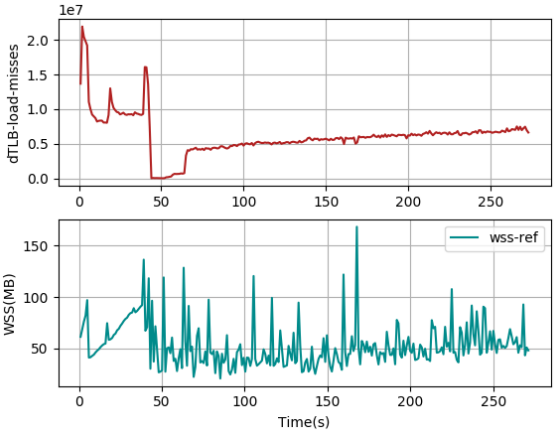
Σχήμα 5.357: 619.lbm_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



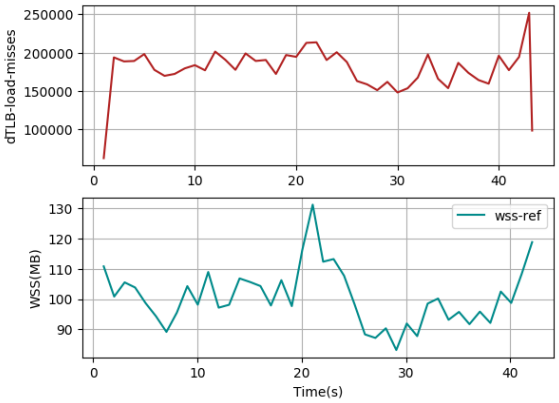
Σχήμα 5.358: 620.omnetpp_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



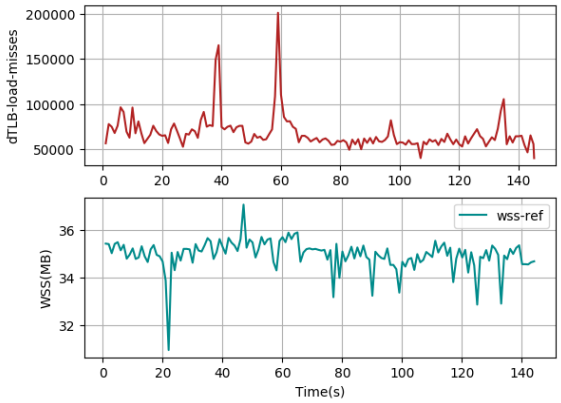
Σχήμα 5.359: 621.wrf_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



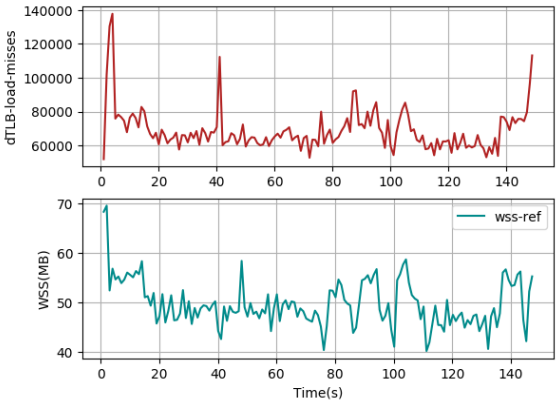
Σχήμα 5.360: 623.xalancbmk_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



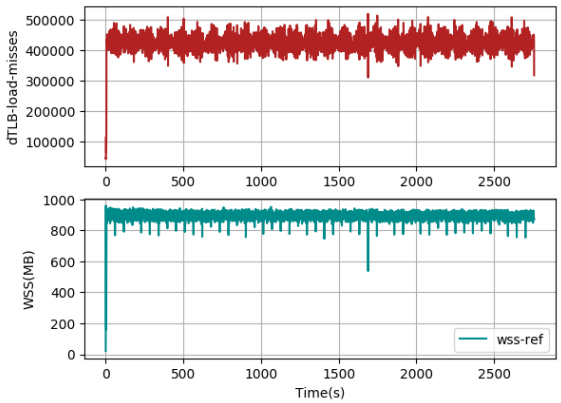
Σχήμα 5.361: 625.x264_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



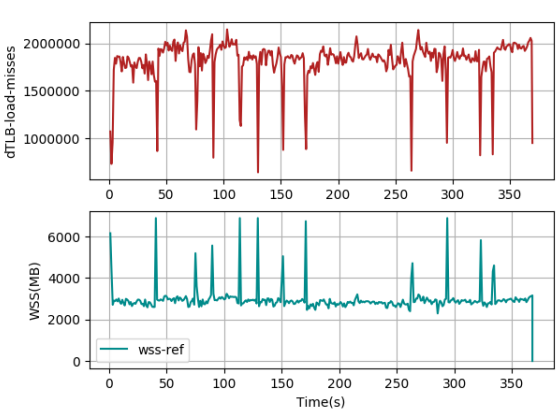
Σχήμα 5.362: 625.x264_s-ref-2[WSS & dTLB-load-misses, interval=1s, user level]



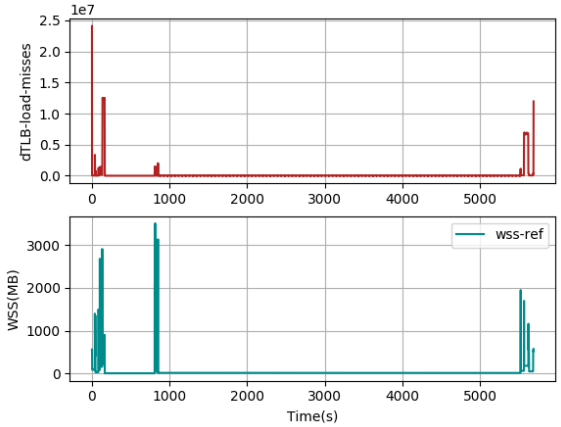
Σχήμα 5.363: 625.x264_s-ref-3[WSS & dTLB-load-misses, interval=1s, user level]



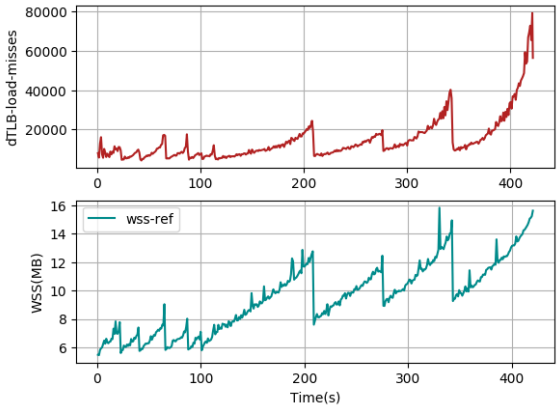
Σχήμα 5.364: 628.pop2_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



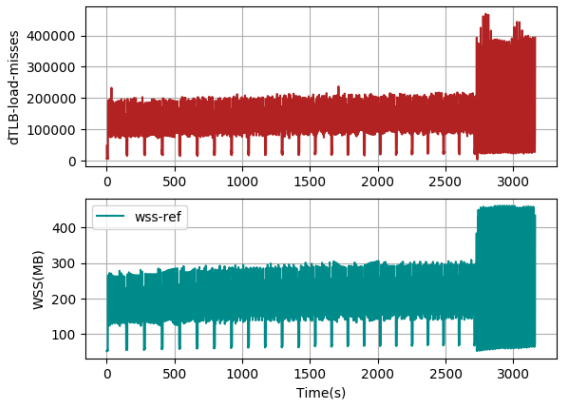
Σχήμα 5.365: 631.deepsjeng_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



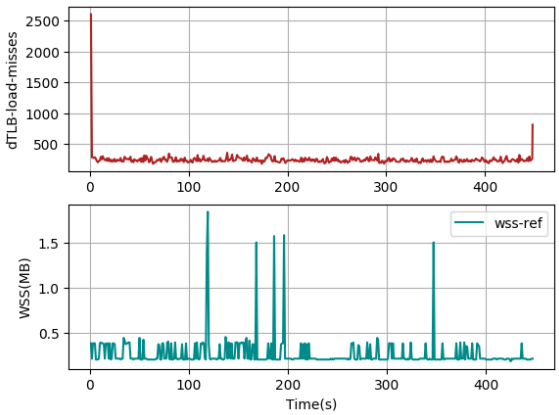
Σχήμα 5.366: 638.imagick_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



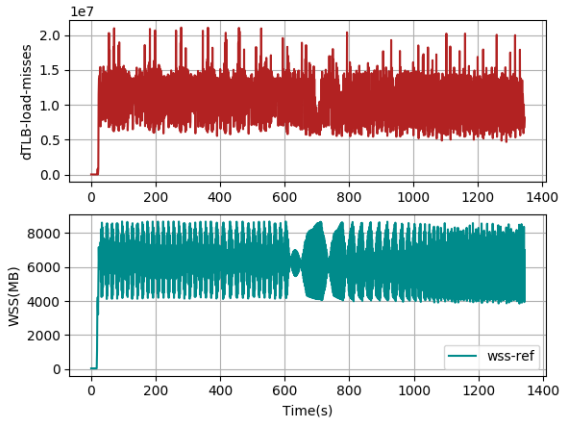
Σχήμα 5.367: 641.leela_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



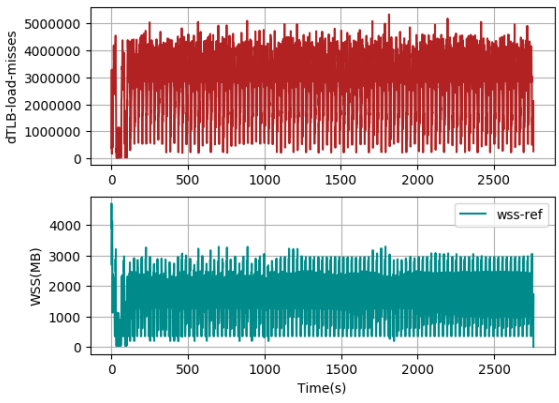
Σχήμα 5.368: 644.nab_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



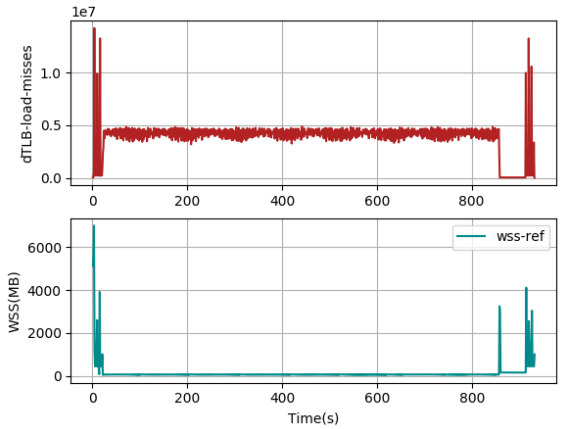
Σχήμα 5.369: 648.exchange2_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



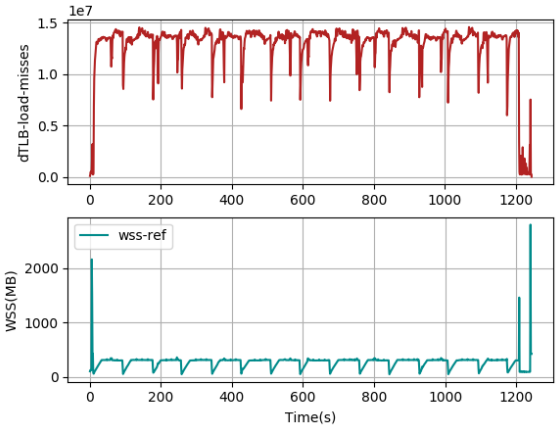
Σχήμα 5.370: 649.fotonik3d_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



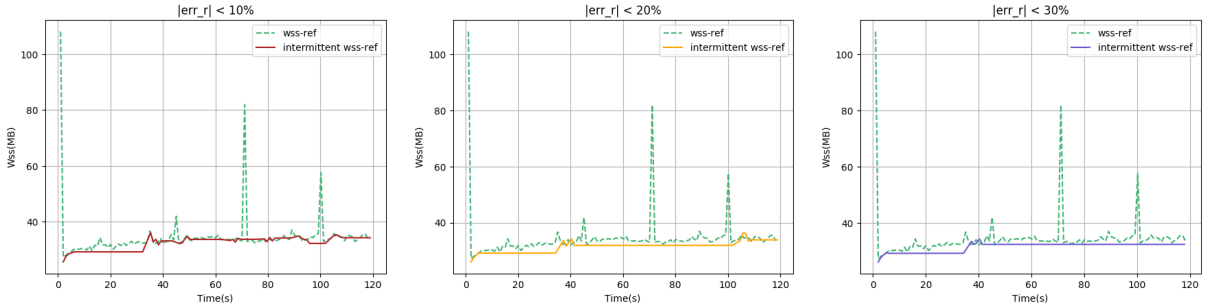
Σχήμα 5.371: 654.roms_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]



Σχήμα 5.372: 657.xz_s-ref-1[WSS & dTLB-load-misses, interval=1s, user level]

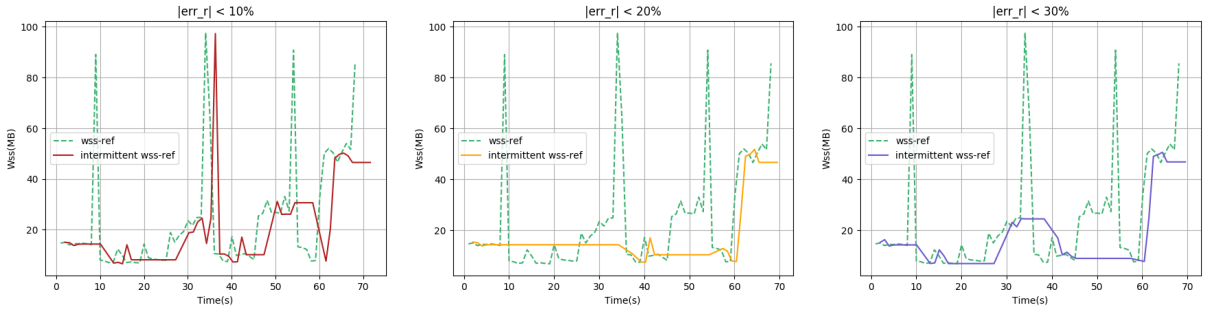


Σχήμα 5.373: 657.xz_s-ref-2[WSS & dTLB-load-misses, interval=1s, user level]



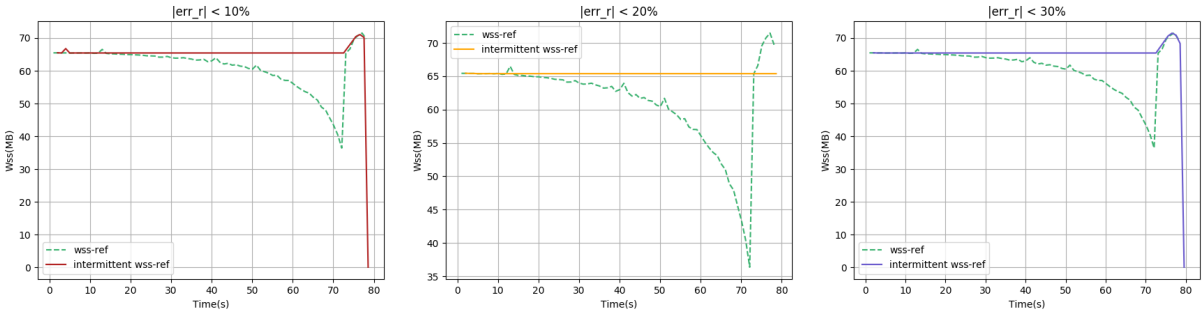
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.374: 600.perlbench_s-ref-1[WSS, interval=1s, user level]



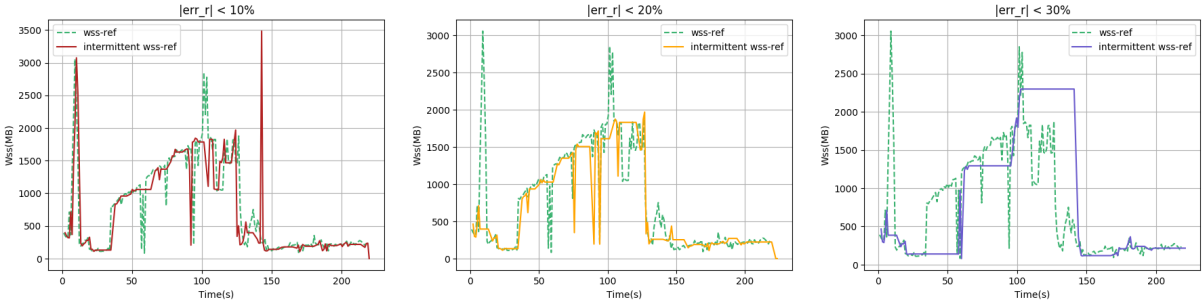
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.375: 600.perlbench_s-ref-2[WSS, interval=1s, user level]



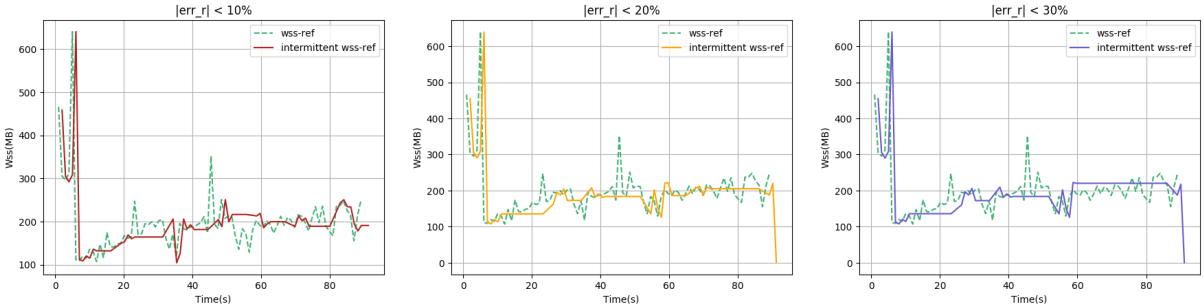
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.376: 600.perlbench_s-ref-3[WSS, interval=1s, user level]



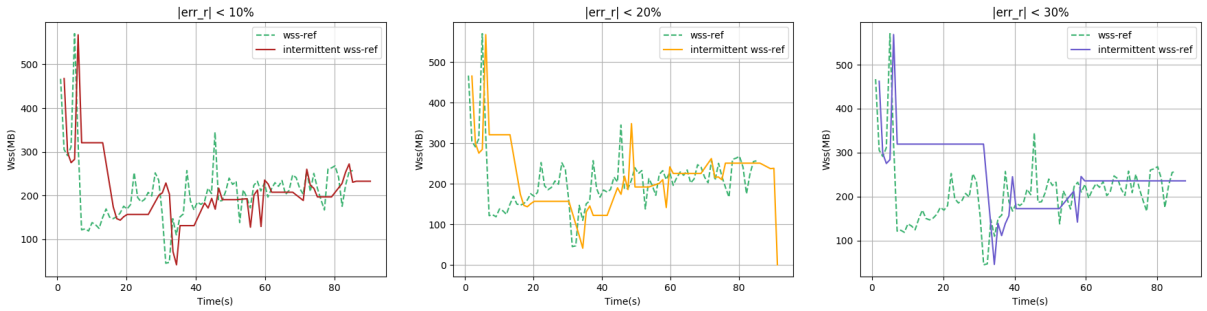
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.377: 602.gcc_s-ref-1[WSS, interval=1s, user level]



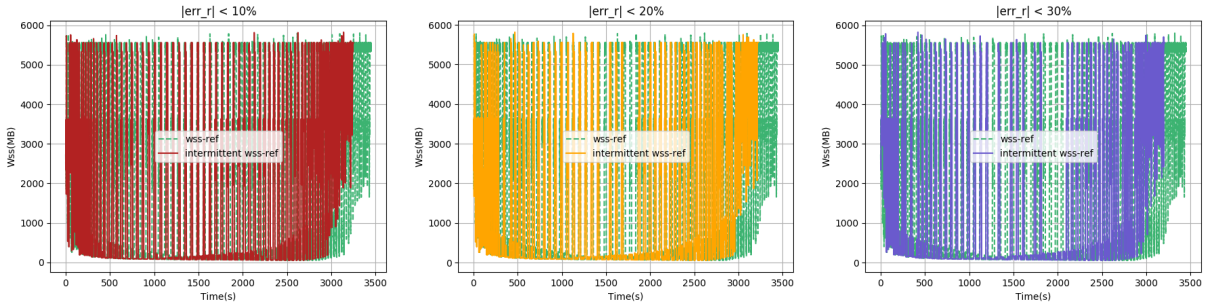
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.378: 602.gcc_s-ref-2[WSS, interval=1s, user level]



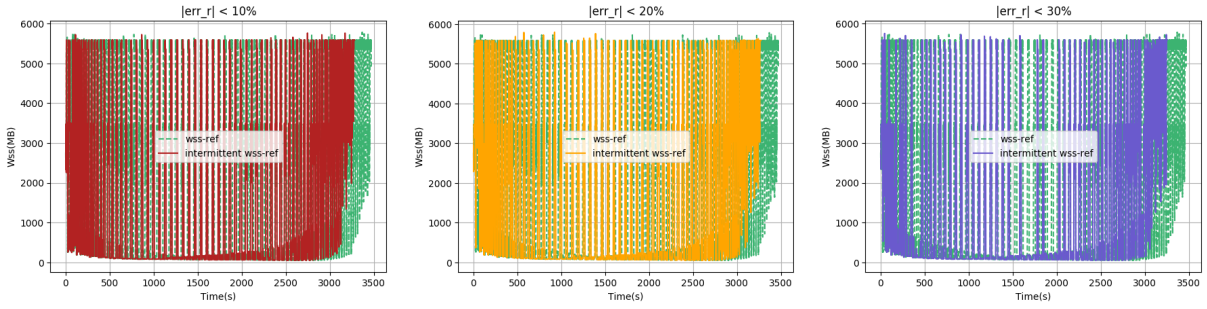
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.379: 602.gcc_s-ref-3[WSS, interval=1s, user level]



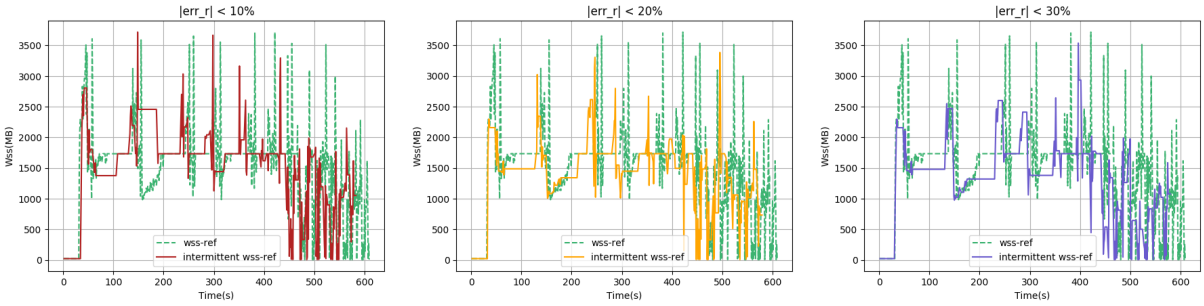
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.380: 603.bwaves_s-ref-1[WSS, interval=1s, user level]



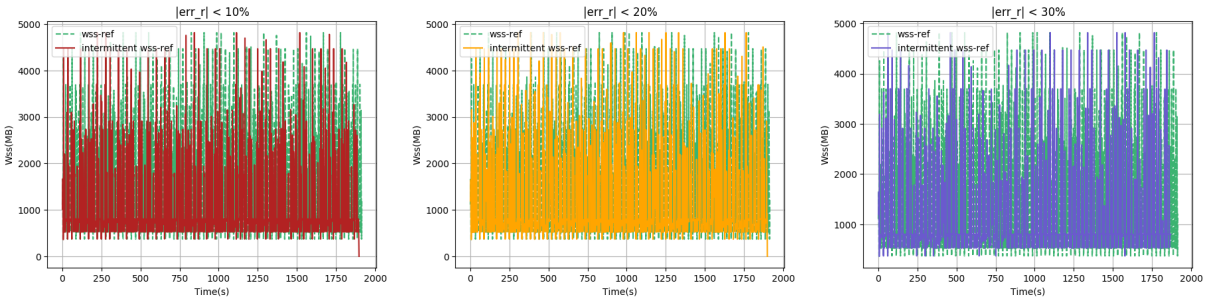
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.381: 603.bwaves_s-ref-2[WSS, interval=1s, user level]



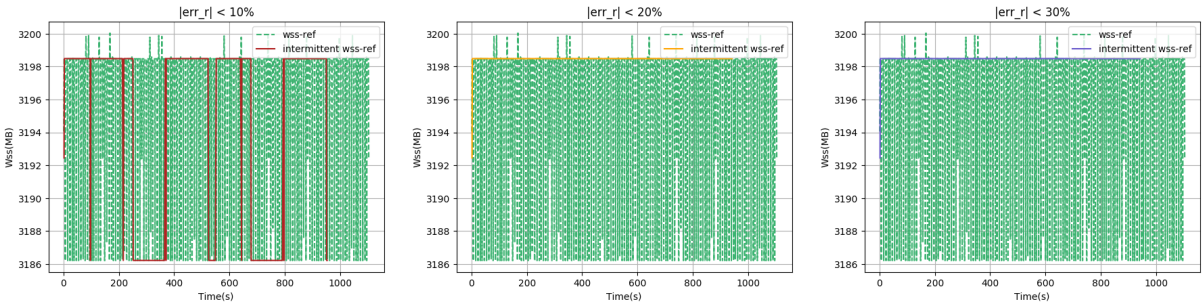
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.382: 605.mcf_s-ref-1[WSS, interval=1s, user level]



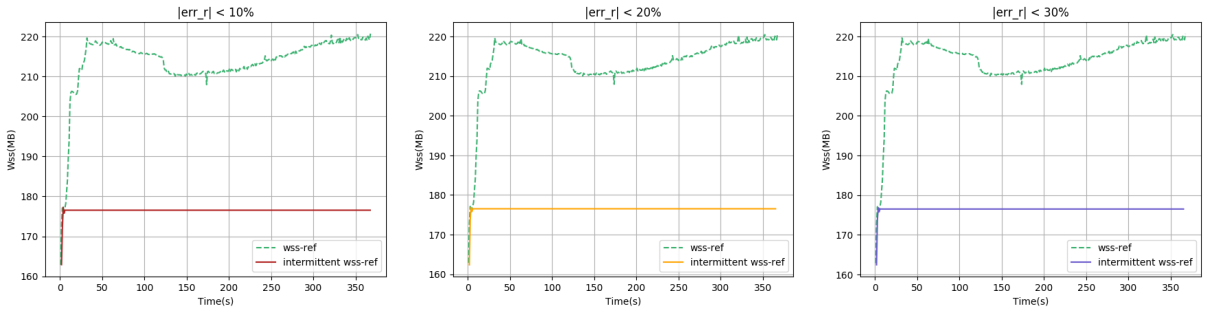
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.383: 607.cactuBSSN_s-ref-1[WSS, interval=1s, user level]



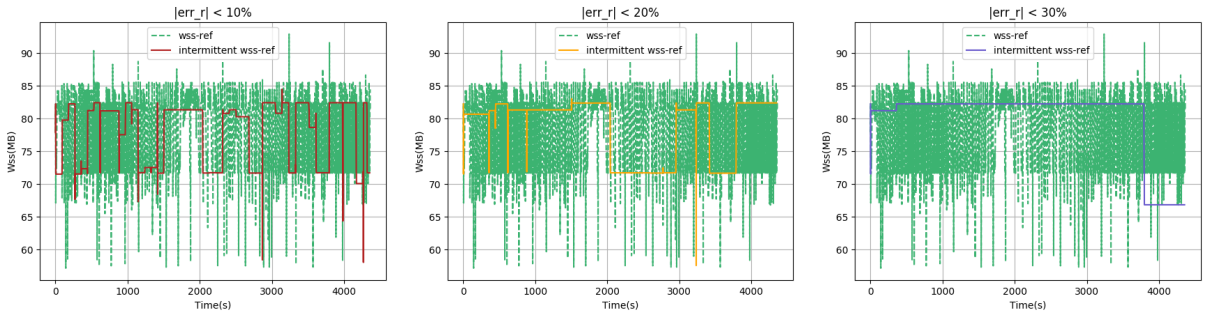
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.384: 619.lbm_s-ref-1[WSS, interval=1s, user level]



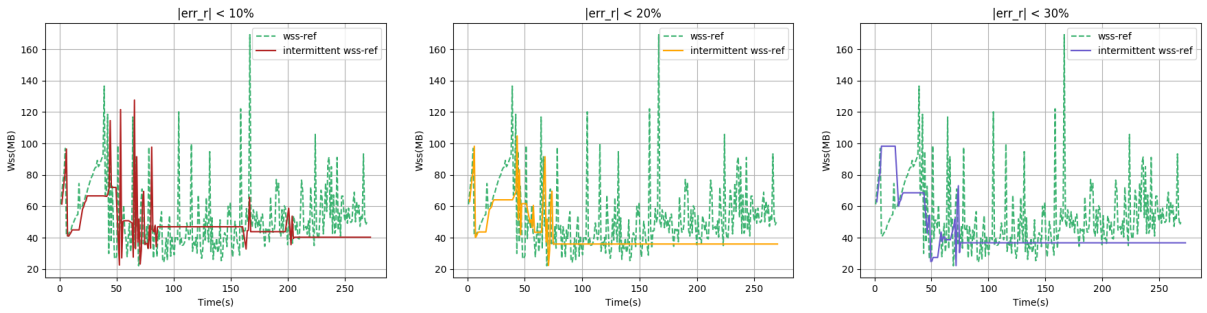
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.385: 620.omnetpp_s-ref-1[WSS, interval=1s, user level]



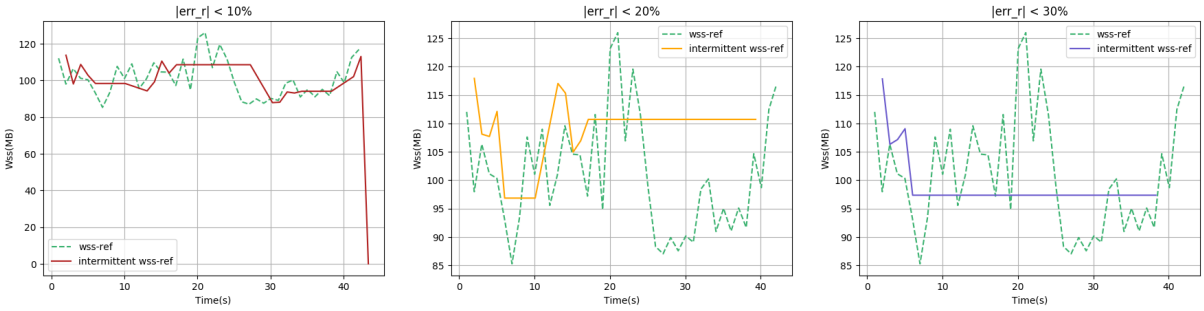
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.386: 621.wrf_s-ref-1[WSS, interval=1s, user level]



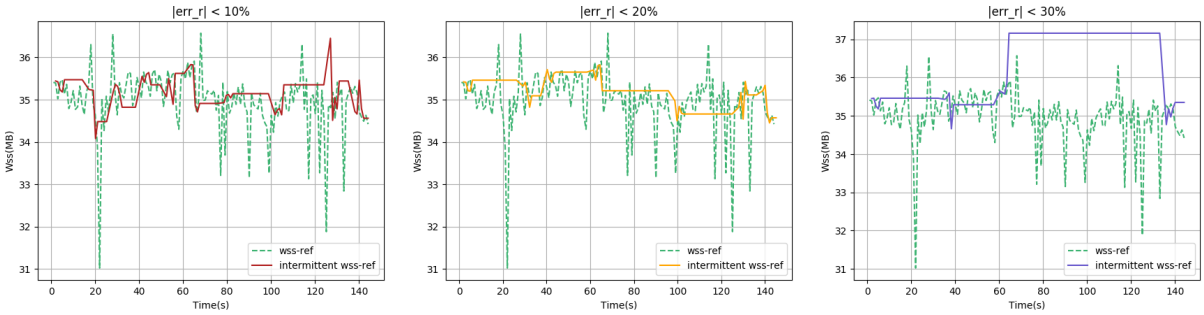
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.387: 623.xalancbmk_s-ref-1[WSS, interval=1s, user level]



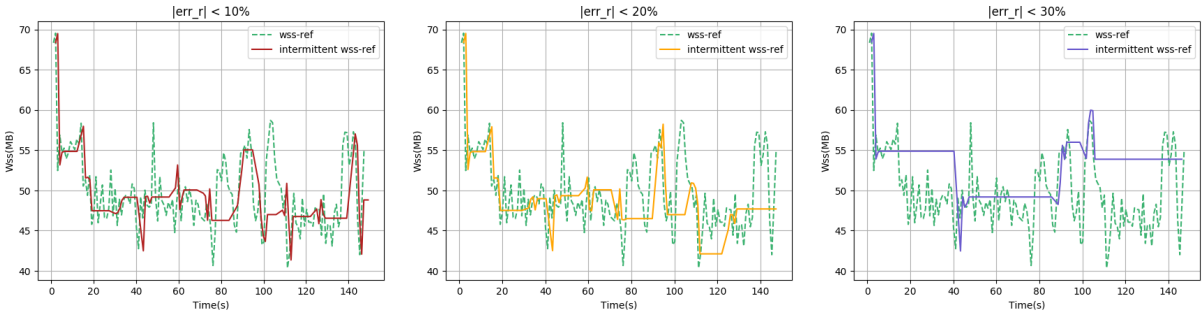
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.388: 625.x264_s-ref-1[WSS, interval=1s, user level]



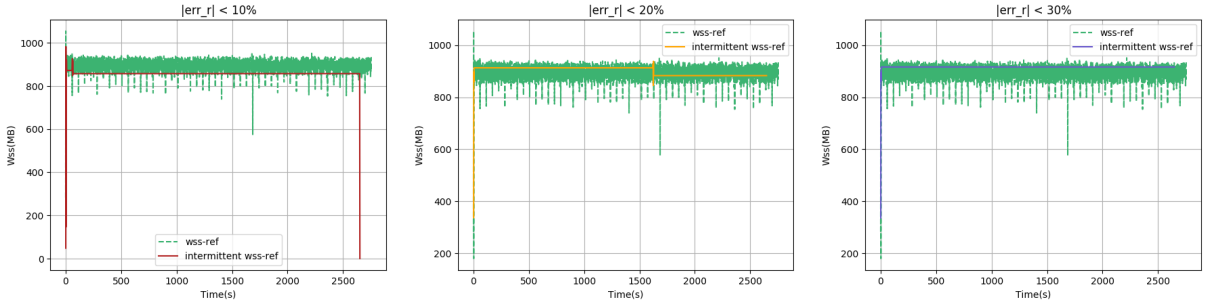
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.389: 625.x264_s-ref-2[WSS, interval=1s, user level]



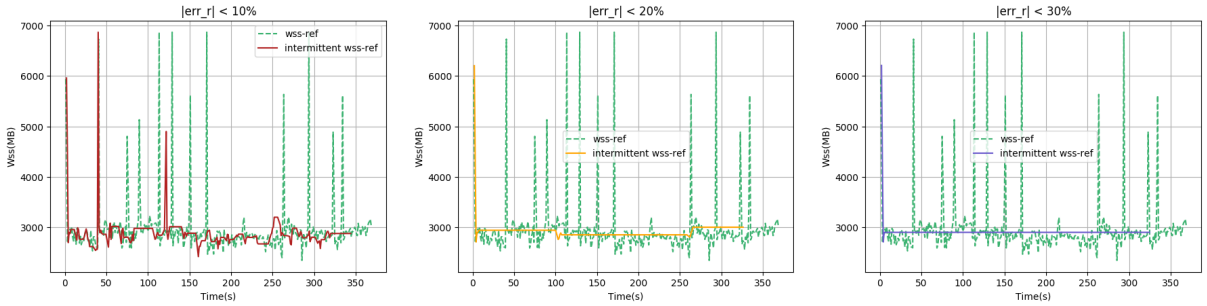
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.390: 625.x264_s-ref-3[WSS, interval=1s, user level]



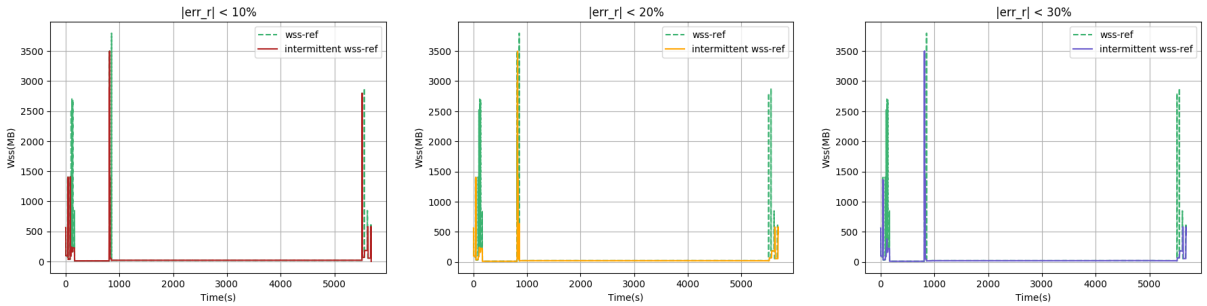
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.391: 628.pop2_s-ref-1[WSS, interval=1s, user level]



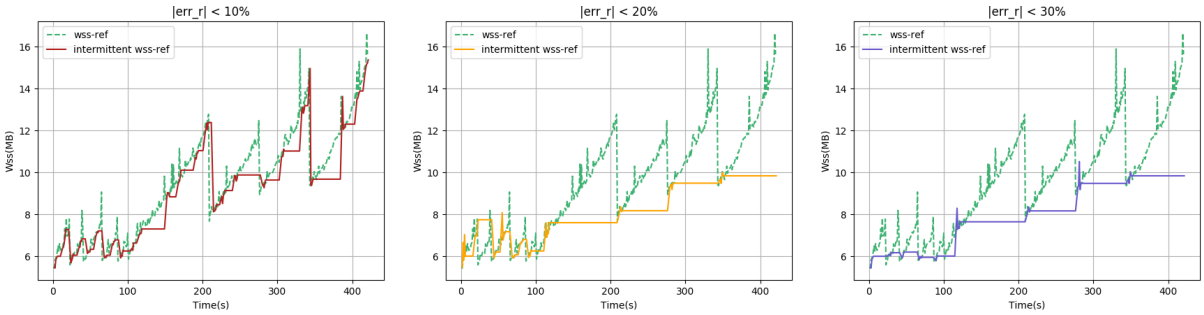
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.392: 631.deepsjeng_s-ref-1[WSS, interval=1s, user level]



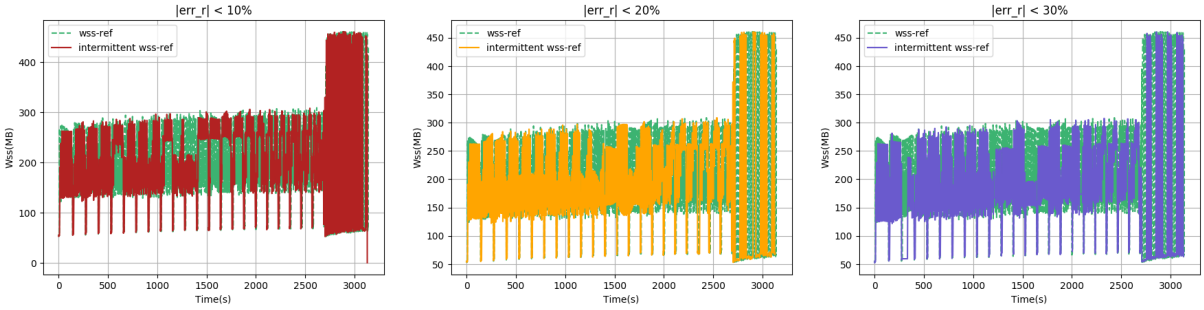
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.393: 638.imagick_s-ref-1[WSS, interval=1s, user level]



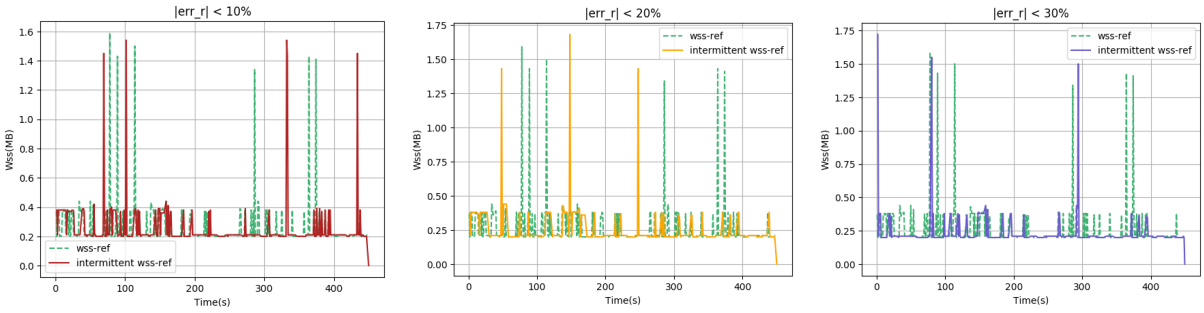
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.394: 641.leela_s-ref-1[WSS, interval=1s, user level]



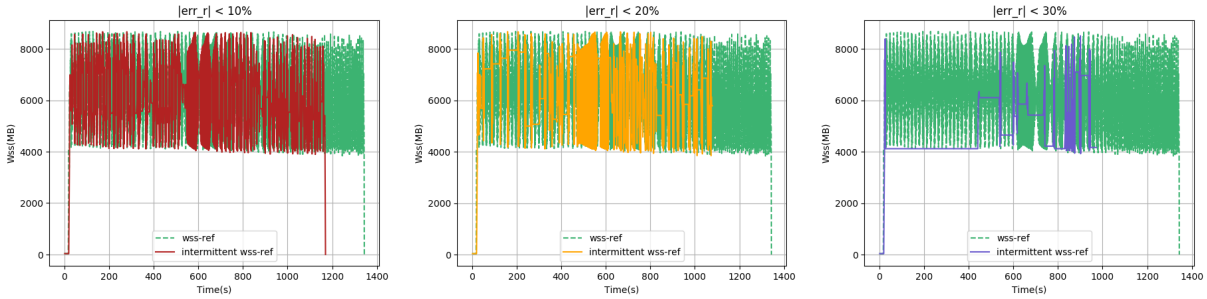
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.395: 644.nab_s-ref-1[WSS, interval=1s, user level]



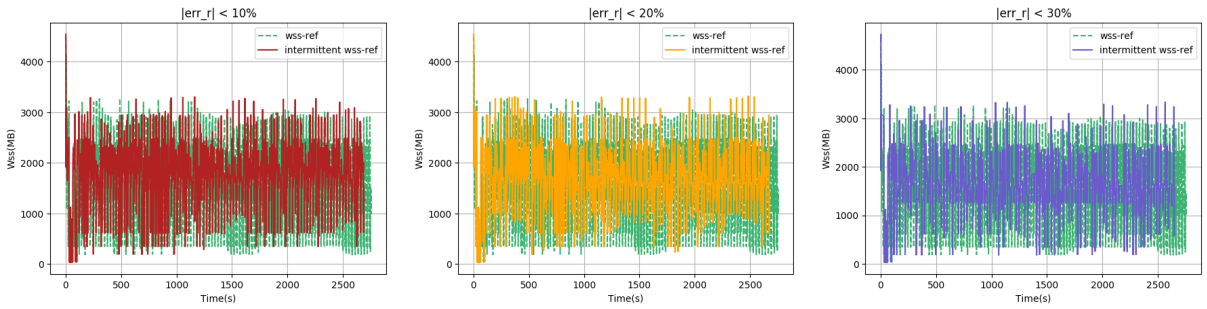
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.396: 648.exchange2_s-ref-1[WSS, interval=1s, user level]



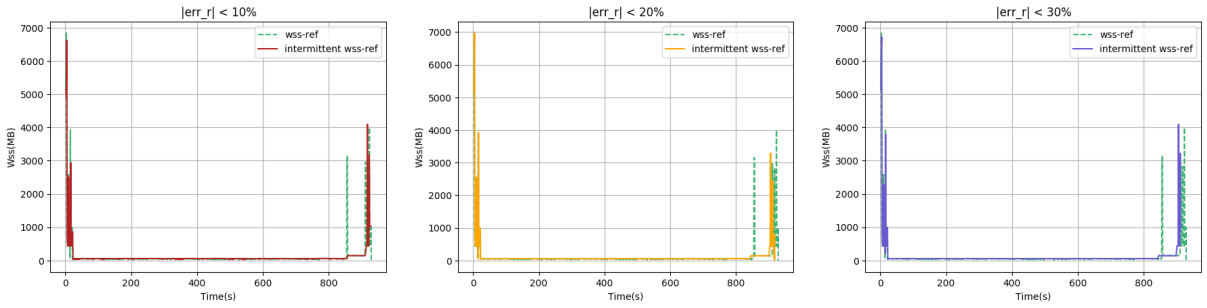
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.397: 649.fotonik3d_s-ref-1[WSS, interval=1s, user level]



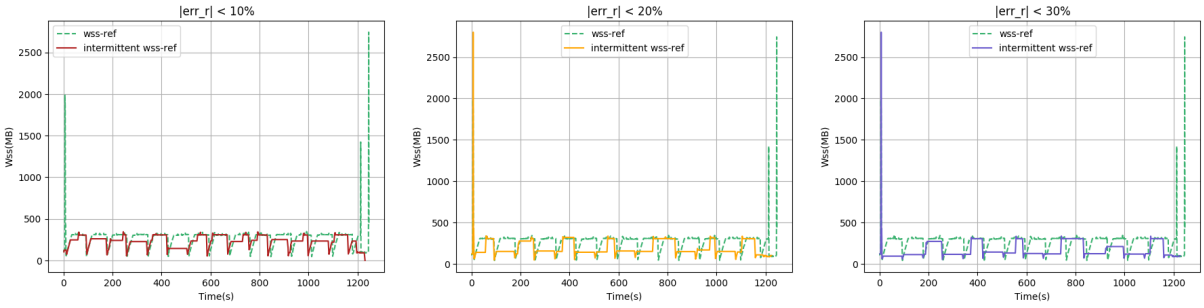
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.398: 654.roms_s-ref-1[WSS, interval=1s, user level]



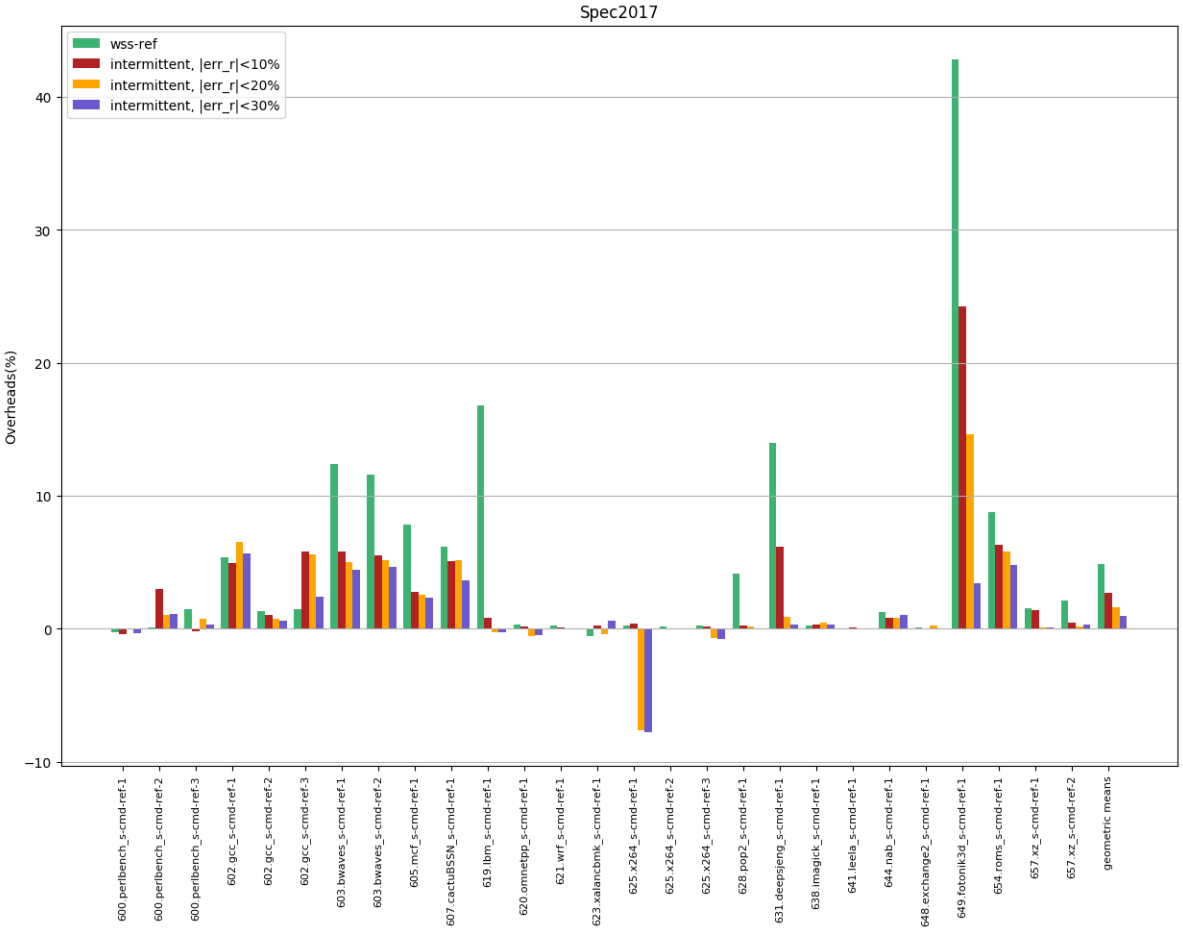
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.399: 657.xz_s-ref-1[WSS, interval=1s, user level]

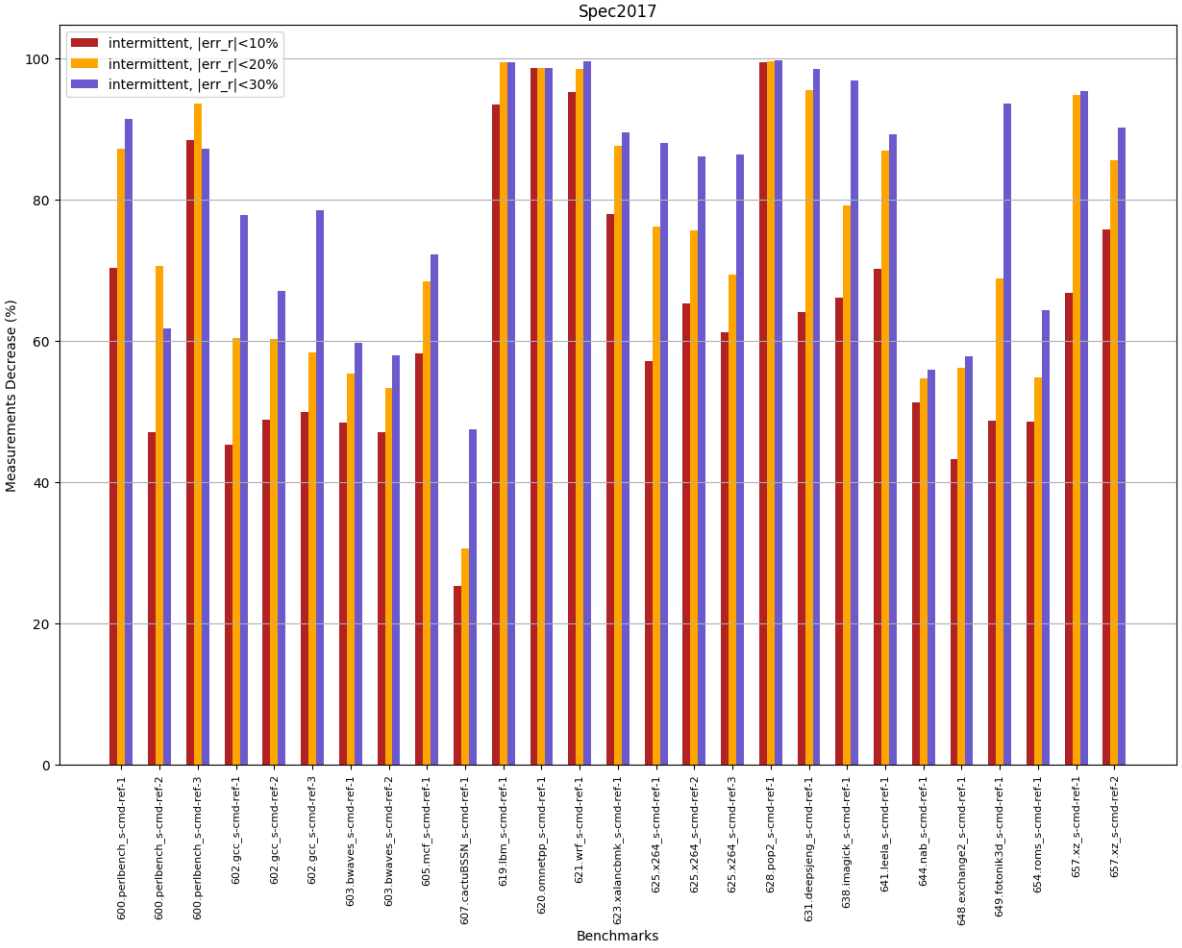


(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.400: 657.xz_s-ref-2[WSS, interval=1s, user level]



Σχήμα 5.401: 6xx benchmarks[Overheads, $err_r = \pm 10, 20, 30\%$, interval=1s, user level]



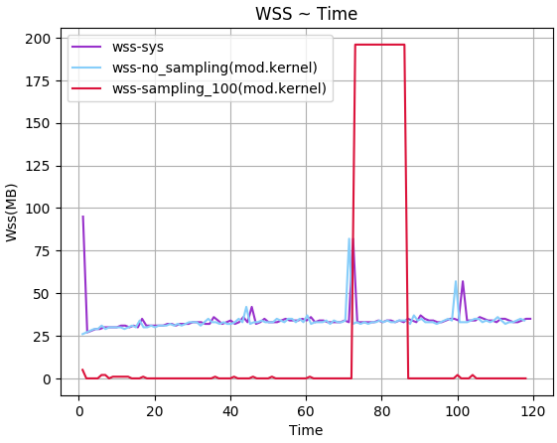
Σχήμα 5.402: 6xx benchmarks[Measurements Decrease, $err_r = \pm 10, 20, 30\%$, interval=1s, user level]

5.2 Μετρήσεις σε τροποποιημένο πυρήνα Linux

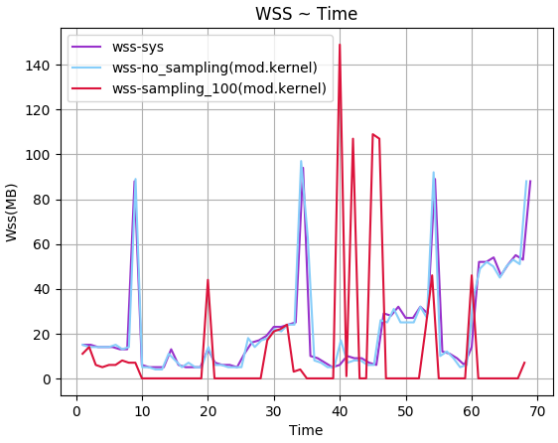
Τα παρακάτω πειράματα έγιναν κάνοντας χρήση των επεκτάσεων που προσθέσαμε στον πυρήνα 4.9.110 του Linux.

5.2.1 Sampling με σταθερό αριθμό σελίδων

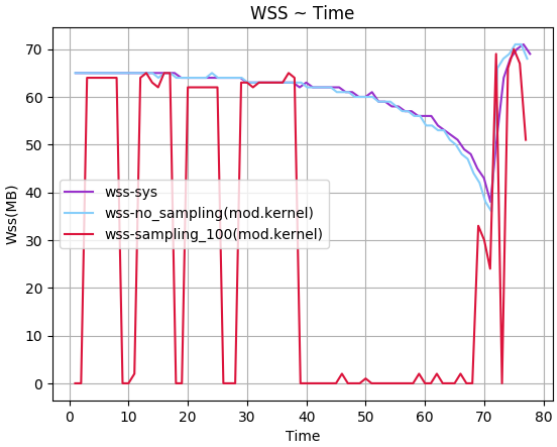
Στη συγκεκριμένη περίπτωση εφαρμόσαμε δειγματοληψία με σταθερό αριθμό δειγμάτων 100 στα πειράματά μας γράφοντας την τιμή 1 στο `/proc/[PID]/clear_idle` . Στα γραφήματα 5.403 - 5.427 απεικονίζονται οι εκτιμήσεις για το wss των εφαρμογών με χρήση της αναφερθείσας μεθόδου, ενώ στα 5.428 και 5.429 παρουσιάζονται αντίστοιχα τα overheads και το πλήθος μετρήσεων.



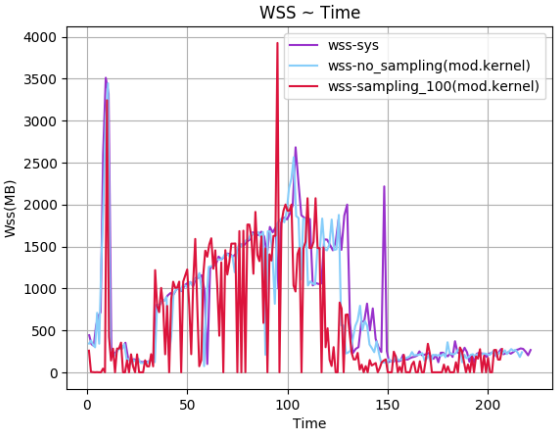
Σχήμα 5.403: 600.perlbench_s-ref-1[WSS, samples=100, interval=1s, kernel level]



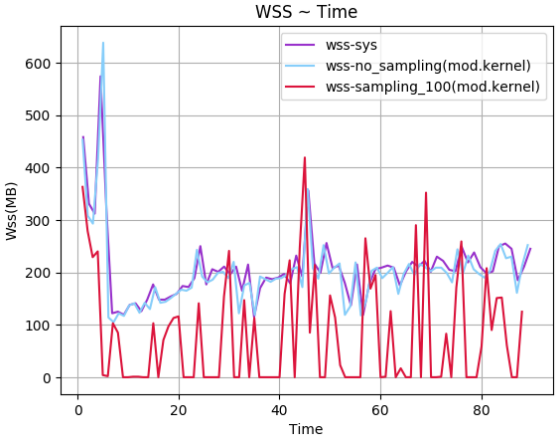
Σχήμα 5.404: 600.perlbench_s-ref-2[WSS, samples=100, interval=1s, kernel level]



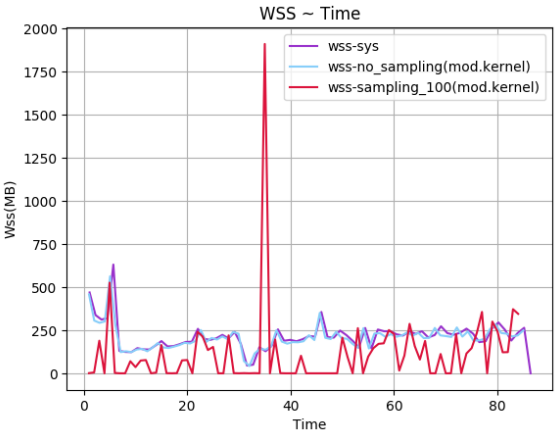
Σχήμα 5.405: 600.perlbench_s-ref-3[WSS, samples=100, interval=1s, kernel level]



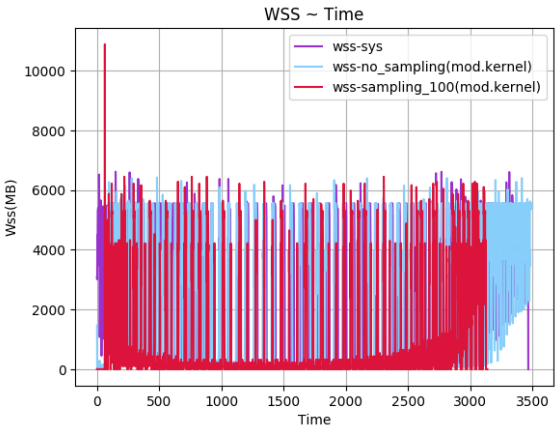
Σχήμα 5.406: 602.gcc_s-ref-1[WSS, samples=100, interval=1s, kernel level]



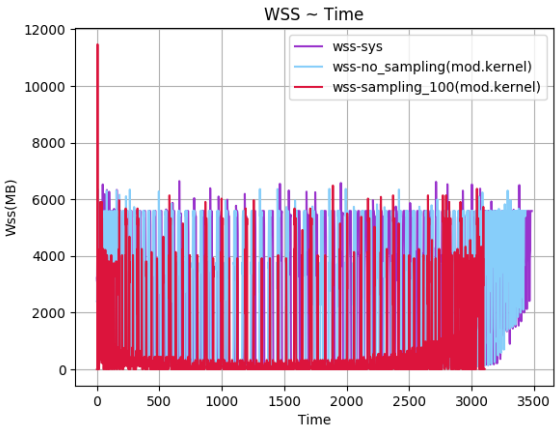
Σχήμα 5.407: 602.gcc_s-ref-2[WSS, samples=100, interval=1s, kernel level]



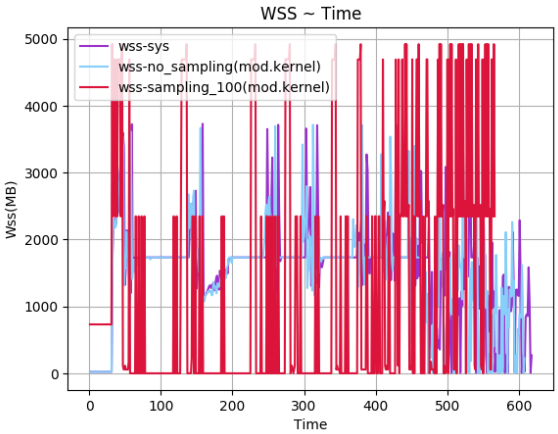
Σχήμα 5.408: 602.gcc_s-ref-3[WSS, samples=100, interval=1s, kernel level]



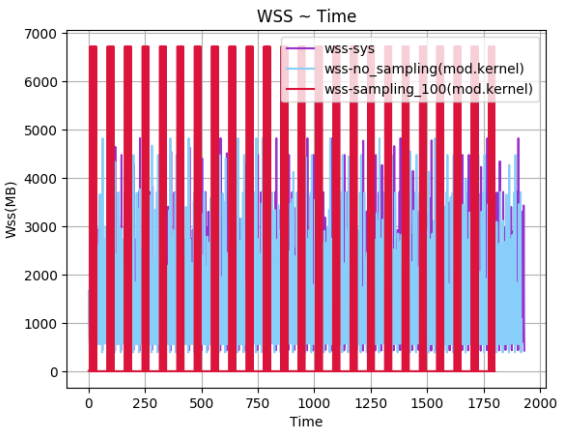
Σχήμα 5.409: 603.bwaves_s-ref-1[WSS, samples=100, interval=1s, kernel level]



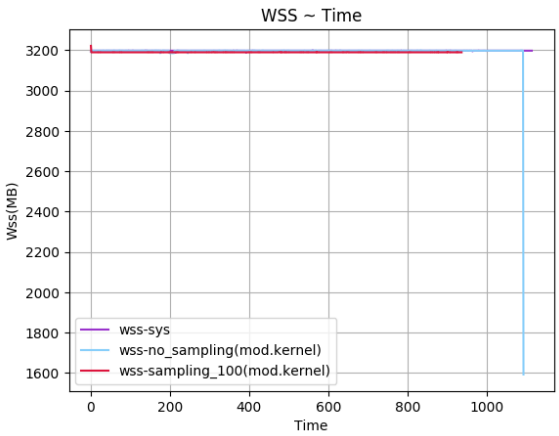
Σχήμα 5.410: 603.bwaves_s-ref-2[WSS, samples=100, interval=1s, kernel level]



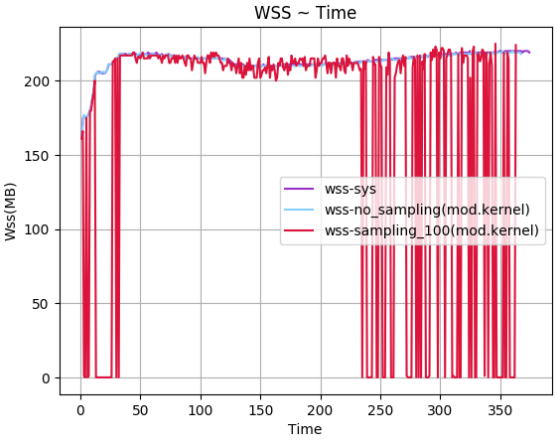
Σχήμα 5.411: 605.mcf_s-ref-1[WSS, samples=100, interval=1s, kernel level]



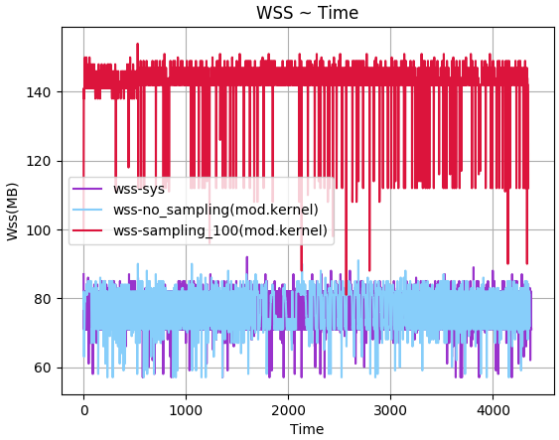
Σχήμα 5.412: 607.cactuBSSN_s-ref-1[WSS, samples=100, interval=1s, kernel level]



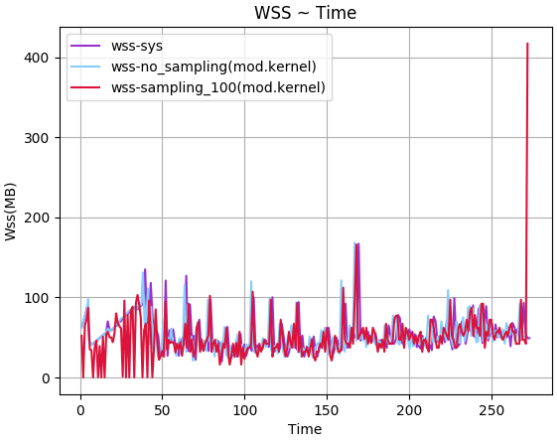
Σχήμα 5.413: 619.lbm_s-ref-1[WSS, samples=100, interval=1s, kernel level]



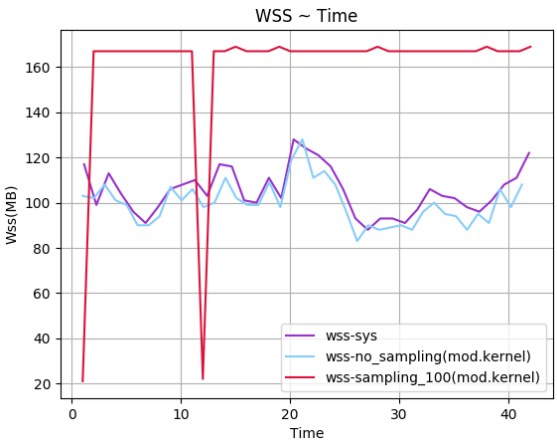
Σχήμα 5.414: 620.omnetpp_s-ref-1[WSS, samples=100, interval=1s, kernel level]



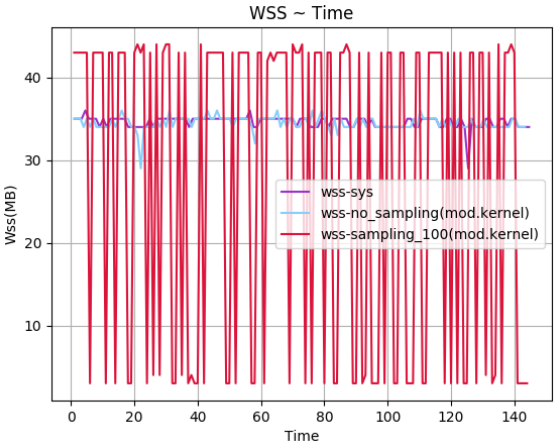
Σχήμα 5.415: 621.wrf_s-ref-1[WSS, samples=100, interval=1s, kernel level]



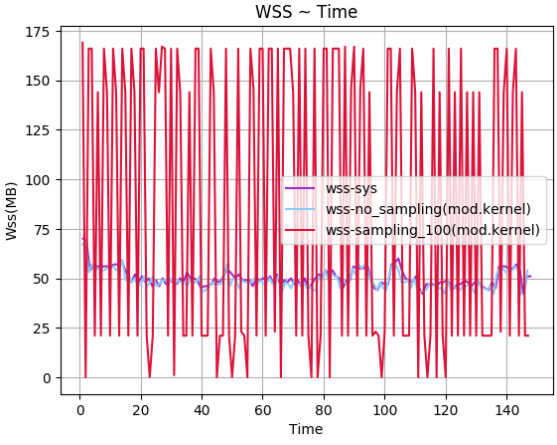
Σχήμα 5.416: 623.xalancbmk_s-ref-1[WSS, samples=100, interval=1s, kernel level]



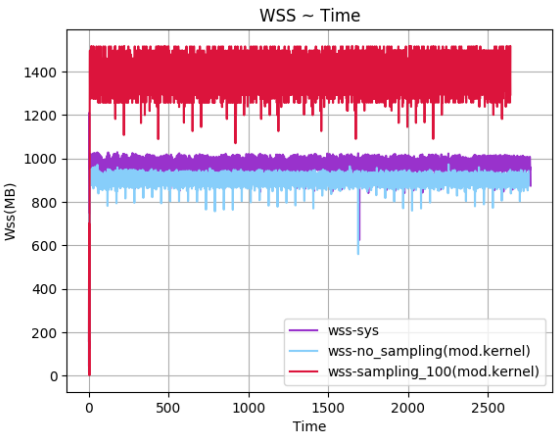
Σχήμα 5.417: 625.x264_s-ref-1[WSS, samples=100, interval=1s, kernel level]



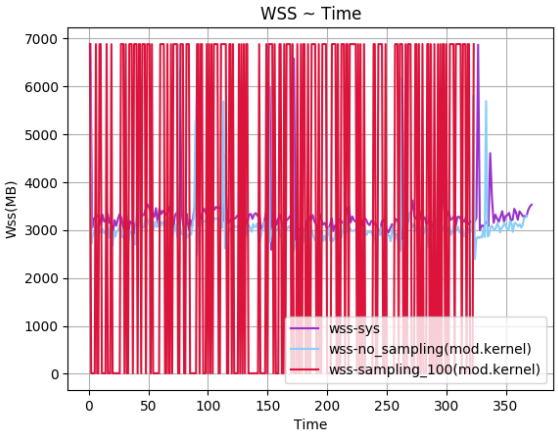
Σχήμα 5.418: 625.x264_s-ref-2[WSS, samples=100, interval=1s, kernel level]



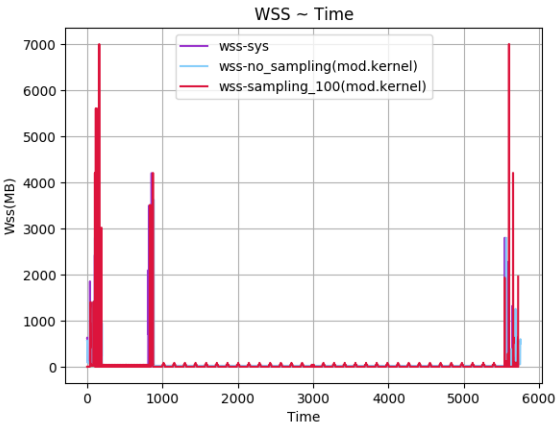
Σχήμα 5.419: 625.x264_s-ref-3[WSS, samples=100, interval=1s, kernel level]



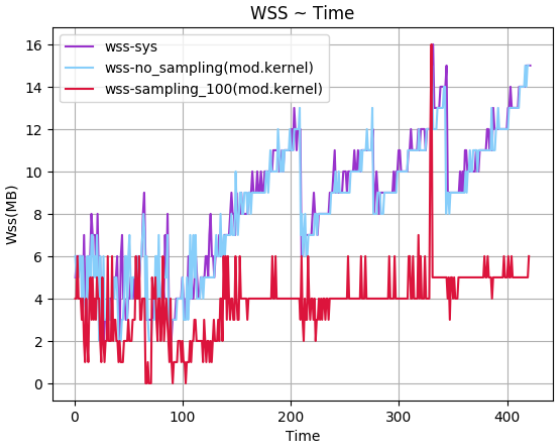
Σχήμα 5.420: 628.pop2_s-ref-1[WSS, samples=100, interval=1s, kernel level]



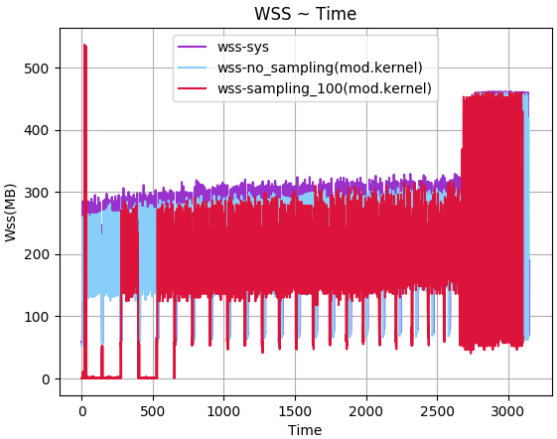
Σχήμα 5.421: 631.deepsjeng_s-ref-1[WSS, samples=100, interval=1s, kernel level]



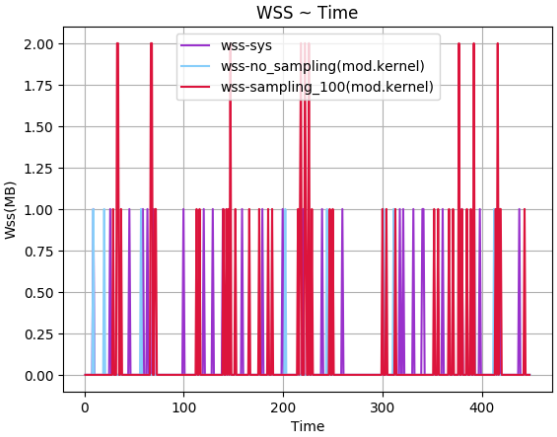
Σχήμα 5.422: 638.imagick_s-ref-1[WSS, samples=100, interval=1s, kernel level]



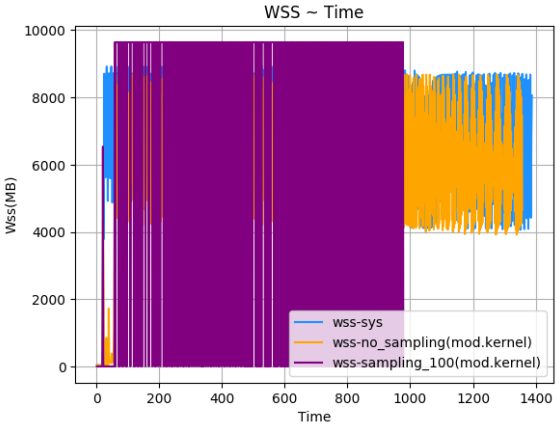
Σχήμα 5.423: 641.leela_s-ref-1[WSS, samples=100, interval=1s, kernel level]



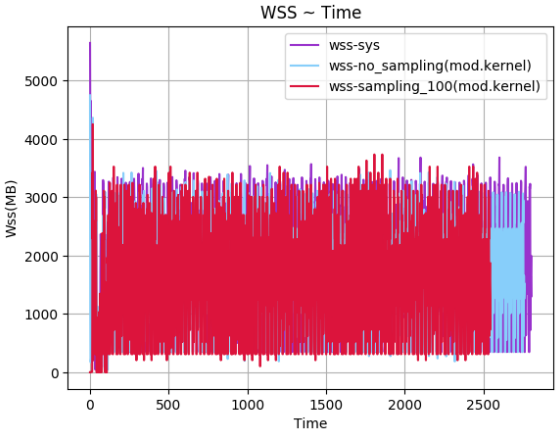
Σχήμα 5.424: 644.nab_s-ref-1[WSS, samples=100, interval=1s, kernel level]



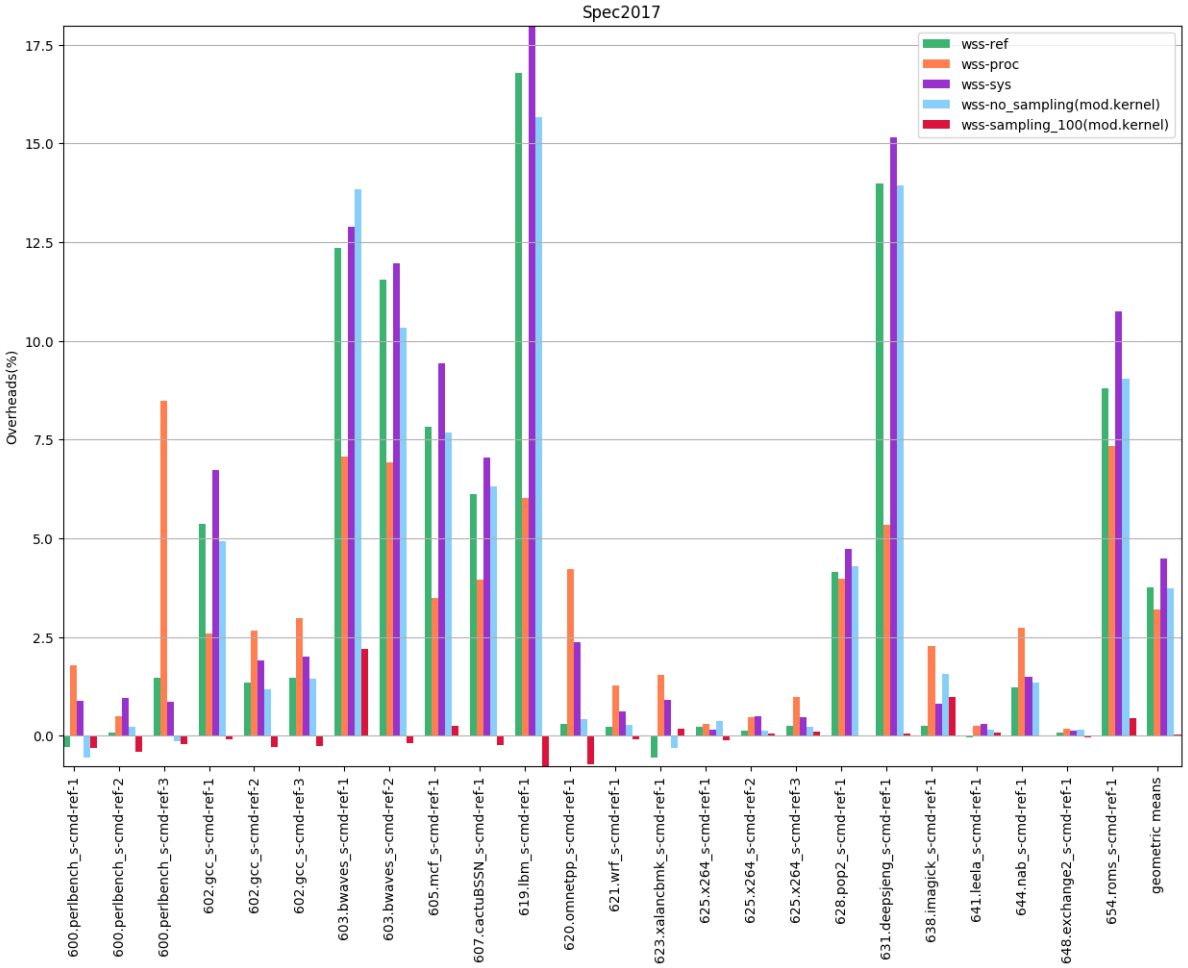
Σχήμα 5.425: 648.exchange2_s-ref-1[WSS, samples=100, interval=1s, kernel level]



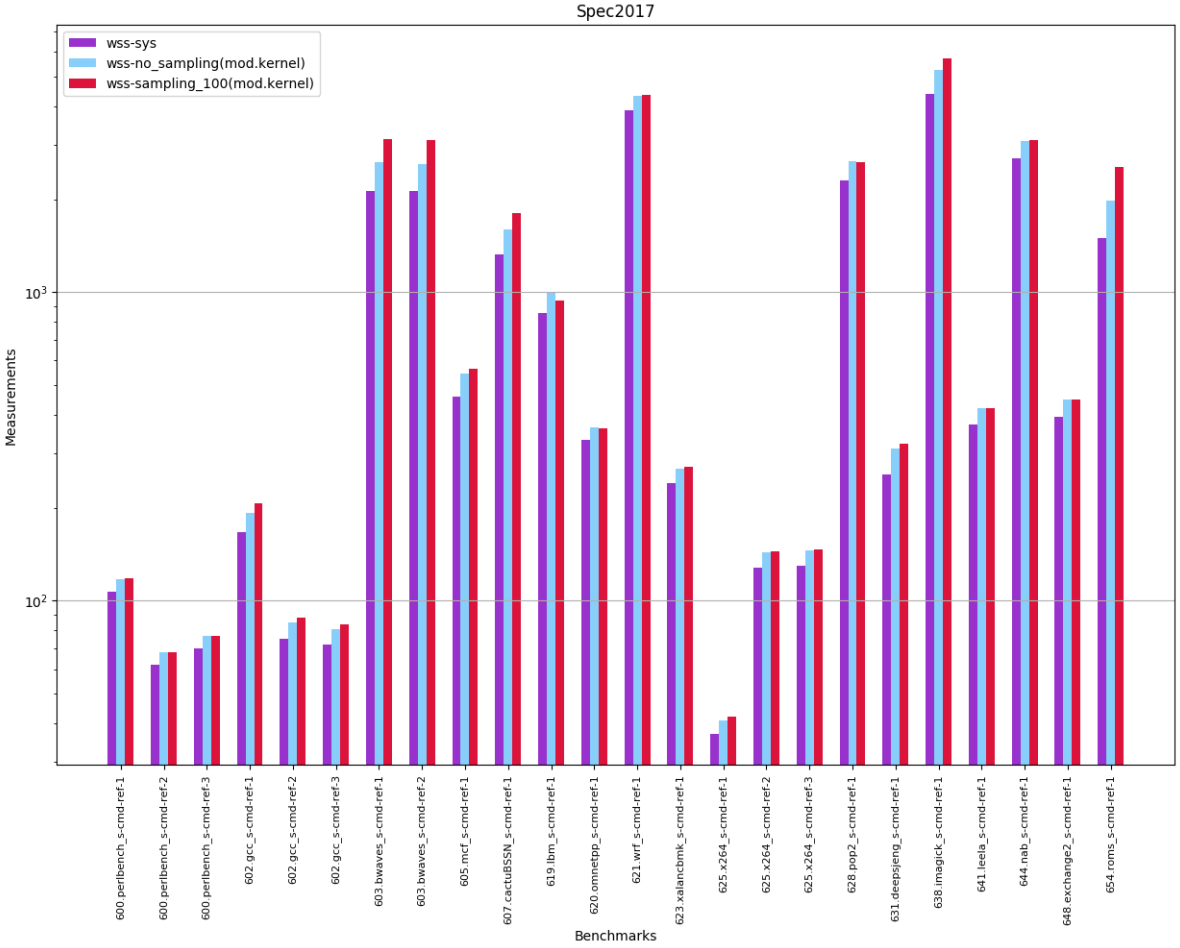
Σχήμα 5.426: 649.fotonik3d_s-ref-1[WSS, samples=100, interval=1s, kernel level]



Σχήμα 5.427: 654.roms_s-ref-1[WSS, samples=100, interval=1s, kernel level]



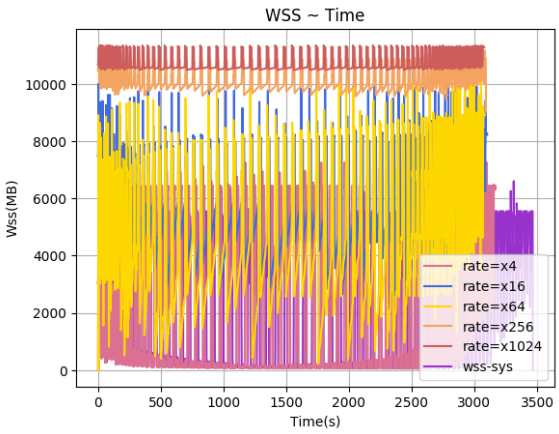
Σχήμα 5.428: 6xx benchmarks[Overheads, samples=100, interval=1s, kernel level]



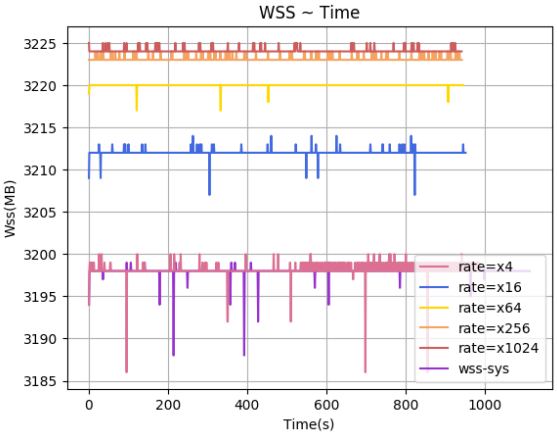
Σχήμα 5.429: 6xx benchmarks[Measurements, samples=100, interval=1s, kernel level]

5.2.2 Sampling με ρυθμό δειγματοληψίας

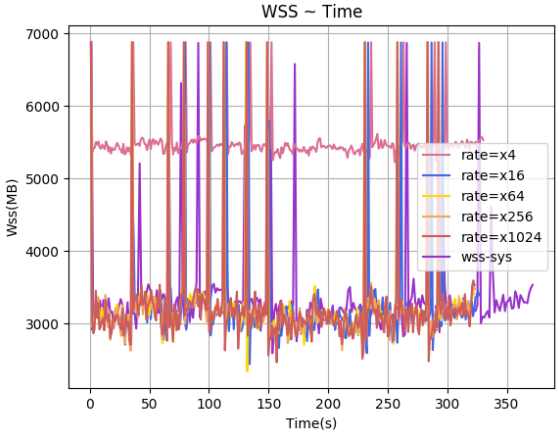
Αντικείμενο των συγκεκριμένων πειραμάτων ήταν η εφαρμογή δειγματοληψίας με ρυθμούς 4, 16, 64, 256 και 1024 γράφοντας τις αντίστοιχες τιμές στο `/proc/[PID] /clear_idle` . Στα γραφήματα 5.430 - 5.433 απεικονίζονται οι εκτιμήσεις για το wss των εφαρμογών 602.gcc_s, 603.bwaves_s, 605.mcf_s, 619.lbm_s, 623.xalancbmk_s, 628.pop2_s, 631.deepsjeng_s, και 649.fotonik3d_s με χρήση της αναφερθείσας μεθόδου, ενώ στο 5.434 παρουσιάζονται τα overheads τους.



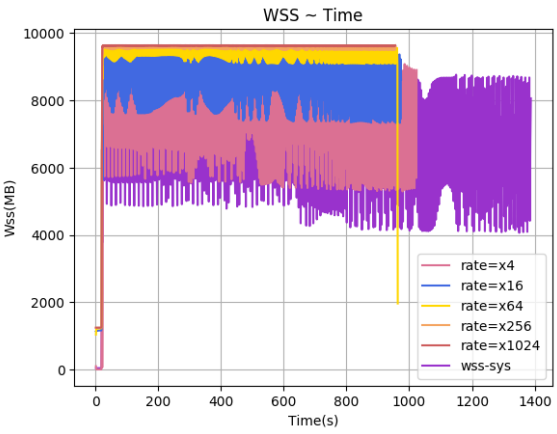
Σχήμα 5.430: 603.bwaves_s-ref-1[WSS, rate=4,16,64,256,1024, interval=1s, kernel level]



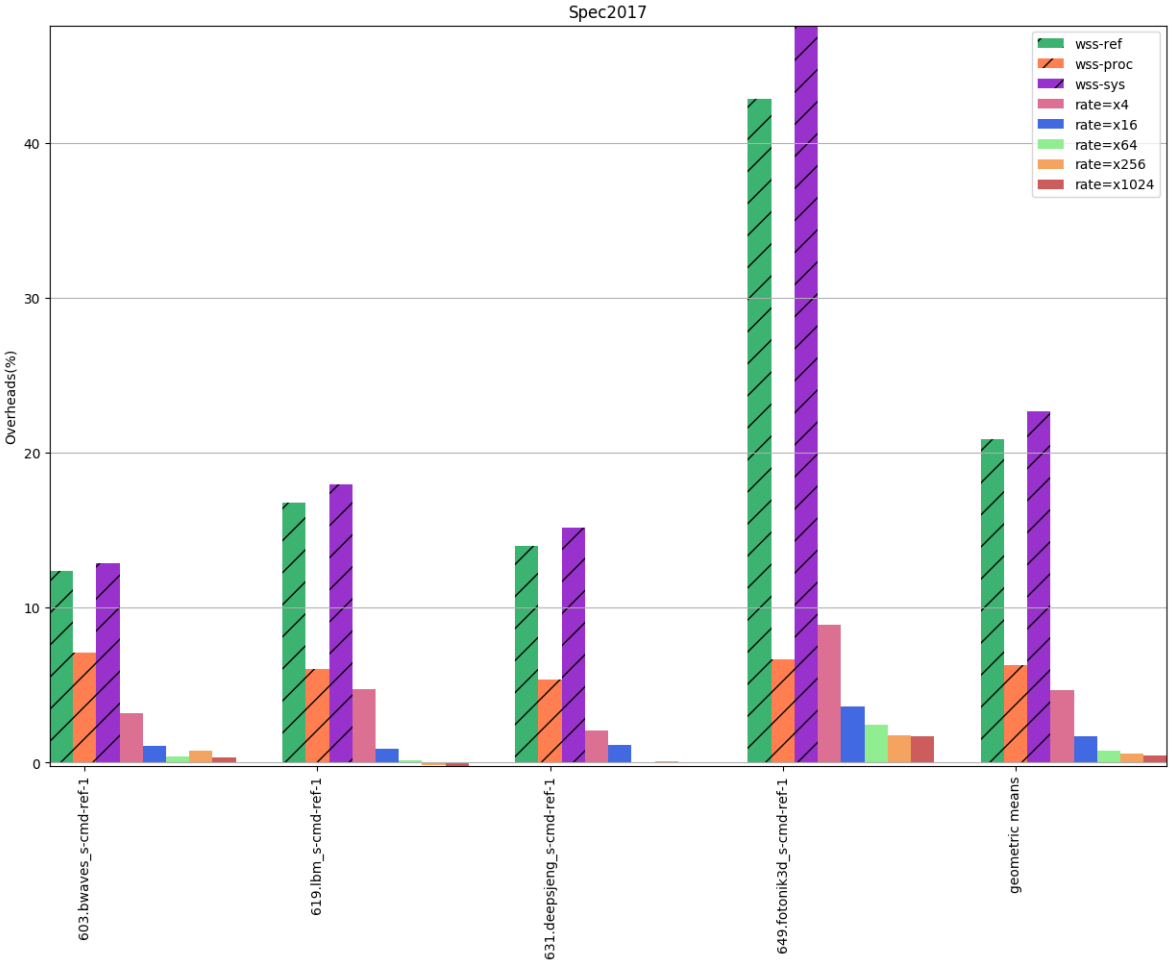
Σχήμα 5.431: 619.lbm_s-ref-1[WSS, rate=4,16,64,256,1024, interval=1s, kernel level]



Σχήμα 5.432: 631.deepsjeng_s-ref-1[WSS, rate=4,16,64,256,1024, interval=1s, kernel level]



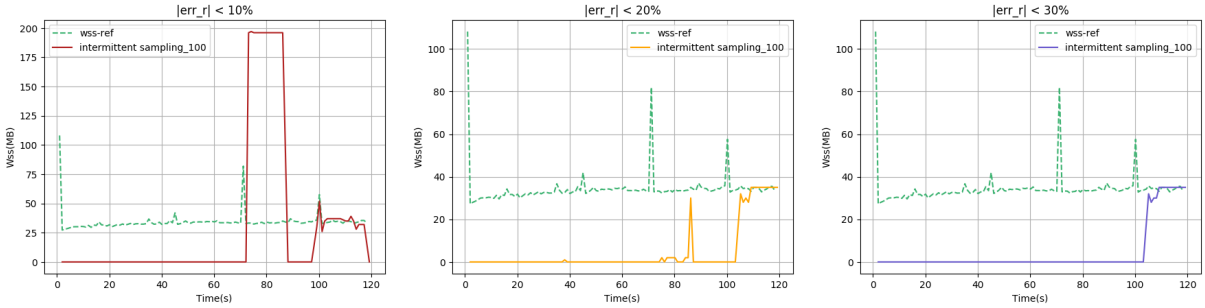
Σχήμα 5.433: 649.fotonik3d_s-ref-1[WSS, rate=4,16,64,256,1024, interval=1s, kernel level]



Σχήμα 5.434: 6xx benchmarks[Overheads, rate=4,16,64,256,1024, interval=1s, kernel level]

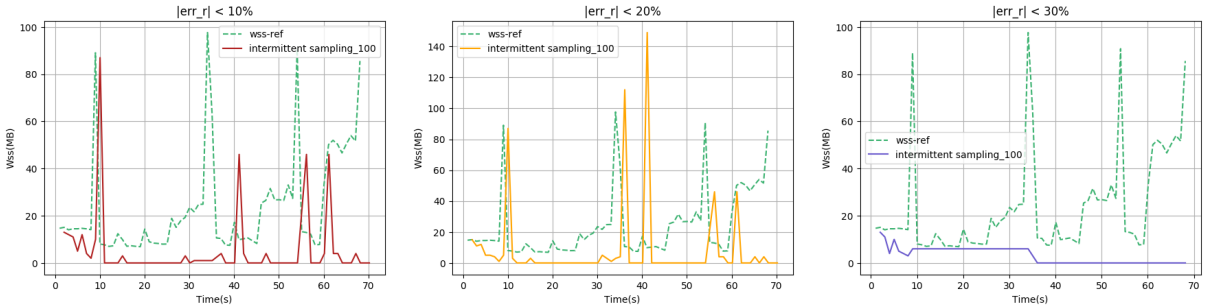
5.2.3 Intermittent Page Tracking & Sampling

Τα παρακάτω πειράματα έγιναν με την εφαρμογή της μεθόδου Intermittent Page Tracking χρησιμοποιώντας τη λειτουργικότητα του τροποποιημένου πυρήνα για δειγματοληψία με σταθερό αριθμό σελίδων. Στα γραφήματα 5.435 - 5.455 παρουσιάζονται οι εκτιμήσεις για το wss των εφαρμογών με χρήση της αναφερθείσας μεθόδου για τα διάφορα εύρη σφάλματος. Τέλος, στα 5.456 και 5.457 παρουσιάζονται αντίστοιχα τα overheads για τα διάφορα εύρη σφάλματος και τα ποσοστά μείωσης των μετρήσεων που λαμβάνουμε με καθένα από αυτά.



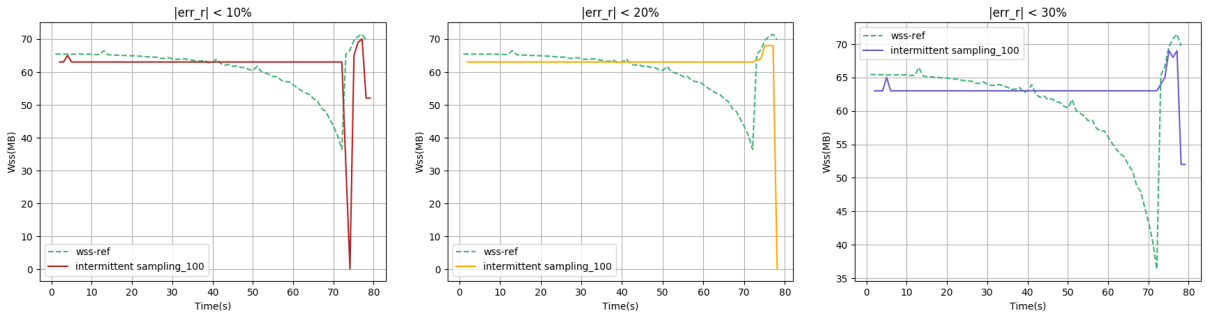
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.435: 600.perlbench_s-ref-1[WSS, samples=100, interval=1s, kernel level]



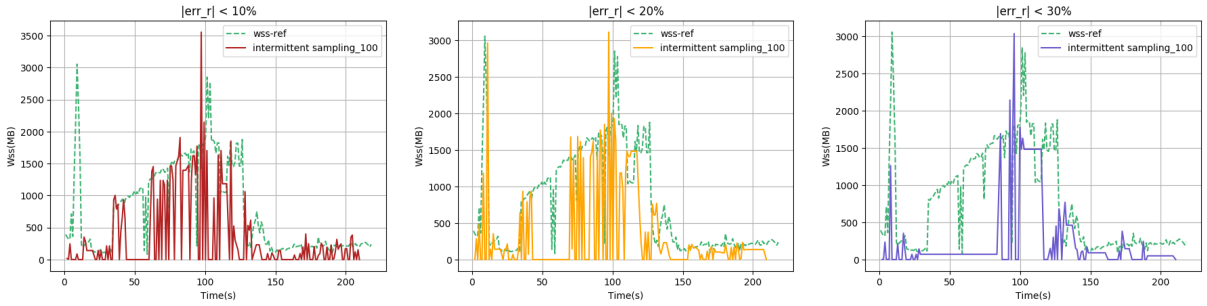
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.436: 600.perlbench_s-ref-2[WSS, samples=100, interval=1s, kernel level]



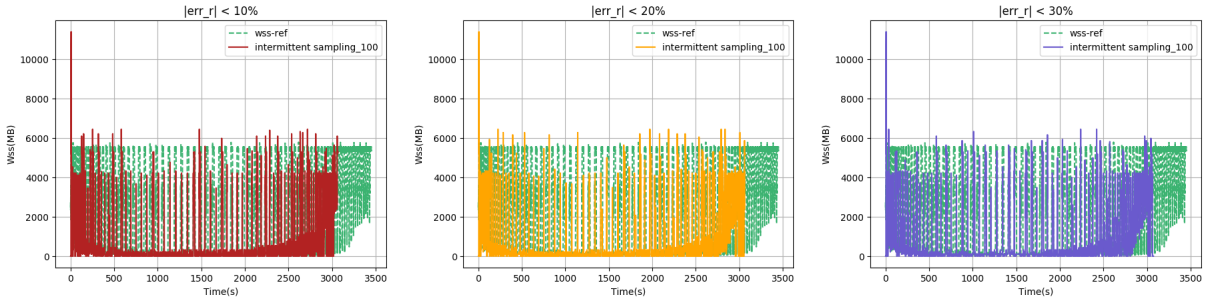
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.437: 600.perlbench_s-ref-3[WSS, samples=100, interval=1s, kernel level]



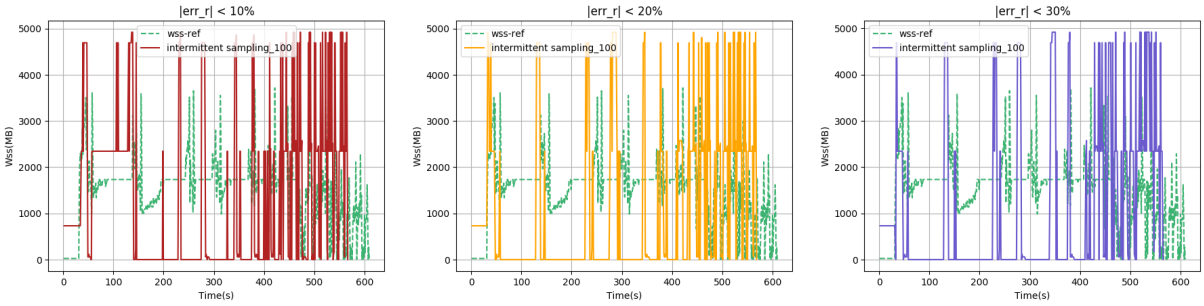
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.438: 602.gcc_s-ref-1[WSS, samples=100, interval=1s, kernel level]



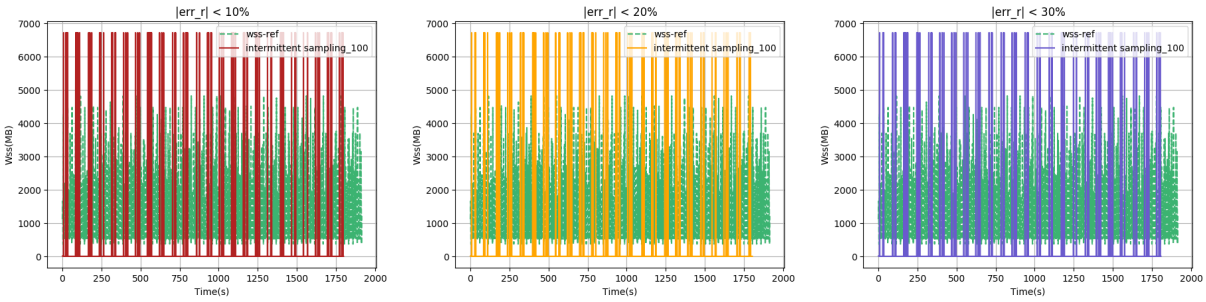
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.439: 603.bwaves_s-ref-1[WSS, samples=100, interval=1s, kernel level]



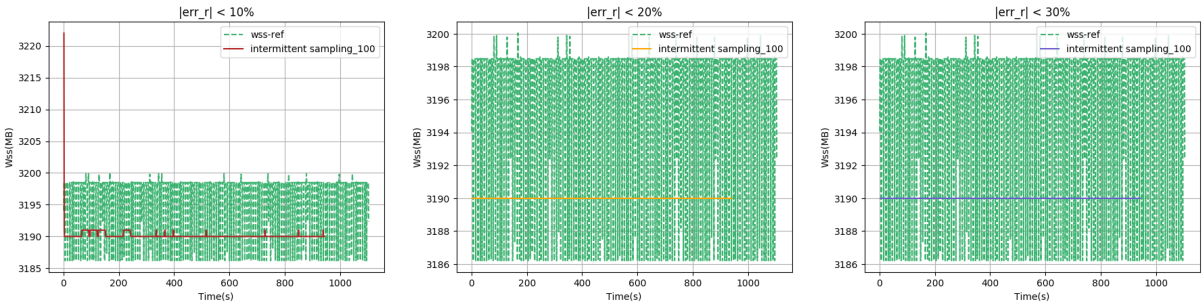
(α') $err_r = \pm 10\%$ (β') $err_r = \pm 20\%$ (γ') $err_r = \pm 30\%$

Σχήμα 5.440: 605.mcf_s-ref-1[WSS, samples=100, interval=1s, kernel level]



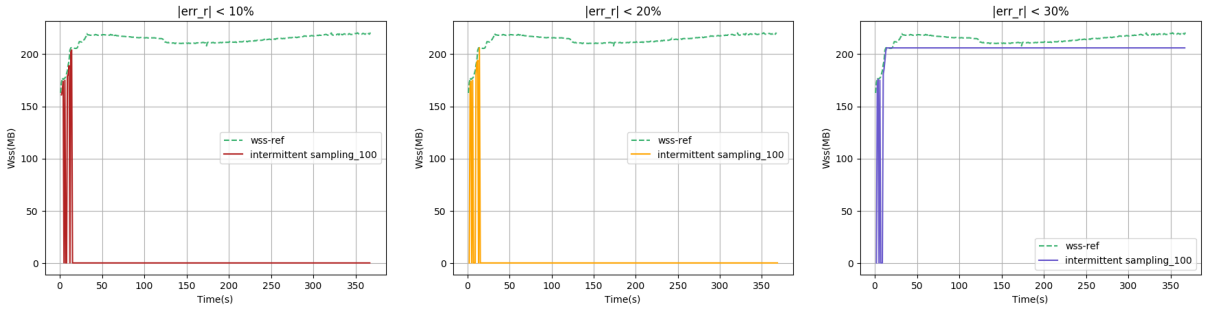
(α') $err_r = \pm 10\%$ (β') $err_r = \pm 20\%$ (γ') $err_r = \pm 30\%$

Σχήμα 5.441: 607.cactuBSSN_s-ref-1[WSS, samples=100, interval=1s, kernel level]



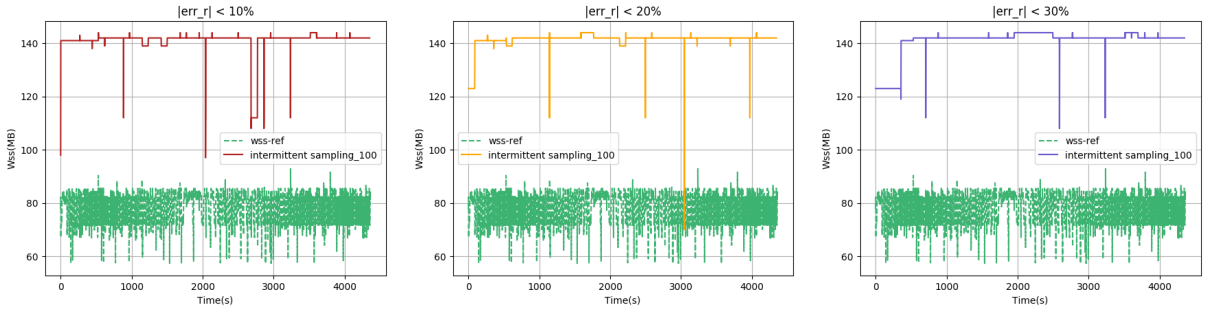
(α') $err_r = \pm 10\%$ (β') $err_r = \pm 20\%$ (γ') $err_r = \pm 30\%$

Σχήμα 5.442: 619.lbm_s-ref-1[WSS, samples=100, interval=1s, kernel level]



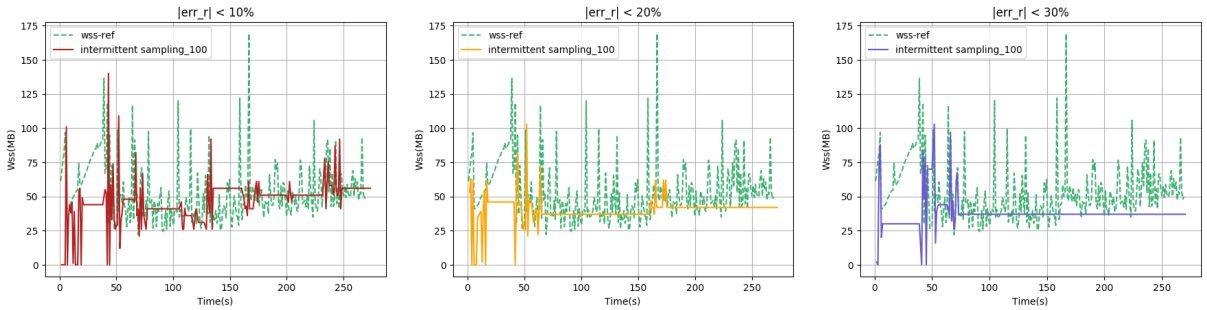
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.443: 620.omnetpp_s-ref-1[WSS, samples=100, interval=1s, kernel level]



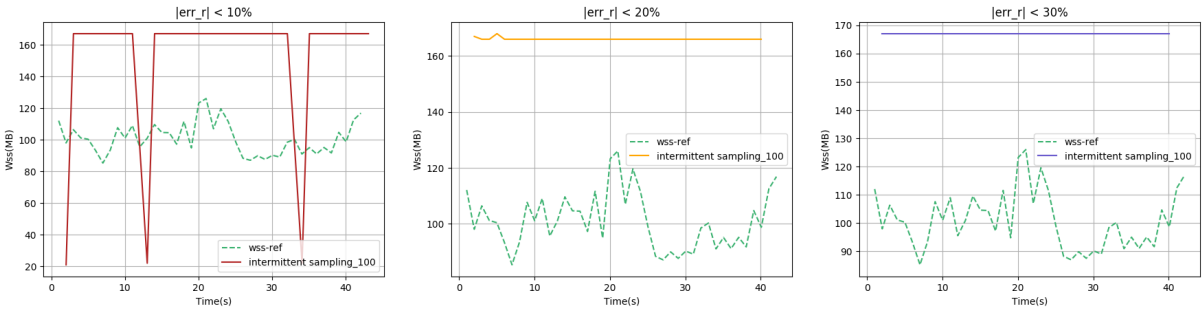
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.444: 621.wrf_s-ref-1[WSS, samples=100, interval=1s, kernel level]



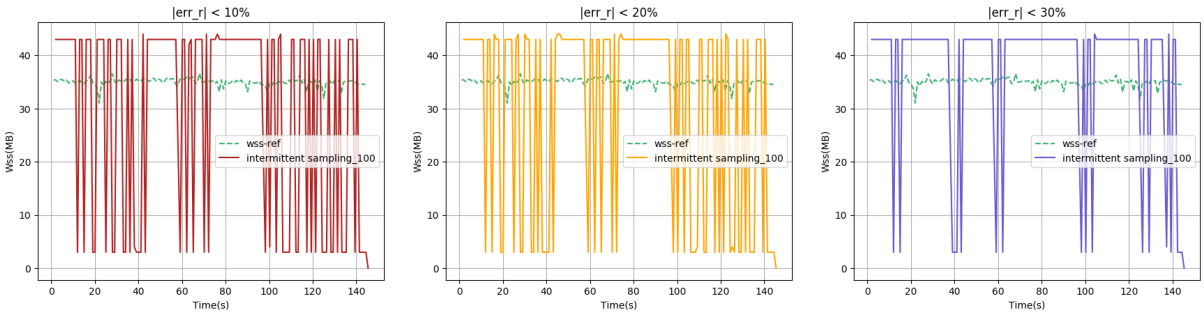
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.445: 623.xalancbmk_s-ref-1[WSS, samples=100, interval=1s, kernel level]



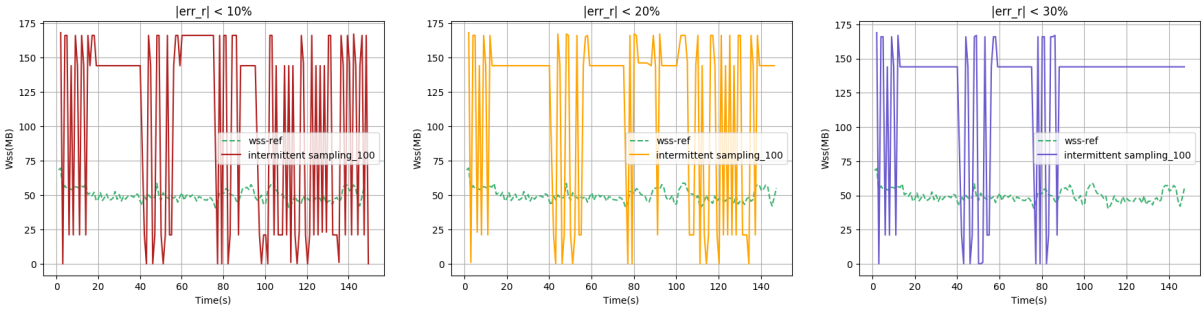
(α') $err_r = \pm 10\%$ (β') $err_r = \pm 20\%$ (γ') $err_r = \pm 30\%$

Σχήμα 5.446: 625.x264_s-ref-1[WSS, samples=100, interval=1s, kernel level]



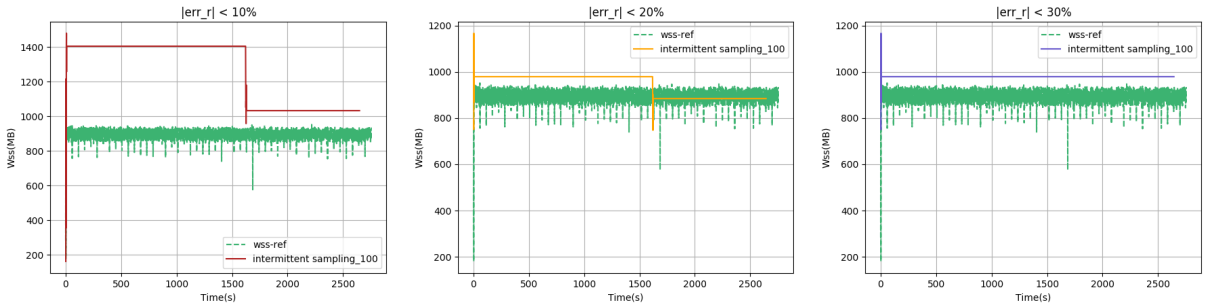
(α') $err_r = \pm 10\%$ (β') $err_r = \pm 20\%$ (γ') $err_r = \pm 30\%$

Σχήμα 5.447: 625.x264_s-ref-2[WSS, samples=100, interval=1s, kernel level]



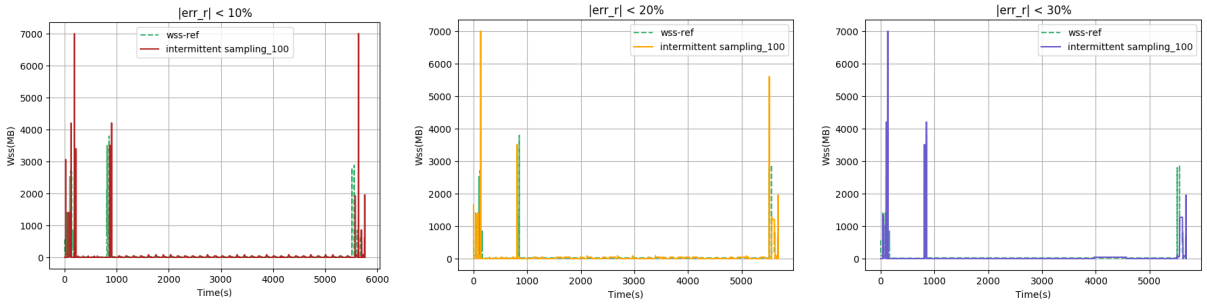
(α') $err_r = \pm 10\%$ (β') $err_r = \pm 20\%$ (γ') $err_r = \pm 30\%$

Σχήμα 5.448: 625.x264_s-ref-3[WSS, samples=100, interval=1s, kernel level]



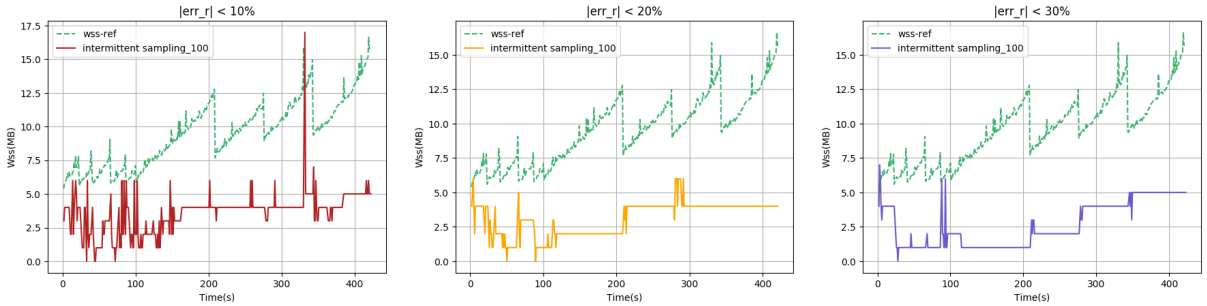
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.449: 628.pop2_s-ref-1[WSS, samples=100, interval=1s, kernel level]



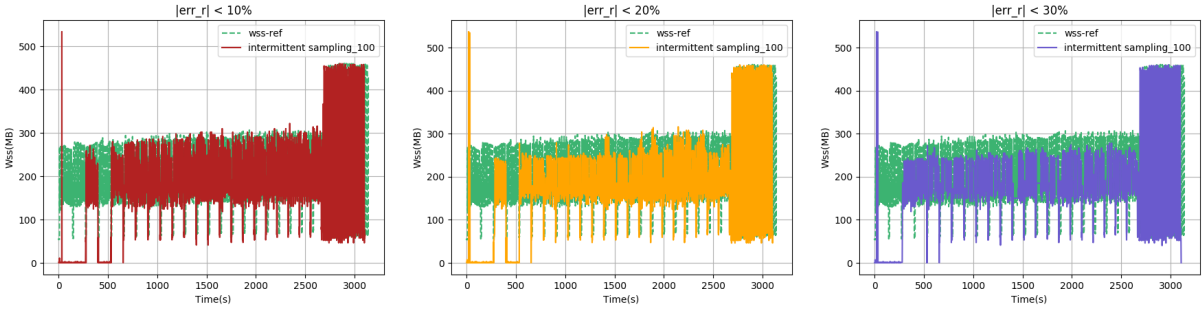
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.450: 638.imagick_s-ref-1[WSS, samples=100, interval=1s, kernel level]



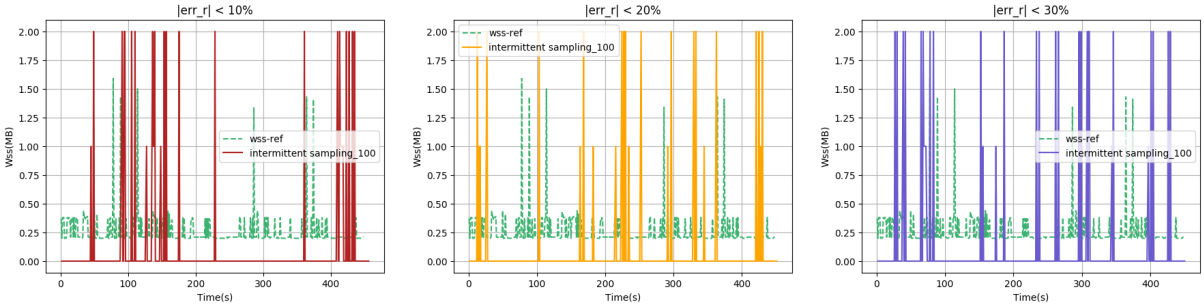
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.451: 641.leela_s-ref-1[WSS, samples=100, interval=1s, kernel level]



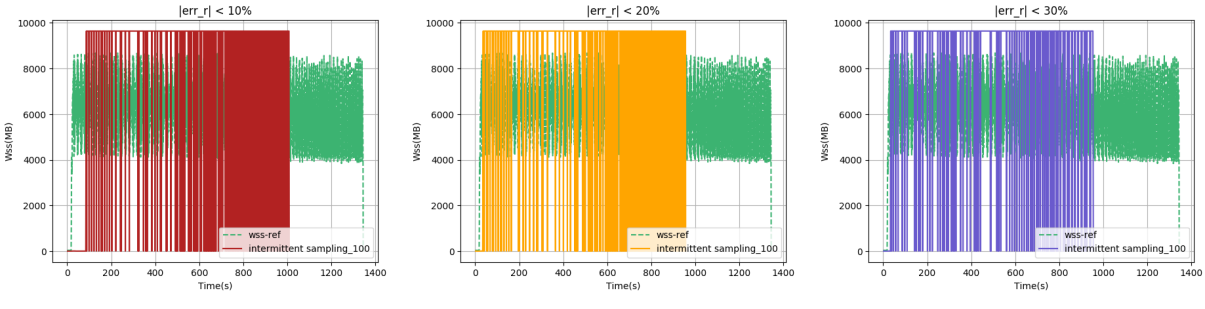
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.452: 644.nab_s-ref-1[WSS, samples=100, interval=1s, kernel level]



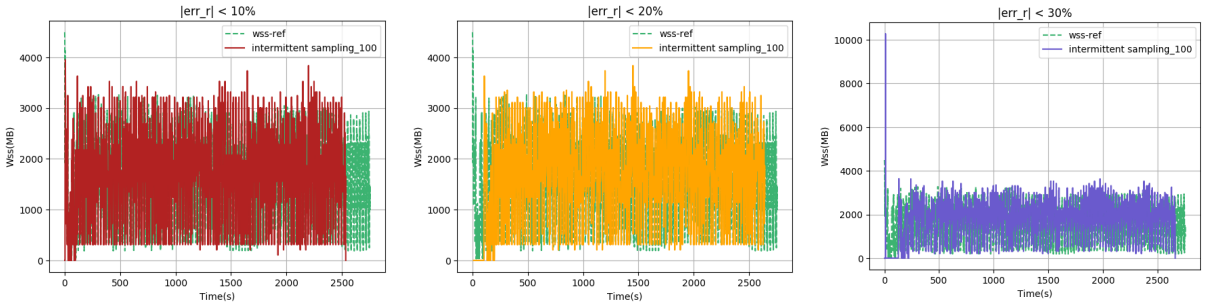
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.453: 648.exchange2_s-ref-1[WSS, samples=100, interval=1s, kernel level]



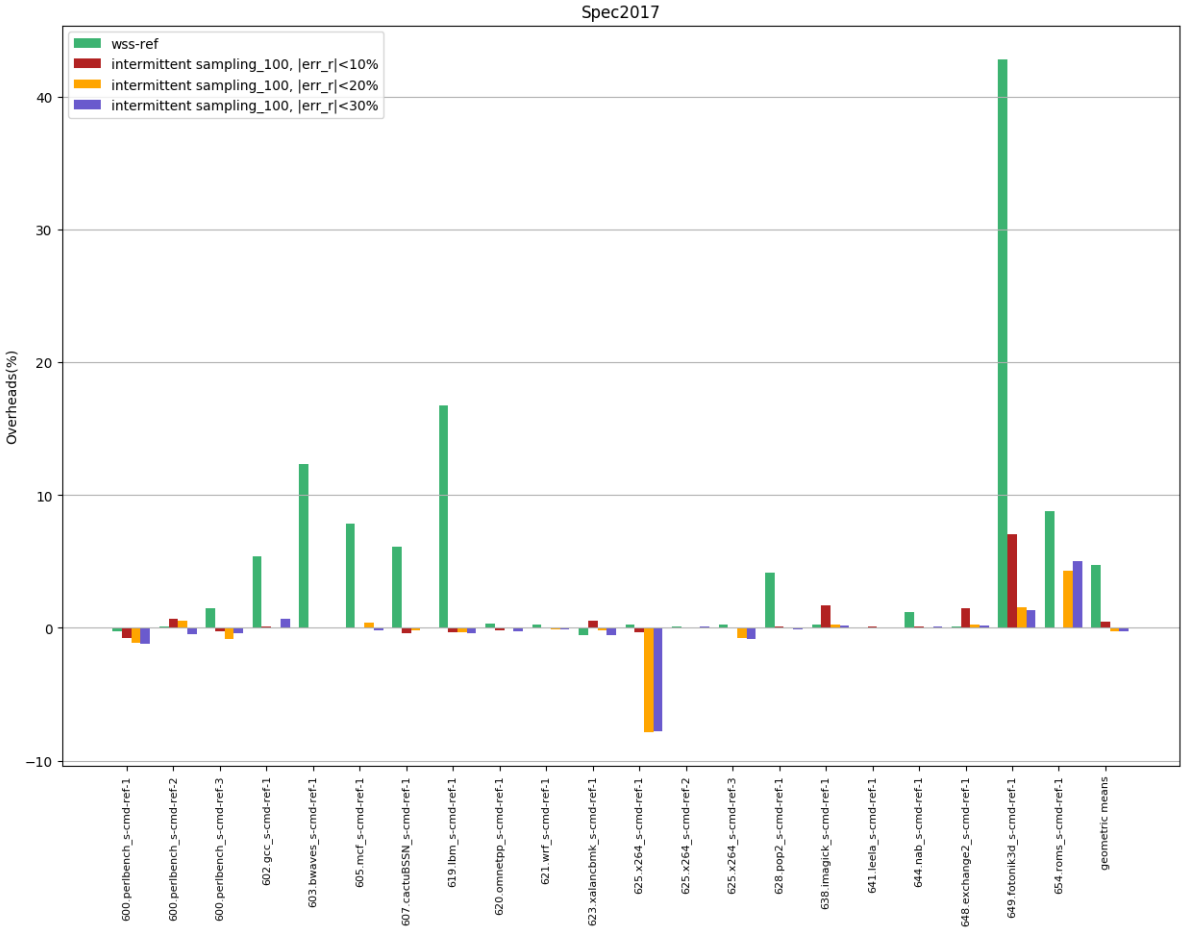
(α′) $err_r = \pm 10\%$ (β′) $err_r = \pm 20\%$ (γ′) $err_r = \pm 30\%$

Σχήμα 5.454: 649.fotonik3d_s-ref-1[WSS, samples=100, interval=1s, kernel level]

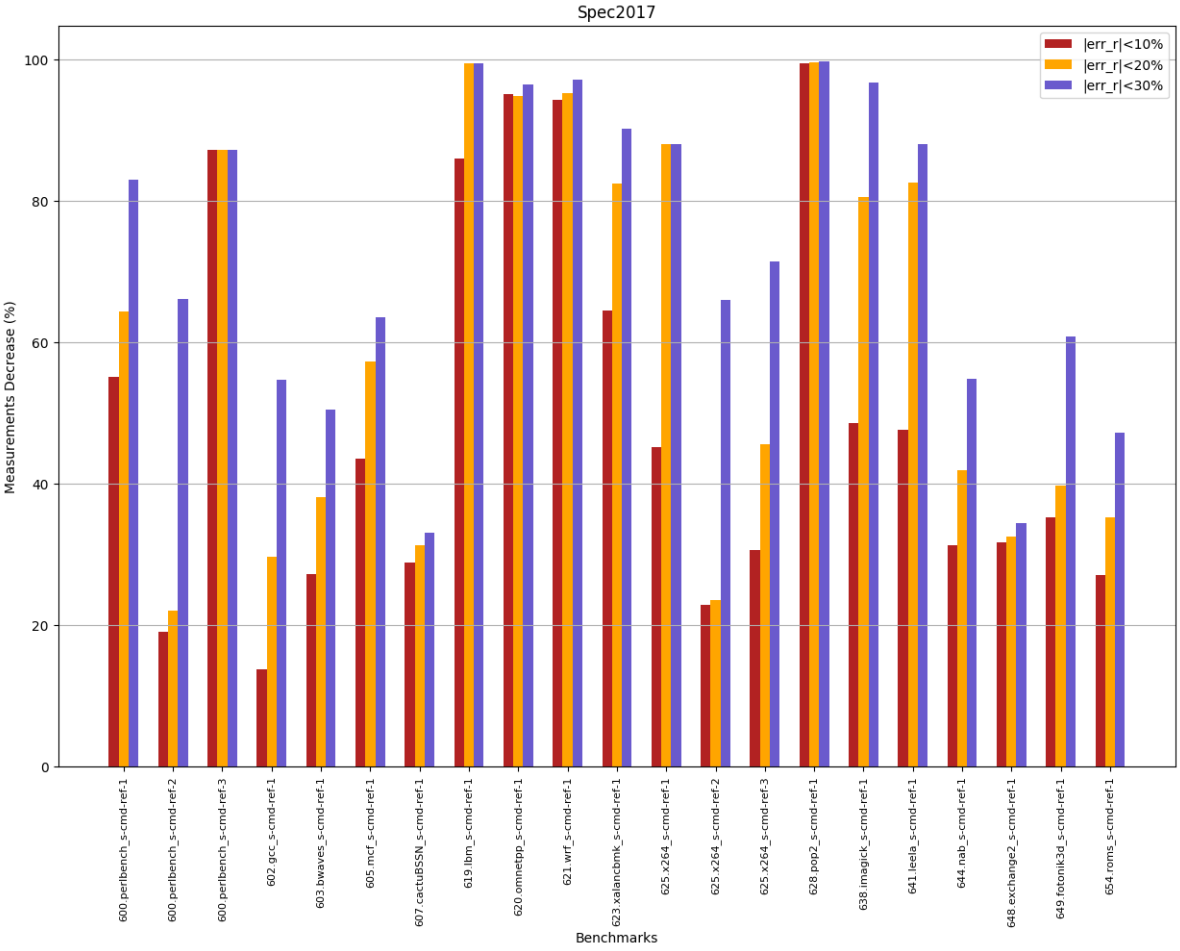


(α') $err_r = \pm 10\%$ (β') $err_r = \pm 20\%$ (γ') $err_r = \pm 30\%$

Σχήμα 5.455: 654.roms_s-ref-1[WSS, samples=100, interval=1s, kernel level]



Σχήμα 5.456: 6xx benchmarks[Overheads, $err_r = \pm 10, 20, 30\%$, samples=100, interval=1s, kernel level]



Σχήμα 5.457: 6xx benchmarks[Measurements Decrease, $err_r = \pm 10, 20, 30\%$, samples=100, interval=1s, kernel level]

Κεφάλαιο 6

Αξιολόγηση Μετρήσεων

Στο κεφάλαιο αυτό θα αξιολογήσουμε τα αποτελέσματα των μετρήσεων που παρουσιάσαμε στο Κεφάλαιο 5.

6.1 Αξιολόγηση πειραμάτων σε μη τροποποιημένο πυρήνα Linux

6.1.1 Idle & Referenced Page Tracking

Αρχικά θα συγκρίνουμε τα διάφορα εργαλεία μέτρησης working set για διάστημα παρακολούθησης το 1s ως προς την μετρική της ακρίβειας. Όπως παρατηρούμε στο σύνολο των γραφημάτων 5.1 - 5.62 οι μετρήσεις των monitoring tools wss-ref, wss-proc και wss-sys έχουν περίπου την ίδια εξέλιξη στο χρόνο. Ωστόσο, αξίζει να αναφερθεί πως σε ορισμένα benchmarks που παρουσιάζουν μεγαλύτερο working set το wss-proc φαίνεται να λαμβάνει μεγαλύτερες μετρήσεις για το WSS. Αυτό μπορεί να δικαιολογηθεί από το ότι καταναλώνει πολύ χρόνο στο να ελέγξει και να διαβάσει κάθε bit του bitmap από αυτά που αντιστοιχούν στη διεργασία, λόγω των απαιτούμενων syscalls, με αποτέλεσμα στο διάστημα αυτό η εφαρμογή να αγγίζει περισσότερες σελίδες και το εργαλείο να υπερεκτιμά το WSS τη συγκεκριμένη χρονική στιγμή.

Όσον αφορά το πλήθος μετρήσεων που λαμβάνουν τα εργαλεία, παρατηρώντας τα γραφήματα 5.65 και 5.66 προκύπτει πως τα wss-ref και wss-sys παρουσιάζουν μεγαλύτερες τιμές από το wss-proc. Αυτό είναι αναμενόμενο, λόγω του χρόνου που καταναλώνει στα συνεχόμενα syscalls το τελευταίο, όπως αναφέραμε παραπάνω. Το συγκεκριμένο πρόβλημα είναι διακριτό και στις γραφικές παραστάσεις των WSS, στις οποίες φαίνεται πως το wss-proc έχει λάβει λιγότερα σημεία στο χρόνο, με αποτέλεσμα να παρουσιάζει κάποιο βαθμό ανακρίβειας σε σχέση με τις μετρήσεις των άλλων εργαλείων.

Τέλος, μελετώντας τα γραφήματα 5.63 και 5.64 παρατηρούμε πως υπάρχουν αρκετές διαφορές ως προς το overheads που εισάγουν τα εργαλεία μέτρησης WSS στις δύο ομάδες των benchmarks. Ξεκινώντας από τα 5xx βλέπουμε πως παρουσιάζουν καθυστερήσεις που κυμαίνονται από αμελητέα ποσοστά μέχρι της τάξης των 10%. Μεγαλύτερη επιβάρυνση χρόνου φαίνεται να προκαλεί το εργαλείο wss-proc ειδικά σε εφαρμογές που αγγίζουν πολλές σελίδες μνήμης όπως είναι οι 503.bwaves_r, 505.mcf_r, 507.cactusBSSN_r, 519.lbm_r, 527.cam4_r, 531.deepsjeng_r, 549.fotoniki3d_r και 554.roms_r με τις καθυστερήσεις τους να είναι μεταξύ του 4% και 10% ως προς το αρχικό χρόνο εκτέλεσής τους. Αυτό δικαιολογείται από το ότι το συγκεκριμένο εργαλείο καταναλώνει επεξεργαστικό χρόνο για να κάνει read και write το εκάστοτε bit του bitmap που αντιστοιχεί σε καθεμία από τις σελίδες μνήμης που αγγίζει η εφαρμογή, σε αντίθεση με τις άλλα εργαλεία που αποφεύγουν τις επαναλαμβανόμενες syscalls τόσο στη διαδικασία του reset όσο και σε αυτή του διαβάσματος των flags. Το μικρότερο κόστος επιβάρυνσης φαίνεται να το έχει το εργαλείο wss-ref που διαβάζει τα referenced flags με τα overheads που εισάγει να μην ξεπερνάνε το 2% κατά γενική περίπτωση και το 4% στην περίπτωση των απαιτητικών ως προς το WSS εφαρμογών. Τέλος, αξίζει να αναφέρουμε πως το εργαλείο wss-sys εισάγει επιβαρύνσεις ίδιας κλίμακας με του wss-ref με τη διαφορά ότι στις εφαρμογές με μεγάλο working set αυτές αγγίζουν το 5%.

Συγκρίνοντας τα overheads των 5xx benchmarks με αυτά των 6xx προκύπτει πως τα τελευταία είναι μεγαλύτερης κλίμακας ειδικά στις προαναφερθείσες απαιτητικές εφαρμογές, που ξεπερνάνε το 10% και αγγίζουν μέχρι και το 50% στην περίπτωση του 549.fotoniki3d_r. Ο λόγος στον οποίο οφείλεται αυτή η διαφορά είναι το μέγεθος του working set των συγκεκριμένων εφαρμογών που είναι πολύ μεγαλύτερο και αγγίζει τα 10Gb σε ορισμένες από αυτές. Σε αντίθεση με την ομάδα των 5xx, σε αυτή των 6xx μικρότερο κόστος επιβάρυνσης στις απαιτητικές εφαρμογές φαίνεται να παρουσιάζει το εργαλείο wss-proc, γεγονός που αποδίδεται στο ότι δεν προλαβαίνει να λάβει πολλές μετρήσεις λόγω του εύρους του WSS τους και συνεπώς δεν πειράζει το σύνολο των page flags που τροποποιούν τα άλλα εργαλεία. Τέλος, και σε αυτή την ομάδα των εφαρμογών τα wss-ref και wss-sys παρουσιάζουν παρόμοιας κλίμακας καθυστερήσεις με το τελευταίο να είναι λίγο μεγαλύτερο σε σχέση με το πρώτο στην πλειοψηφία των περιπτώσεων.

Όσον αφορά τις μετρήσεις που λάβαμε με το εργαλείο wss-proc για διαστήματα παρακολούθησης 1, 10 και 20s που απεικονίζονται στα 5.67 - 5.93 παρατηρούμε αρκετές διαφορές κυρίως στα benchmarks με μικρή χρονική διάρκεια. Αυτό θα μπορούσε να αποδοθεί στο ότι το εργαλείο λαμβάνει πολύ λιγότερες μετρήσεις, όπως απεικονίζεται και στο 5.95, και στο ότι όσο μεγαλώνει το διάστημα παρακολούθησης αυξάνονται και οι χρησιμοποιούμενες σελίδες με αποτέλεσμα να μεγαλώνει το μετρούμενο WSS. Επιπλέον, όπως είναι αναμενόμενο, κατά γενική περίπτωση τα overheads του εργαλείου μικραίνουν όσο αραιώνουν οι λαμβανόμενες μετρήσεις, το οποίο επιβεβαιώνεται και στο 5.94. Βλέποντας την τελευταία στήλη του γραφήματος, που απεικονίζει τους

γεωμετρικούς μέσους των overheads όλων των εφαρμογών, θα λέγαμε πως για 20s διάστημα παρακολούθησης η καθυστέρηση μειώνεται περίπου κατά 1%.

6.1.2 Sampling & Idle Page Tracking

Στη συγκεκριμένη ενότητα θα αξιολογήσουμε τα αποτελέσματα των μετρήσεων που λάβαμε εκτελώντας τις δύο μεθόδους δειγματοληψίας κατά την παρακολούθηση του working set των εφαρμογών 6xx.

Sampling με σταθερό αριθμό δειγμάτων

Όπως βλέπουμε στα γραφήματα 5.96 - 5.122 οι εκτιμήσεις του WSS που λαμβάνουμε με αυτή τη μέθοδο για διάστημα παρακολούθησης 1s και πλήθος δειγμάτων 100 είναι αρκετά κοντά στις πραγματικές με εξαίρεση τις περιπτώσεις των εφαρμογών 602.gcc_s, 605.mcf_s, 623.xalancbmk_s, 628.pop2_s, και 649.fotonik3d_s, στις οποίες το υπολογιζόμενο μέγεθος του working set παίρνει μηδενικές τιμές. Αυτό μπορεί να αποδοθεί στις σελίδες που λάβαμε ως δείγματα κατά την εκτέλεση των συγκεκριμένων πειραμάτων, οι οποίες μπορεί να δεσμεύτηκαν από τις εφαρμογές αλλά να μη χρησιμοποιήθηκαν ενεργά καθόλη τη διάρκεια εκτέλεσής τους.

Όσον αφορά το πλήθος των μετρήσεων που λαμβάνει το τροποποιημένο εργαλείο wss-proc, παρατηρώντας το 5.124, είναι φανερό πως αυξήθηκε σημαντικά σε βαθμό που ξεπερνάει το αντίστοιχο πλήθος του wss-sys. Αυτό είναι αναμενόμενο, καθώς δεν ελέγχεται πια το σύνολο των σελίδων μνήμης που αγγίζουν τα benchmarks και αποφεύγεται το μεγαλύτερο ποσοστό των syscalls που καλούνταν σε αντίθεση περίπτωση. Σημαντική βελτίωση παρουσιάζεται και όσον αφορά τη μετρική της απόδοσης, όπως βλέπουμε στο 5.123, με τα overheads που εισάγει το εργαλείο να μην ξεπερνούν το 1% σχεδόν σε όλες τις εφαρμογές.

Μελετώντας τα γραφήματα 5.125 - 5.205, στα οποία αναπαρίστανται οι μετρήσεις με το wss-proc για διαστήματα παρακολούθησης 1, 10 και 20s παρατηρούμε ορισμένες διαφορές. Συγκεκριμένα, η δειγματοληψία φαίνεται να αποδίδει ακριβέστερες εκτιμήσεις για διαστήματα παρακολούθησης 10 και 20s ιδιαίτερα στα benchmarks 602.gcc_s, 603.bwaves_s, 607.cactuBSSN_s, 628.pop2_s, 631.deepsjeng_s, 644.nab_s και 649.fotonik3d_s. Αυτό θα μπορούσε να αποδοθεί στο ότι αφού το διάστημα παρακολούθησης είναι πιο ευρύ, οι σελίδες που έχουν επιλεχθεί ως δείγματα εμφανίζονται ως ενεργές σε μεγαλύτερο ποσοστό ελέγχων-μετρήσεων συνεισφέροντας έτσι σε καλύτερες εκτιμήσεις. Επιπλέον, όπως είναι αναμενόμενο, το πλήθος μετρήσεων και τα overheads που απεικονίζονται στα 5.206 - 5.211, μειώνονται ακόμα περισσότερο.

Αυξάνοντας τα δείγματα από 100 σε 1000 δεν παρατηρούμε πολλές διαφορές στις εκτιμήσεις των WSS που απεικονίζονται στα γραφήματα 5.212 - 5.238, με εξαίρεση τα benchmarks 600.perlbench_s-3, 619.lbm_s, 625.x264_s-2. Παρόλο που αυξάνεται

το πλήθος κατά μία τάξη μεγέθους, τα overheads όπως βλέπουμε στο 5.239, δεν αυξάνονται σχεδόν καθόλου και παραμένουν γύρω στο 1%. Ομοίως με τα πειράματα που εκτελέσαμε για 100 δείγματα, και σε αυτή την περίπτωση όταν αυξάνουμε το διάστημα παρακολούθησης σε 10 και 20s παρατηρούμε αρκετές βελτιώσεις στα προαναφερθέντα benchmarks που συνοδεύονται με περαιτέρω μειώσεις στα overheads.

Sampling με ρυθμό δειγματοληψίας

Μελετώντας τα γραφήματα 5.328 - 5.335 παρατηρούμε πως οι εκτιμώμενες μετρήσεις του WSS των εφαρμογών πλησιάζουν κατά μεγάλο βαθμό τις πραγματικές και αυτό ισχύει για όλους τους ρυθμούς δειγματοληψίας που επιλέξαμε να εκτελέσουμε τα πειράματά μας. Επιπλέον, από το 5.336 προκύπτει πως η μέθοδος απέδωσε σημαντικά στη μείωση των overheads που εισήχθησαν στις εφαρμογές με τον ρυθμό των 1024 να παρουσιάζει τη μεγαλύτερη μείωση, όπως ήταν αναμενόμενο.

Sampling με ρυθμό δειγματοληψίας σε εικονικό περιβάλλον

Παρατηρώντας τα γραφήματα 5.337 - 5.344 βλέπουμε πως οι εκτιμώμενες μετρήσεις του WSS των εφαρμογών από τα εργαλεία που εκτελέστηκαν σε host πλησιάζουν αρκετά τη μορφή των πραγματικών όπως αυτές λήφθηκαν σε περιβάλλον guest. Η μόνη διαφορά είναι ότι παρουσιάζουν μια αύξηση στα μετρούμενα WSS που δικαιολογείται καθώς τα εργαλεία εκτιμούν το WSS της εφαρμογής qemu-system-x86_64 . Παρόλο που οι εφαρμογές στον guest δε συνεκτελέστηκαν με καμία άλλη διεργασία, το αυξημένο WSS θα μπορούσε να αποδοθεί στις background διεργασίες του συστήματος. Αυτό επιβεβαιώνεται από το 5.338 που παρουσιάζει μεγάλο WSS και συνεπώς οι εκτιμώμενες μετρήσεις δεν αλλοιώθηκαν λόγω του μικρού working set των background διεργασιών του guest.

Επιπλέον, από το 5.345 προκύπτει πως η μέθοδος δειγματοληψίας απέδωσε σημαντικά στη μείωση των overheads που εισήχθησαν στις εφαρμογές ρίχνοντάς τα κατά μέσο όρο στο 3%. Τέλος, το πλήθος μετρήσεων αυξήθηκε κατά πολύ πλησιάζοντας αρκετά αυτό του wss-ref που εκτελέστηκε στο εικονικό περιβάλλον μαζί με τις εφαρμογές.

6.1.3 Intermittent Page Tracking

Σύμφωνα με τα γραφήματα 5.374 - 5.400 παρατηρούμε πως η συγκεκριμένη μέθοδος μέτρησης working set αποδίδει πολύ καλά όσον αφορά τη μετρική της ακρίβειας και για τα τρία εύρη σφάλματος που μελετήσαμε. Πιο συγκεκριμένα, για σφάλμα $|err_r| < 10\%$ το εκτιμώμενο WSS κατά γενική περίπτωση ακολουθεί τις συχνές εναλλαγές του πραγματικού WSS, ενώ όσο αυτό αυξάνεται η ευαισθησία των εκτιμήσεων μειώνεται, χωρίς βέβαια να επηρεάζει κατά σημαντικό βαθμό την ακρίβεια τους.

Μελετώντας το γράφημα 5.402 παρατηρούμε πως υπάρχει σημαντικότατη μείωση στις μετρήσεις που λαμβάνονται με τη συγκεκριμένη μέθοδο με το εργαλείο wss-ref που χρησιμοποιήθηκε να απενεργοποιείται για τουλάχιστον το μισό χρόνο της εκτέλεσης των διαφόρων εφαρμογών. Πιο συγκεκριμένα, για τα benchmarks των οποίων το working set δεν παρουσιάζει ιδιαίτερες μεταβολές, όπως είναι τα 619.lbm_s, 620.omnetpp_s, 621.wrf_s και 628.pop2_s το monitoring tool μειώνει σχεδόν 100% τις ληφθείσες μετρήσεις του.

Όσον αφορά τη μετρική της απόδοσης, παρατηρούμε και σε αυτή την περίπτωση μεγάλη βελτίωση ιδίως στις περιπτώσεις που το wss-ref παρουσίαζε overheads μεγαλύτερα του 10%. Πιο συγκεκριμένα, βλέποντας το 5.401 και ειδικότερα τις περιπτώσεις των 603.bwaves_s, 619.lbm_s, 631.deepsjeng_s και 649.fotoniki3d_s, οι καθυστερήσεις των εφαρμογών όταν το μελετούμενο εύρο σφάλματος είναι ±30% δεν ξεπερνάνε το 5% του χρόνου εκτέλεσής τους, ενώ μάλιστα για τα 619.lbm_s και 631.deepsjeng_s αυτές είναι αμελητέες.

6.2 Αξιολόγηση πειραμάτων σε τροποποιημένο πυρήνα Linux

6.2.1 Sampling με σταθερό αριθμό σελίδων

Με βάση τα γραφήματα 5.403 - 5.427 που αφορούν τα πειράματα που εκτελέσαμε σε τροποποιημένο πυρήνα με τη λειτουργικότητα της δειγματοληψίας και συγκεκριμένα αυτή που λαμβάνει σταθερό αριθμό δειγμάτων, παρατηρούμε πως, όσον αφορά τη μετρική της ακρίβειας, τα αποτελέσματα δεν είναι σε όλες τις περιπτώσεις ικανοποιητικά. Παραδείγματα ανακριβών εκτιμήσεων με μεγάλη απόκλιση φαίνεται να είναι τα 5.403, 5.405, 5.415, 5.418, 5.419, 5.421 και 5.426. Αυτό θα μπορούσε να αποδοθεί στη λήψη μη αντιπροσωπευτικών σελίδων ως δείγματα. Στην αντίστοιχη τεχνική μέτρησης που εφαρμόζουμε σε μη τροποποιημένο πυρήνα στο userspace ελέγχουμε στον πρώτο γύρο μετρήσεων εάν σελίδες που λαμβάνουμε ως δείγματα να παρουσιάζουν μηδενικό RSS, και σε αυτή την περίπτωση φροντίζουμε να διαλέξουμε διαφορετικά. Ωστόσο, σε επίπεδο πυρήνα αυτό δεν είναι δυνατό, καθώς η επιλογή των δειγμάτων γίνεται κατά τη διαδικασία του reset στη συνάρτηση clear_idle_write() και το μέγεθος του RSS ή του WSS γίνεται γνωστό αφότου γίνει το read από τα αντίστοιχα αρχεία στο /proc. Συνεπώς, τα δείγματα που λαμβάνουμε μπορεί να είναι αδρανείς σελίδες που δε χρησιμοποιούνται ενεργά από τη διεργασία και συνεπώς να μην συντελούν στον υπολογισμό σωστών εκτιμήσεων.

Όσον αφορά τη μετρική της απόδοσης, παρατηρώντας το 5.428 συμπεραίνουμε πως η μέθοδος είναι πολύ αποδοτική, μειώνοντας τις καθυστερήσεις κάτω από 2.5% σε σχέση με το χρόνο εκτέλεσης των εφαρμογών, με την πλειοψηφία των overheads να είναι αμελητέα. Τέλος, από το 5.429 είναι φανερό πως η συγκεκριμένη τεχνική

επιταχύνει τη λήψη των μετρήσεων κατά σημαντικό βαθμό, με αποτέλεσμα το πλήθος τους να αυξάνεται συγκριτικά με τις περιπτώσεις που δεν είναι ενεργοποιημένη η δειγματοληψία και όταν δε χρησιμοποιείται το interface του clear_idle.

6.2.2 Sampling με ρυθμό δειγματοληψίας

Μελετώντας τα γραφήματα 5.430 - 5.433 παρατηρούμε πως χρησιμοποιώντας τη συγκεκριμένη μέθοδο μέτρησης working set λαμβάνουμε αρκετά ακριβείς μετρήσεις σε όλα τα πειράματα που εκτελέσαμε με τις πιο ακριβείς να είναι εκείνες με ρυθμό δειγματοληψίας μικρότερο του 64. Σημαντική βελτίωση παρατηρούμε και στα overheads όπως απεικονίζονται στο 5.434 με τις καθυστερήσεις να πέφτουν κάτω από το 5% στις καλύτερες περιπτώσεις.

6.2.3 Intermittent Page Tracking & Sampling

Μελετώντας τα γραφήματα 5.435 - 5.455 παρατηρούμε πως στις περιπτώσεις που η δειγματοληψία με σταθερό αριθμό δειγμάτων δεν είχε αποδώσει, έχουμε παρόμοια αποτελέσματα όσον αφορά την ακρίβεια των μετρήσεων. Αυτό είναι αναμενόμενο, καθώς η συγκεκριμένη μέθοδος βασίζεται στη συγκεκριμένη λειτουργικότητα και μπορεί μόνο να χειροτερέψει την ακρίβεια και όχι να τη βελτιώσει. Όσον αφορά τις εφαρμογές που δεν παρουσίαζαν τέτοια θέματα, έχουμε ικανοποιητικά αποτελέσματα ακόμα και για πιο χαλαρά εύρη σφάλματος, όπως είναι το $err_r = \pm 30\%$.

Σχετικά με τα overheads που εισάγει η συγκεκριμένη μέθοδος, όπως αυτά παρουσιάζονται στο 5.456, βλέπουμε σημαντική μείωση σε σχέση με αυτά του εργαλείου wss-ref ιδιαίτερα στις περιπτώσεις που οι καθυστερήσεις του τελευταίου ξεπερνούσαν το 5%. Σχεδόν στο σύνολο των benchmarks η χρονική επιβάρυνση της μεθόδου είναι αμελητέα και δεν ξεπερνά το 2%. Τέλος, παρατηρώντας το 5.457 προκύπτει πως το monitoring tool μειώνει σημαντικά τις μετρήσεις του ειδικά για πιο χαλαρά εύρη σφάλματος με τις περιπτώσεις των 619.lbm_s, 620.omnetpp_s, 621.wrf_s και 628.pop2_s να πλησιάζουν κοντά στο 100% του ποσοστού μείωσης μετρήσεων.

Κεφάλαιο 7

Περιπτώσεις Χρήσης

Στο κεφάλαιο αυτό θα γίνει αναφορά σε περιπτώσεις χρήσης της μετρικής working set τόσο σε εικονικό περιβάλλον εργασίας όσο και σε τοπικό.

7.1 Εκτέλεση σε Εικονικό Περιβάλλον

7.1.1 Ορισμοί

Εικονικοποίηση είναι η τεχνολογία που επιτρέπει τη δημιουργία πολλαπλών προσομοιωμένων περιβάλλοντων ή ειδικών πόρων από ένα φυσικό σύστημα (hardware). Ειδικό λογισμικό γνωστό ως hypervisor συνδέεται άμεσα στο hardware και επιτρέπει στο χρήστη τη διαμέριση ενός συστήματος σε διακριτά και ασφαλή περιβάλλοντα, γνωστά ως εικονικές μηχανές (VMs). Οι hypervisors μπορούν να εφαρμοστούν είτε πάνω σε ένα λειτουργικό σύστημα είτε να εγκατασταθούν κατευθείαν στο hardware. Τα VMs βασίζονται στην ικανότητα του hypervisor να διαχωρίζει τους πόρους του συστήματος από το hardware και να τους κατανέμει κατάλληλα. Μια εικονική μηχανή (VM) είναι στην ουσία ένα λειτουργικό σύστημα (OS) ή ένα περιβάλλον εφαρμογής που είναι εγκατεστημένο πάνω σε λογισμικό το οποίο προσομοιώνει ειδικό hardware. Ο χρήστης της εικονικής μηχανής έχει την ίδια εμπειρία σα να χρησιμοποιούσε ο ίδιος το ειδικό hardware.

Το φυσικό σύστημα, εξοπλισμένο με τον hypervisor, καλείται host, ενώ τα VMs που κάνουν χρήση των πόρων του καλούνται guests. Οι guests αντιμετωπίζουν τους υπολογιστικούς πόρους, όπως είναι η CPU, μνήμη και ο αποθηκευτικός χώρος, σαν ένα σύνολο πόρων που μπορούν εύκολα να μεταφερθούν και να επαναδεσμευθούν. Λογισμικό με τα κατάλληλα δικαιώματα μπορεί να ελέγξει τους εικονικούς πόρους, ώστε οι guests να λαμβάνουν αυτούς που έχουν ανάγκη όποτε τους χρειάζονται.

Ο hypervisor, προσομοιώνει πλήρως τους πόρους του φυσικού συστήματος, όπως είναι η CPU, η μνήμη, ο σκληρός δίσκος, το δίκτυο και άλλοι, δίνοντας τη δυνατότητα

στις εικονικές μηχανές να μοιράζονται από κοινού τους φυσικούς πόρους. Ο hypervisor μπορεί να προσομοιώσει πολλαπλές εικονικές πλατφόρμες hardware που είναι απομονωμένες μεταξύ τους, επιτρέποντας στις εικονικές μηχανές να τρέξουν πιθανώς διαφορετικά λειτουργικά συστήματα πάνω από τον ίδιο host. Οι εικονικές μηχανές χρησιμοποιούν τους διαθέσιμους πόρους με αποδοτικό τρόπο, μειώνοντας τις ανάγκες για επιπλέον hardware και τα κόστη συντήρησης που τις συνοδεύουν, όπως και την κατανάλωση ενέργειας. Επιπλέον, διευκολύνουν τη διαχείριση σφαλμάτων καθώς το εικονικό hardware δεν παθαίνει βλάβες. Οι διαχειριστές μπορούν να κάνουν χρήση των εικονικών μηχανών για να απλοποιήσουν τις διαδικασίες των backups, την 'ανάρρωση' από σφάλματα και γενικά διάφορες άλλες λειτουργίες διαχείρισης.

Οι εικονικές μηχανές δεν απαιτούν εξειδικευμένο hardware για hypervisors. Η εικονικοποίηση, ωστόσο, απαιτεί μεγαλύτερο εύρος, αποθηκευτικό χώρο, και επεξεργαστική ικανότητα αν πρόκειται το φυσικό σύστημα να φιλοξενήσει πολλές εικονικές μηχανές. Τα VMs μεταφέρονται εύκολα, μπορούν να αντιγραφούν και ανακατανεμηθούν μεταξύ host servers για να βελτιστοποιηθεί η χρήση των φυσικών πόρων.

Η χρήση των εικονικών μηχανών συνοδεύεται από αρκετά προβλήματα διαχείρισης, πολλά από τα οποία μπορούν να αντιμετωπιστούν μέσω εφαρμογής γενικών πρακτικών διαχείρισης συστημάτων υπολογιστών και χρήσης ειδικών εργαλείων για τη διαχείριση των VMs. Υπάρχουν ορισμένοι κίνδυνοι επίσης σε περιπτώσεις που ο host φιλοξενεί πολλές εικονικές μηχανές, συμπεριλαμβανομένης της επιβάρυνσης πόρων ή ενδεχόμενης διακοπής λειτουργίας σε πολλαπλά VMs λόγω πιθανού σφάλματος hardware.

Οι εικονικές μηχανές έχουν πολλαπλές χρήσεις, αλλά γενικά χρησιμοποιούνται όταν υπάρχει ανάγκη για διαφορετικά λειτουργικά συστήματα και επεξεργαστικές δυνατότητες από εφαρμογές που τρέχουν ταυτόχρονα.

7.1.2 Memory Ballooning

Virtual memory ballooning είναι η μέθοδος ανάκλησης μνήμης που χρησιμοποιείται από τον hypervisor για να επιτρέπει στον host να ανακτήσει αχρησιμοποίηση μνήμη από συγκεκριμένους guests εικονικών μηχανών και να τη μοιράσει σε άλλους που τη χρειάζονται. Το memory ballooning επιτρέπει το συνολικό μέγεθος της κύριας μνήμης που χρειάζεται κάθε guest να υπέρβει το διάθεσιμο μέγεθος της φυσικής μνήμης του host. Όταν οι πόροι μνήμης του host εξαντλούνται, τότε το memory ballooning δεσμεύει την απαιτούμενη από τα κατάλληλα VMs.

Εάν ένα VM χρησιμοποιεί μόνο ένα μέρος της μνήμης που έχει δεσμεύσει, τότε μέσω της τεχνικής του memory ballooning η υπόλοιπη διατίθεται στον host για να τη χρησιμοποιήσει κατάλληλα. Για παράδειγμα, έαν όλα τα VMs σε ένα host έχουν δεσμεύσει 8Gb μνήμης, κάποια από αυτά θα χρησιμοποιήσουν μόνο το μισό από το μερίδιό τους. Την ίδια στιγμή, ένα VM μπορεί να χρειαστεί 12Gb μνήμης για μία

απαιτητική διεργασία. Η μέθοδος του memory ballooning επιτρέπει στον host να δανειστεί την αχρησιμοποίητη μνήμη και να την αποδώσει στα VMs με τις υψηλότερες απαιτήσεις σε μνήμη.

Το guest λειτουργικό σύστημα τρέχει μέσα στο VM, το οποίο έχει δεσμεύσει ένα κομμάτι μνήμης και δεν έχει γνώση της συνολικής μνήμης που διαθέτει ο host. Με το memory ballooning το guest λειτουργικό ενημερώνεται αν υπάρξει έλλειψη μνήμης από τη μεριά του host.

Εταιρείες που παρέχουν τεχνολογία εικονικοποίησης όπως η VMware προσφέρουν και τη λειτουργικότητα του memory ballooning. Τα VMware memory ballooning, Microsoft Hyper-V dynamic memory και το ανοιχτού λογισμικού KVM balloon είναι παρόμοια σαν ιδέες. Ο host χρησιμοποιεί drivers που τρέχουν στα VMs για να αποφασίσει πόση μνήμη μπορεί να πάρει πίσω από ένα VM που διαθέτει πλεόνασμα. Οι balloon drivers θα πρέπει να έχουν εγκατασταθεί σε κάθε VM που συμμετέχει στη τεχνική του memory ballooning.

Οι balloon drivers λαμβάνουν την πληροφορία για το μέγεθος που πρέπει να λάβει το balloon από τον hypervisor και έπειτα 'φουσκώνουν'(inflate) το balloon δεσμεύοντας τον κατάλληλο αριθμό φυσικών σελίδων από τον guest μέσα στο VM. Η αντίθετη διαδικασία κατά την οποία οι drivers απελευθερώνουν τις διαθέσιμες σελίδες στον guest είναι γνωστό ως 'ξεφούσκωμα'(deflate) του balloon.

Η τεχνική του virtual memory ballooning μπορεί να δημιουργήσει αρκετά προβλήματα στην απόδοση τόσο του host όσο και των guests. Όταν ο balloon driver κάνει inflate το balloon σε τέτοιο βαθμό που το VM δεν έχει αρκετή μνήμη για να τρέξει τις διεργασίες του, τότε θα εκκινήσει η διαδικασία του swapping. Αυτό θα προκαλέσει καθυστέρηση στο VM αναλόγως το μέγεθος της μνήμης που πρέπει να αποζημιώσει και την ποιότητα των λειτουργιών input/output που πρέπει να αναλάβει.

Προβλήματα σαν τα προαναφερθέντα μπορούν να αντιμετωπιστούν αν ο hypervisor γνωρίζει το μέγεθος του working set των VMs, ώστε κάθε φορά που χρειάζεται να δεσμεύσει μνήμη από ένα VM να ελέγχει αν το μέγεθος του balloon ξεπερνάει το WSS του. Με αυτόν τον τρόπο μπορεί αποφευχθεί το ενδεχόμενο swappingκαι να βελτιωθεί η απόδοση του VM.

7.1.3 Διαχείριση Μνήμης

Οι μοντέρνες πολιτικές διαχείρισης μνήμης βελτιστοποιούν την απόδοση μεταβάλλοντας το χώρο που αναλογεί σε κάθε εργασία (task) κάθε φορά που οι ανάγκες του αλλάζουν. Οι πολιτικές αυτές χωρίζονται στις λεγόμενες τοπικές (local) και στις καθολικές (global). Μια τοπική πολιτική εκτιμά τις ανάγκες μνήμης του κάθε task ανεξάρτητα από αυτές των άλλων tasks και δεσμεύει επαρκή μνήμη για να διαφυλάξει

την τοπικότητα καθενός από αυτά. Μια καθολική πολιτική συσχετίζει τη δέσμευση μνήμης του task με την τοπικότητά του και δεν δεσμεύει απαραίτητα επαρκή μνήμη για την τοπικότητα κάθε ενεργού task. Οι τοπικές πολιτικές είναι αντιπροσωπευτικό παράδειγμα μεθόδων που βασίζονται στο working set των tasks, ενώ οι καθολικές πολιτικές αντιπροσωπεύουν αλγόριθμους όπως ο CLOCK που χρησιμοποιούν καθολικές λίστες ελάχιστα χρησιμοποιημένων σελίδων (LRU - Least Recently Used). Οι Carr et al. [2] πρότειναν ένα αλγόριθμο διαχείρισης εικονικής μνήμης, τον επονομαζόμενο WSCLOCK, που συνδυάζει τον αλγόριθμο τοπικού working set, τον καθολικό CLOCK αλγόριθμο και ένα μηχανισμό ελέγχου νέου φορτίου για βοηθητικές προσβάσεις μνήμης. Το μέγεθος του working set εκτιμάται βάση του referenced flag των σελίδων.

7.1.4 Επαναφορά Μνήμης

Για να είναι πιο πρακτικές οι λειτουργίες της αποθήκευσης και επαναφοράς, οι αποθηκευμένες εικονικές μηχανές (VMs) θα πρέπει να είναι σε θέση να επαναφέρονται γρήγορα σε λειτουργία. Δυστυχώς, η διαδικασία της ανάκλησης του αποθηκευμένου image του VM από συσκευή μόνιμης αποθήκευσης μπορεί να είναι πολύ αργή, ειδικά όταν τα VMs μεγαλώνουν πολύ σε μνήμη. Μια πιθανή λύση για να ελαχιστοποιηθεί ο χρόνος επαναφοράς του VM είναι να ανακληθεί η μνήμη του με lazy τρόπο αφότου ξεκινήσει πρώτα. Ωστόσο, προσβάσεις σε μνήμη που δεν έχει ανακληθεί πλήρως μπορούν υποβαθμίσουν την απόδοση και σε ορισμένες περιπτώσεις να καταστήσουν το VM ακατάλληλο για χρήση για πολύ περισσότερο χρόνο. Οι Zhang et al. [12] πρότειναν μία νέα lazy τεχνική επαναφοράς, επονομαζόμενη ως working set restore, που βελτιώνει την απόδοση της διαδικασίας επαναφοράς αποθηκευμένων VMs. Το working set για κάθε VM εκτιμάται ελέγχοντας τα access-bits των σελίδων και τα ίχνη μνήμης για να παρακολουθήσει τις προσβάσεις σε αυτή.

7.1.5 Ανάκληση Μνήμης και Συμπίεση

Οι σύγχρονοι hypervisors διαχειρίζονται τη φυσική μνήμη ενός εικονικοποιημένου server δεσμεύοντας για κάθε ενεργό VM το κατάλληλο μέγεθος φυσικής μνήμης , ενώ την ελαχιστοποίηση της κατανάλωσης πόρων μνήμης την πετυχαίνουν ενοποιώντας σελίδες με πανομοιότυπο περιεχόμενο και συμπιέζοντας σελίδες φυσικής μνήμης. Με αυτές τις τεχνικές ο hypervisor κατορθώνει να μεγιστοποιεί το πλήθος των VMs που τρέχουν σε έναν server. Το πλήθος των φυσικών σελίδων που χρειάζεται ένα VM σε οποιαδήποτε στιγμή κατά την εκτέλεσή του, είναι εξ΄ ορισμού το working set του για εκείνη τη χρονική στιγμή. Οι Chiang et al. [20] υλοποίησαν έναν διαχειριστή μνήμης, επονομαζόμενο Gatun, που μετράει συνεχώς το working set καθενός από τα ενεργά VMs και εκμεταλλεύεται αυτή την πληροφορία για να βελτιστοποιήσει τη χρήση φυσικής μνήμης στον server. Η πληροφορία του WSS επιτρέπει στο Gatun να ανακαλεί μόνο την αχρησιμοποίηση μνήμη από τα τρέχοντα VMs χωρίς να μετριάσει την απόδοσή τους και να να συμπιέζει το σωστό υποσύνολο σελίδων μνήμης για

κάθε VM, ώστε να περιορίσει όσον το δυνατόν περισσότερο την πίεση μνήμης που υφίστανται.

7.2 Εκτέλεση σε τοπικό περιβάλλον

7.2.1 Shared Cache Partitioning

Η χρήση κρυφών μνημών cache (on-chip memory) βελτιώνει κατά κανόνα την επίδοση των εφαρμογών επειδή, διατηρώντας κάποια δεδομένα on chip, μειώνει τις χρονοβόρες και ενεργειακά κοστοβόρες προσβάσεις του επεξεργαστή στην κύρια μνήμη. Στους σύγχρονους επεξεργαστές συνήθως συνυπάρχουν ταυτόχρονα ιδιωτικές (private caches) και διαμοιραζόμενες μνήμες cache (shared caches). Οι ιδιωτικές μνήμες ανήκουν αποκλειστικά σε έναν φυσικό πυρήνα και είναι προσπελάσιμες μόνο από αυτόν. Οι διαμοιραζόμενες μνήμες βρίσκονται στο τελευταίο επίπεδο στην ιεραρχία μνήμης (Last-Level-Cache, LLC) και είναι κοινόχρηστες μεταξύ των επεξεργαστών μιας υπολογιστικής μονάδας. Παρότι οι ιδιωτικές μνήμες εξασφαλίζουν απομόνωση (isolation), είναι μικρές για να εξυπηρετήσουν το συγκριτικά μεγάλο φορτίο των εφαρμογών που τρέχουν στα κέντρα δεδομένων. Από την την άλλη πλευρά, οι διαμοιραζόμενες μνήμες cache έχουν επαρκές μέγεθος ώστε να εξυπηρετούν τις εφαρμογές, όμως, κατά τη συνεκτέλεση αυτών δημιουργείται ανταγωνισμός για αυτόν τον κοινόχρηστο πόρο, που αυξάνει με τη σειρά του το συνολικό miss-rate, την κατανάλωση ενέργειας, προκαλεί κορεσμό στο bandwidth, κακή ποιότητα υπηρεσίας, μη δικαιοσύνη (unfaireness) στο σύστημα και κακή επίδοση.

Οι παραπάνω περιορισμοί οδήγησαν στη μελέτη και ανάπτυξη τεχνικών διαχείρισης της κοινόχρηστης μνήμης cache με στόχο τον κατάλληλο διαμοιρασμό των τμημάτων της στις εφαρμογές (Cache Partitioning, CP). Το CP σε περιβάλλον συνεκτέλεσης παρέχει απομόνωση των εφαρμογών, συμβάλει στην καλύτερη επίδοση της εκτέλεσής τους και στη δικαιοσύνη του συστήματος. Αποτελέσματα προσομοιώσεων αυτής της τεχνικής δείχνουν πως αυτή είναι πιο αποδοτική όταν η εκάστοτε διαμέριση που αντιστοιχεί σε μία εφαρμογή είναι αρκετά μεγάλη ώστε να χωράει το working set της ή έστω ένα μεγάλο τμήμα της. Συνεπώς, και σε αυτή την περίπτωση κρίνεται σημαντική η πληροφορία του μεγέθους του working set για τη βελτιστοποίηση του CP.

Κεφάλαιο 8

Σχετική Έρευνα

Σε αυτό το κεφάλαιο θα γίνει αναφορά σε άλλες μεθόδους μέτρησης και εκτίμησης του μεγέθους του working set που έχουν μελετηθεί σε σχετική βιβλιογραφία.

8.1 Τεχνικές Εκτίμησης του Working Set σε Software

Ο Waldspurger [3] πρότεινε μία στατιστική μέθοδο δειγματοληψίας για να λάβει εκτιμήσεις του working set χωρίς την ανάμειξη του guest. Κάθε VM δειγματοληπτείται ανεξάρτητα, με περίοδο δειγματοληψίας που είναι προσαρμόσιμη και μετριέται αναλογικά με το χρόνο εκτέλεσης του VM. Κάθε σελίδα του δείγματος παρακολουθείται μέσω της ακύρωσης οποιωνδήποτε αντιστοιχήσεων που έχουν αποθηκευτεί και σχετίζονται με το PFN της, όπως οι καταχωρήσεις στον TLB ή η κατάσταση της εικονικοποιημένης MMU. Η επόμενη πρόσβαση του guest στη σελίδα θα γίνει αντιληπτή λόγω των σφαλμάτων, οι αντιστοιχίσεις της θα αποκατασταθούν και ο μετρητής 'ενεργών' σελίδων θα αυξηθεί κατά ένα. Στο τέλος της περιόδου δειγματοληψίας, μία στατιστική εκτίμηση του τμήματος της μνήμης στο οποίο απέκτησε πρόσβαση το VM θα υπολογιστεί ως το κλάσμα $f = t/n$. Οι εκτιμήσεις εξομαλύνονται κατά τις συνεχόμενες περιόδους δειγματοληψίας κάνοντας χρήση εκθετικά σταθμισμένων κινούμενων μέσων όρων με διαφορετικές παραμέτρους κέρδους.

Οι Berg et al. [4] παρουσίασαν την StatCache, που αποτελεί υλοποίηση ενός γρήγορου εργαλείου παρακολούθησης cache βασισμένο σε στατιστική τεχνική μοντελοποίησης. Μπορεί να εντοπίσει τις εντολές του πηγαίου κώδικα που προκαλούν μη αποδοτική χρήση της cache, να αναλύσει εξαρτήσεις δεδομένων μεταξύ εντολών πηγαίου κώδικα, να παράγει γράφους working set και να σκιαγραφήσει το προφίλ μη τροποποιημένων μονονηματικών εφαρμογών που τρέχουν στον host, συμπεριλαμβανομένων και των συνεπειών της cache στον κώδικα βιβλιοθηκών. Το working set κάθε διεργασίας εξάγεται από την καμπύλη miss ratio, που δείχνει το κλάσμα των

cache misses που θα μετατρέπονταν σε cache hits εάν η δεσμευμένη μνήμη του VM αυξανόταν.

Οι Lu et al. [10] πρότειναν την τεχνική της Exclusive Cache, κατά την οποία κάθε VM διαθέτει ένα μικρό κομμάτι μνήμης (direct memory), και η υπόλοιπη μνήμη είναι υπό τη διαχείριση του hypervisor ως exclusive cache. Μόλις εξαντληθεί η direct μνήμη, τότε το VM θα στέλνει πακέτα σελίδων στον hypervisor αντί στη μνήμη swap, και συνεπώς όλες οι προσβάσεις που θα προκαλέσουν σφάλμα στην εναπομείνασα μνήμη του VM θα μπορούν να εντοπιστούν από τον hypervisor. Με αυτόν τον τρόπο θα μπορούν να προβλεφθούν η καμπύλη miss ratio όπως και τα μοτίβα προσβάσεων μνήμης και κατ΄ επέκταση να εκτιμηθεί το working set του VM με στόχο την κατάλληλη δέσμευση μνήμης του.

Οι Zhao et al. [11] εισήγαγαν τη χρήση του MEmory Balancer (MEB), εργαλείου που παρακολουθεί δυναμικά τη χρήση μνήμης κάθε VM, προβλέπει τις ανάγκες τους όσον αφορά τη μνήμη, και περιοδικά ανακατανέμει τη μνήμη του host στις διάφορες εικονικές μηχανές. Μέσω της δυναμικής παρακολούθησης των προσβάσεων στη μνήμη κατασκευάζει το LRU ιστόγραμμα, από το οποίο μπορεί να προβλεφθεί το working set του guest. Θα πρέπει να σημειωθεί πως με τη μέθοδο δειγματοληψίας δε μπορεί να γίνει πρόβλεψη του working set που να υπερβαίνει την τρέχουσα δεσμευμένη μνήμη του host, ενώ με το συνδυασμό του LRU ιστογράμματος και των δεδομένων που έχουν ληφθεί όσον αφορά το swapping μπορεί να γίνει ακριβέστερη εκτίμηση του WSS.

Οι Zhao et al. [13] σχεδίασαν την τεχνική διαλείπουσας παρακολούθησης, κατά την οποία ο έλεγχος των προσβάσεων στη μνήμη διακόπτεται τα χρονικά διαστήματα κατά τα οποία έχει προβλεφθεί πως οι ανάγκες μνήμης διατηρούνται στα ίδια επίπεδα. Με τη χρήση των performance countersμπορεί να εκτιμηθεί πότε οι ανάγκες για μνήμη έχουν αλλάξει και να επανεκκινήσει ακολούθως η διαδικασία παρακολούθησης μνήμης. Με αυτόν τον τρόπο το overhead που προκαλείται από τη δημιουργία LRU miss-ratio καμπύλων ελαχιστοποιείται με μια αμελητέα απώλεια σε ακρίβεια.

Η Melekhova [16] προτείνει τη μέθοδο ανάλυσης με παλινδρόμηση, ως νέα προσέγγιση για την εκτίμηση του μεγέθους του working set. Η συγκεκριμένη μέθοδος εκτιμά την κατανάλωση μνήμης μέσω της χρήσης κάποιων συμβάντων εικονικοποίησης, όπως είναι οι εντολές ή τα σήματα που διακόπτουν τη λειτουργία του επεξεργαστή στον guest και απαιτούν υποστήριξη εικονικοποίησης. Η παλινδρόμηση εφαρμόζεται για τη συσχέτιση της συμπεριφοράς του λειτουργικού του guest και των προκληθέντων συμβάντων, γεγονός που θα οδηγήσει ακολούθως στην εκτίμηση του WSS χωρίς να γίνουν υποθέσεις εκ των προτέρων για τη συμπεριφορά του guest αλλά και χωρίς κάποια επιπλέον καθυστέρηση.

Ο Pusukuri [19] ανέπτυξε ένα στατιστικό μοντέλο βασισμένο σε μεθόδους τεχνι-

κές μάθησης για την εκτίμηση του working set πολυνηματικών προγραμμάτων που τρέχουν σε πολυπύρηνα συστήματα. Η βασική ιδέα είναι ο συσχετισμός του μεγέθους του working set του προγράμματος με διάφορες μετρικές χρήσης των πόρων του, όπως είναι το μέγεθος της resident μνήμης ή το TLB miss rate. Τα αποτελέσματα τέτοιων μετρικών συλλέγονται με τη χρήση performance counters ή απλών εργαλείων που είναι διαθέσιμα στα σύγχρονα λειτουργικά συστήματα. Τα μοντέλα συσχέτισης που χρησιμοποιούνται δημιουργούνται με μεθόδους εποπτευόμενης μάθησης (supervised learning), όπου ζευγάρια τιμών εισόδου και εξόδου εντοπίζονται και έπειτα ένα στατιστικό μοντέλο εκπαιδεύεται ώστε να προβλέπει αντίστοιχες τιμές εξόδου όταν παρατηρούνται ανάλογες τιμές εισόδου.

Οι Chiang et al. [20] σχεδίασαν το Gatun, ένα πρόγραμμα διαχείρισης μνήμης που μετράει συνεχώς το μέγεθος του working set καθενός από τα ενεργά VMs και εκμεταλλεύεται αυτή την πληροφορία ώστε να ανακαλεί μόνο αχρησιμοποίητη φυσική μνήμη από αυτά για να μη μετριάσει την απόδοσή τους. Επιπλέον για κάθε VM συμπιέζεται το κατάλληλο υποσύνολο σελίδων μνήμης τους ώστε να μειώθει η πίεση μνήμης που υφίστανται στο μέγιστο βαθμό. Στο πρόγραμμα Gatun όλες οι λειτουργικότητες που αφορούν το memory ballooning, τη δέσμευση φυσικής μνήμης όπως και το βαθμό συμπίεσης των σελίδων, βασίζονται στη μετρική του true working set (TWS) κάθε VM. Αυτό εξάγεται μέσω της σταδιακής μείωσης της διαθέσιμης φυσικής μνήμης του VM μέχρι το σημείο που θα ξεκινήσει η διαδικασία του swapping, και της συλλογής πληροφοριών από το μηχανισμό ανάκλησης μνήμης που είναι εγκατεστημένος σε κάθε guest.

Οι Wang et al. [22] ανέπτυξαν ένα αποδοτικό σύστημα μέτρησης WSS με βελτιστοποιήσεις για δομές δεδομένων, ένα μηχανισμό για τη μείωση της συχνότητας παρακολούθησης όπως και ένα εργαλείο για την εξισορρόπηση φορτίου μνήμης (Memory Balancer), που ανακατανέμει δυναμικά τη μνήμη των guests βασιζόμενο σε προβλεπόμενες εκτιμήσεις του working set. Συγκρίνοντας τη συγκεκριμένη δουλειά με αυτή που δημοσιεύτηκε από τους [11] και [13], αξίζει να αναφερθεί πως στη συγκεκριμένη εργασία παρουσιάζεται μια περιεκτική και εις βάθος ανάλυση και σύγκριση των τεχνικών που είχαν στόχο τη μείωση του overhead στη μέτρηση του μεγέθους του working set εικονικών μηχανών. Μεταξύ των τεχνικών που αναλύονται περιλαμβάνεται αυτή που υλοποιούνταν με λίστα LRU βασισμένη σε δέντρα AVL για την κατάταξη των προσβάσεων μνήμης σε επίπεδο σελίδας όπως και αυτή που χρησιμοποιούσε ένα Dynamic Hot Set για τη μείωση των φυσικών σελίδων που παρακολουθούνταν ενεργά. Επιπλέον ο ευριστικός αναδρομικός αλγόριθμος εξισορρόπησης φορτίου μνήμης που είχε προταθεί στο VEE'09 αντικαθίσταται με έναν δυναμικό αλγόριθμο ώστε να βελτιστοποιείται η λειτουργία του συστήματος όταν υπάρχει μεγάλος αριθμός από VMs.

Οι Nitu et al. [25] πρότειναν το Badis, ένα σύστημα παρακολούθησης και εκτίμησης working set το οποίο εκμεταλλεύεται τα οφέλη διάφορων υπάρχοντων μεθόδων με στόχο την επίτευξη υψηλού βαθμού ακρίβειας χωρίς μεγάλη επεμβατικότητα σε

επίπεδο κώδικα και προσαρμόζει δυναμικά τη φυσική μνήμη που δεσμεύει κάθε VM ανάλογα με το εκτιμώμενο WSS του. Πιο συγκεκριμένα το Badis συνδυάζει τις τεχνικές των VMware [3] και Geiger [7] για την εκτίμηση του WSS, υλοποιώντας μία μηχανή τριών πεπερασμένων καταστάσεων (FSM), κατά την οποία το working set υπολογίζεται είτε με την πρώτη τεχνική, ή με τη δεύτερη ή και με τις δύο. Η τεχνική που εφαρμόζεται στο VMware υπολογίζει το WSS δειγματοληπτώντας κάποιες από τις σελίδες μνήμης του VM και ακυρώνοντας τις αντιστοιχήσεις τους σε δομές όπως οι πίνακες σελίδων, ή οι TLB. Αν και με αυτή τη μέθοδο μπορεί να υπολογίστει το WSS όταν το VM έχει πλεόνασμα μνήμης, δε μπορεί να εκτιμηθεί σε καταστάσεις έλλειψης μνήμης. Αντίθετα, η τεχνική Geiger παρακολουθεί τις μετακινήσεις των σελίδων και τις επαναφορτώσεις τους προς και από τη μνήμη swap , καθιστώντας δυνατή τη μέτρηση του working set μόνο σε περιπτώσεις έλλειψης μνήμης από το VM.

8.2 Τεχνικές Εκτίμησης του Working Set σε Hardware

Οι Dani et al. [18] πρότειναν μια ακριβή μέθοδο εκτίμησης του working set διεργασιών ή νημάτων που εκτελούνται ταυτόχρονα σε ένα τσιπ μικροεπεξεργαστή με αμελητέο overhead υλικού της τάξης 0.1% του μεγέθους της cache κάνοντας χρήση επιπλέον tags. Πέρα από τα τελευταία που βοηθάνε στην αναγνώριση των διευθύνσεων των cache linesμε μοναδικό τρόπο, η μέθοδος περιλαμβάνει τη διατήρηση ενός 'ενεργού' bit για κάθε cache line, το οποίο τίθεται σε κατάσταση θέσης όταν γίνει cache hit. Ταυτόχρονα αυξάνεται ένας μετρητής (activeLineReplaced) για την εκτίμηση του αριθμού των σελίδων που έχουν μεταφερθεί εκτός της cache, ενώ χρησιμοποιήθηκαν στο ίδιο χρονικό διάστημα i. Το μέγεθος του working set υπολογίζεται με βάση το πλήθος των cache lines που έχουν το 'ενεργό' bit τους σε κατάσταση θέσης, και του μετρητή που υποδηλώνει ποιες cache lines αντικαταστάθηκαν με ενεργό το τελευταίο bit.

8.3 Ανάλυση του Working Set σε Benchmark Suites

Ο Gove [9] ανέλυσε το μέγεθος του working set των benchmarks Spec2006, με στόχο να εξετάσει αν η αύξηση στις ανάγκες μνήμης συνοδεύεται και με την αύξηση της χρησιμοποιούμενης μνήμης. Οι εκτιμήσεις για το WSS έγιναν με τη χρήση ενός εργαλείου τύπου SHADE, το οποίο εντοπίζει τη διεύθυνση μνήμης στην οποία αντιστοιχεί κάθε load και store της διεργασίας που εκτελείται. Οι διευθύνσεις μνήμης που εντοπίζονται είναι στο επίπεδο μπλοκ των 64-bytes και το WSS υπολογίζεται ως το γινόμενο του πλήθους των χρησιμοποιημένων μπλοκ μνήμης με το μέγεθος του καθενός (64 bytes). Η συγκεκριμένη εργασία παραθέτει το μέσο μέγεθος του working set που παρατηρείται καθόλη τη διάρκεια εκτέλεσης του φορτίου εργασίας.

8.4 Memory Ballooning

Οι Jones et al. [7] δημιούργησαν το Geiger, ένα διαχειριστή εικονικών μηχανών (Virtual Machine Manager) υλοποιημένο σε Xen hypervisor , που μπορεί να συνάγει επακριβώς ποιες σελίδες μεταφέρονται εντός και εκτός του buffer cache του συστήματος. Το Geiger μπορεί να χρησιμοποιηθεί και για την υλοποίηση του MemRx, ενός VMM που εντοπίζει τα working sets των guest VMs, παρατηρώντας τις μετακινήσεις και τις συνεχόμενες επαναφορτώσεις από το buffer cache του λειτουργικού του guest. Έπειτα το MemRx ποσοτικοποιεί τον αριθμό των προσβάσεων μνήμης που θα μπορούσαν να μετατραπούν από misses σε hits για διάφορα μεγέθη μνήμης. Αυτή η πληροφορία χρησιμοποιείται για τη δημιουργία της καμπύλης miss-ratio. Το μέγεθος του working set μπορεί να προκύψει από αυτήν αν εντοπιστεί το κύριο γόνατο της καμπύλης.

Οι Tasoulas et al. [14] εφάρμοσαν το μοντέλο των Μπεϋσιανών δικτύων για να προσφέρουν πιθανοτικές προβλέψεις όσον αφορά τη χρήση μνήμης από το σύστημα. Ένα Μπεϋσιανό δίκτυο πίστης είναι ένα πιθανοτικό γραφικό κατευθυνόμενο ακυκλικό μοντέλο, που δείχνει την υποθετική εξάρτηση μεταξύ ενός συνόλου τυχαίων μεταβλητών. Η συγκεκριμένη εργασία προτείνει τη χρήση του εργαλείου Bayllocator, που προβλέπει πόση επιπλέον μνήμη θα χρειαστούν τα VMs, δεσμεύει αυτό το κομμάτι μνήμης μαζί με ένα επιπλέον ποσοστό, διασφαλίζει πως οι εικονικές μηχανές δεν λαμβάνουν περισσότερη ή λιγότερη μνήμη από αυτή που τους αναλογεί, εξακριβώνει πως ο hypervisor δε θα κάνει swap κανένα μέρος της μνήμης, ανακατανέμει δίκαια την πλεονάζουσα μνήμη του hypervisor στις εικονικές μηχανές και ανακαλεί μνήμη δίκαια για τα VMs που χρειάζονται περισσότερη από το επιτρεπόμενο ποσοστό ώστε να αποφευχθεί οποιοδήποτε swapping από τον hypervisor.

Οι Chiang et al. [15] περιέγραψαν τη σχεδίαση, υλοποίηση και εκτίμηση ενός αλγορίθμου memory ballooning που κάνει χρήση του μεγέθους του working set των VMs. Για να εξάγει αυτή την πληροφορία από κάθε VM, ο αλγόριθμος εκμεταλλεύεται το μηχανισμό ανάκλησης σελίδων μνήμης που είναι εγκατεστημένος σε κάθε guestμειώνοντας σταδιακά τη διαθέσιμη μνήμη κάθε VM μέχρι να ξεκινήσει το swapping. Για αυτό το σκοπό γίνεται χρήση μιας μηχανής τριών πεπερασμένων καταστάσεων (FSM) για να επιλεχθεί ο κατάλληλος κάθε φορά τρόπος για την εκτίμηση του WSS.

Οι Solomie et al. [17] παρουσίασαν τη μέθοδο ballooning σε επίπεδο εφαρμογής (Application-Level Ballooning), κατά την οποία η κύρια μνήμη διαμερίζεται μεταξύ ενός συνόλου εφαρμογών που διαχειρίζονται οι ίδιες τη μνήμη τους τρέχοντας σε συνεγκατεστημένα VMs. Η μέθοδος ALB ανακατανέμει την κύρια μνήμη χωρίς να τερματίζει τις εφαρμογές ή να τις επαναδιαμορφώνει, και τους επιτρέπει να βελτιστοποιούν την απόδοσή τους γνωστοποιώντας τους το ποσοστό της μνήμης που τους αναλογεί.

Οι Kim et al. [21] εισήγαγαν μια νέα τεχνική δυναμικής διαχείρισης μνήμης, με το όνομα Memory Pressure Aware (MPA) Ballooning, κατά την οποία δεσμεύεται μνήμη για κάθε VM με βάση το βαθμό πίεσης μνήμης που υφίστανται. Επιπλέον, το σύστημα αντιδρά προληπτικά και προσαρμόζεται σε ξαφνικές αλλαγές που παρατηρούνται στις ανάγκες μνήμης κάθε VM. Ο hypervisor μετρά κάθε φορά την πίεση μνήμης του συστήματος, συγκρίνοντας το άθροισμα των ανώνυμων σελίδων και των αρχείων σελίδων με τη συνολική διαθέσιμη μνήμη του host, και προσαρμοστικά αλλάζει κάθε φορά την πολιτική που ακολουθείται για τη δέσμευση μνήμης. Η τελευταία λειτουργικότητα βασίζεται σε εκτιμήσεις του working settων VMs, οι οποίες εξάγονται από αναφορές του κάθε guest στον hypervisor σχετικά με τις λίστες ενεργών και ανενεργών σελίδων μνήμης.

Οι Zhang et al. [23] πρότειναν το iBalloon, έναν εξισορροπητή μνήμης για VM χαμηλού κόστους που λειτουργεί με υψηλή ακρίβεια και διαφάνεια. Το iBalloon τρέχει ένα ελαφρύ δαίμονα σε κάθε VM, ο οποίος εκτιμά το μέγεθος του working set του, διαβάζοντας περιοδικά τις μετρικές της συνολικής μνήμης και της ελεύθερης μνήμης από το εικονικό αρχείο του Linux `/proc/meminfo`. Ταυτόχρονα, στον host τρέχει ένας άλλος δαίμονας που συλλέγει πληροφορίες από τους δαίμονες των VMs και αυτομάτως παίρνει την απόφαση για το πως να ανακατανείμει τη μνήμη μεταξύ των VMs. Έπειτα, ο συγκεκριμένος δαίμονας επικοινωνεί με τον balloon driver του host ώστε να πραγματοποιηθεί η ανακατανομή της μνήμης.

Κεφάλαιο 9

Συμπεράσματα

Στο κεφάλαιο αυτό θα παρουσιάσουμε συγκεντρωτικά τα συμπεράσματα που προκύπτουν από την αξιολόγηση των μετρήσεων του Κεφαλαίου 6.

9.1 Τεχνικές σε μη τροποποιημένο πυρήνα Linux

Συγκρίνοντας τις τεχνικές μέτρησης του working set που εφαρμόστηκαν σε επίπεδο χρήστη με μη τροποποιημένο πυρήνα, θα λέγαμε πως αυτές που εφαρμόζουν δειγματοληψία φαίνεται να είναι οι πιο αποτελεσματικές τόσο από άποψη ακρίβειας όσο και απόδοσης. Πιο συγκεκριμένα, παρατηρούμε πως και οι δύο τεχνικές καταφέρνουν να μειώσουν τις καθυστερήσεις που εισάγει το εργαλείο wss-proc κάτω του 1% κατά γενική περίπτωση και η ακρίβεια στις εκτιμήσεις τους είναι κατά μεγάλο βαθμό ικανοποιητική. Σε αυτό το σημείο θα πρέπει να αναφέρουμε πως αν και δεν εφαρμόσαμε τη δειγματοληψία με ρυθμό σε όλα τα 6xx benchmarks, αυτή η τεχνική επέδειξε μεγαλύτερη ακρίβεια σε εφαρμογές που η δειγματοληψία με σταθερό αριθμό δειγμάτων έλαβε μηδενικές μετρήσεις. Ωστόσο, τα overheads που εισήγαγε ήταν συγκρίσιμα με της προηγούμενης μόνο για ρυθμούς μεγαλύτερους του 256. Επιπλέον, η μέθοδος με το σταθερό πλήθος δειγμάτων όταν εφαρμόστηκε για μεγαλύτερα διαστήματα παρακολούθησης σε ορισμένες εφαρμογές έλαβε ακριβέστερες μετρήσεις και όχι μηδενικές όπως στην περίπτωση του 1s interval. Βελτίωση στην ακρίβεια ορισμέων εφαρμογών παρουσιάστηκε και όταν αυξήσαμε το πλήθος δειγμάτων από 100 σε 1000 με μηδαμινό αντίκτυπο στα overheads.

Όσον αφορά τη μέθοδο της διακοπτόμενης μέτρησης working set ή αλλιώς του intermittent tracking, όπως αυτή συνδυάστηκε με το εργαλείο wss-ref, παρατηρούμε πως κατάφερε να μειώσει κατά μεγάλο βαθμό τα overheads που εισήγαγε το εργαλείο ειδικά για πιο χαλαρά εύρη σφάλματος, όπως το ±30%, με αυτά να κυμαίνονται κατά μέσο όρο στο 3%. Ωστόσο, η ακρίβεια δεν ήταν πάντα ικανοποιητική για το συ-

γκεκριμένο εύρος και λαμβάνοντας υπόψη πως οι μέθοδοι δειγματοληψίας επέδειξαν καλύτερη απόδοση σε όλες τις μετρικές που τις αξιολογήσαμε θα λέγαμε πως αυτές είναι προτιμότερες.

9.2 Τεχνικές σε τροποποιημένο πυρήνα Linux

Μελετώντας τα αποτελέσματα των τεχνικών που εφαρμόσαμε σε επίπεδο πυρήνα εκμεταλλευόμενοι τις επεκτάσεις που υλοποιήθηκαν στον πυρήνα 4.9.110 Linux καταλήγουμε σε παρόμοια συμπεράσματα όπως τα παραπάνω. Πιο συγκεκριμένα, και οι δύο μέθοδοι δειγματοληψίας φάνηκαν πολύ αποδοτικές και από άποψη ακρίβειας αλλά και απόδοσης. Αυτή που λαμβάνει σταθερό αριθμό δειγμάτων κατάφερε να μειώσει τα overheads του εργαλείου μέτρησης WSS κάτω του 1% , και αυτή που λαμβάνει δείγματα με ρυθμό εισήγαγε καθυστερήσεις από 5% έως και 1% για ρυθμούς από 4 μέχρι 1024 αντίστοιχα. Ωστόσο, η τελευταία επέδειξε μεγαλύτερη ακρίβεια στις εκτιμήσεις της σε ορισμένα πειράματα που η προηγούμενη υστερούσε. Ανάγοντας όμως τα συμπεράσματα των μεθόδων που εφαρμόσαμε σε επίπεδο χρήστη στη συγκεκριμένη περίπτωση, θα μπορούσαμε να υποθέσουμε πως αν λαμβάναμε περισσότερα δείγματα ή αν εκτελούσαμε τις μετρήσεις με ευρύτερο διάστημα παρακολούθησης τα αποτελέσματα μπορεί να ήταν ικανοποιητικότερα για τη μέθοδο δειγματοληψίας με σταθερό αριθμό δειγμάτων.

Όσον αφορά τη μέθοδο διακοπτόμενης μέτρησης όπως αυτή συνδυάστηκε με τη μέθοδο δειγματοληψίας με σταθερό αριθμό δειγμάτων, θα μπορούσαμε να πούμε πως αν και κατέστησε τα overheads του εργαλείου αμελητέα για όλα τα εφαρμοσθέντα εύρη σφάλματος, η ακρίβεια των εκτιμήσεων της δεν ήταν το ίδιο ικανοποιητική σε σύγκριση με τις άλλες μεθόδους μέτρησης. Αυτό θα μπορούσε να αλλάξει αν συνδυαζόταν με τη μέθοδο δειγματοληψίας με ρυθμό ή αν αυξάναμε το πλήθος δειγμάτων στην πρώτη μέθοδο.

9.3 Σύγκριση τεχνικών σε επίπεδο χρήστη και πυρήνα

Θέλοντας να βγάλουμε ένα γενικό συμπέρασμα για τις δύο κατηγορίες μεθόδων που εφαρμόσαμε, θα λέγαμε πως οι μέθοδοι δειγματοληψίας που υλοποιήθηκαν σε επίπεδο πυρήνα εισήγαγαν λίγο μικρότερες καθυστερήσεις σε σύγκριση με τις αντίστοιχές τους σε επίπεδο χρήστη. Ωστόσο, η ακρίβεια των τελευταίων ήταν ικανοποιητικότερη σε ορισμένα benchmarks. Αυτό ωστόσο θα μπορούσε να αλλάξει, όπως προαναφέραμε, αν εφαρμόζαμε ευρύτερο διάστημα παρακολούθησης στις μετρήσεις μας και λαμβάναμε μεγαλύτερο πλήθος δειγμάτων όσον αφορά την πρώτη μέθοδο δειγματοληψίας. Επειδή οι καθυστερήσεις και των δύο κατηγοριών είναι αμελητέες σχεδόν, θα μπορούσαμε να καταλήξουμε πως αυτές που εφαρμόζονται σε μη τροποποιημένο

πυρήνα είναι προτιμότερες από άποψη επεμβατικότητας, καθώς δεν απαιτούν από το χρήστη να επεμβαίνει στον κώδικα πυρήνα.

Τέλος, σχετικά με τις μεθόδους διακοπτόμενης παρακολούθησης, παρατηρούμε πως αυτή σε επίπεδο χρήστη είναι ικανοποιητικότερη καθώς δεν εφαρμόζει δειγματοληψία και πετυχαίνει εξίσου αμελητέα overheads και καλύτερη ακρίβεια.

Bibliography

[1] Peter J. Denning. "The Working Set Model for Program Behavior". In: *Commun. ACM* 11.5 (May 1968), pp. 323–333. ISSN: 0001-0782. DOI: 10.1145/363095.363141. URL: http://doi.acm.org/10.1145/363095.363141.

[2] Richard W. Carr and John L. Hennessy. "WSCLOCK;a Simple and Effective Algorithm for Virtual Memory Management". In: *Proceedings of the Eighth ACM Symposium on Operating Systems Principles*. SOSP '81. Pacific Grove, California, USA: ACM, 1981, pp. 87–95. ISBN: 0-89791-062-1. DOI: 10.1145/800216.806596. URL: http://doi.acm.org/10.1145/800216.806596.

[3] Carl A. Waldspurger. "Memory Resource Management in VMware ESX Server". In: *SIGOPS Oper. Syst. Rev.* 36.SI (Dec. 2002), pp. 181–194. ISSN: 0163-5980. DOI: 10.1145/844128.844146. URL: http://doi.acm.org/10.1145/844128.844146.

[4] Erik Berg and Erik Hagersten. "Fast Data-locality Profiling of Native Execution". In: *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. SIGMETRICS '05. Banff, Alberta, Canada: ACM, 2005, pp. 169–180. ISBN: 1-59593-022-1. DOI: 10.1145/1064212.1064232. URL: http://doi.acm.org/10.1145/1064212.1064232.

[5] Daniel P. Bovet and Marco Cesati. *Understanding Linux Kernel*. O'Reilly, 2005.

[6] Corbet Jonathan, Rubini Alessandro, and Kroah-Hartman Greg. *Linux Device Drivers*. O'Reilly, 2005.

[7] Stephen T. Jones, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. "Geiger: Monitoring the Buffer Cache in a Virtual Machine Environment". In: *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS XII. San Jose, California, USA: ACM, 2006, pp. 14–

24. ISBN: 1-59593-451-0. DOI: 10.1145/1168857.1168861. URL: http://doi.acm.org/10.1145/1168857.1168861.

[8] K. Qureshi Moinuddin and Patt Yale N. "Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches". In: MICRO 39 (2006), pp. 423–432. DOI: 10.1109/MICRO.2006.49. URL: https://dl.acm.org/citation.cfm?id=1194855.

[9] Darryl Gove. "CPU2006 Working Set Size". In: *SIGARCH Comput. Archit. News* 35.1 (Mar. 2007), pp. 90–96. ISSN: 0163-5964. DOI: 10.1145/1241601.1241619. URL: http://doi.acm.org/10.1145/1241601.1241619.

[10] Pin Lu and Kai Shen. "Virtual Machine Memory Access Tracing with Hypervisor Exclusive Cache". In: *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*. ATC'07. Santa Clara, CA: USENIX Association, 2007, 3:1–3:15. ISBN: 999-8888-77-6. URL: http://dl.acm.org/citation.cfm?id=1364385.1364388.

[11] Weiming Zhao and Zhenlin Wang. "Dynamic Memory Balancing for Virtual Machines". In: *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. VEE '09. Washington, DC, USA: ACM, 2009, pp. 21–30. ISBN: 978-1-60558-375-4. DOI: 10.1145/1508293.1508297. URL: http://doi.acm.org/10.1145/1508293.1508297.

[12] Irene Zhang et al. "Fast Restore of Checkpointed Memory Using Working Set Estimation". In: *Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. VEE '11. Newport Beach, California, USA: ACM, 2011, pp. 87–98. ISBN: 978-1-4503-0687-4. DOI: 10.1145/1952682.1952695. URL: http://doi.acm.org/10.1145/1952682.1952695.

[13] Weiming Zhao et al. "Low Cost Working Set Size Tracking". In: *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference*. USENIXATC'11. Portland, OR: USENIX Association, 2011, pp. 17–17. URL: http://dl.acm.org/citation.cfm?id=2002181.2002198.

[14] Evangelos Tasoulas, Hårek Haugerud, and Kyrre Begnum. "Bayllocator: A Proactive System to Predict Server Utilization and Dynamically Allocate Memory Resources Using Bayesian Networks and Ballooning". In: *Proceedings of the 26th International Conference on Large*

Installation System Administration: Strategies, Tools, and Techniques. lisa'12. San Diego, CA: USENIX Association, 2012, pp. 111–122. URL: http://dl.acm.org/citation.cfm?id=2432523.2432532.

[15] Jui-Hao Chiang, Han-Lin Li, and Tzi cker Chiueh. "Working Set-based Physical Memory Ballooning". In: *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13)*. San Jose, CA: USENIX, 2013, pp. 95–99. ISBN: 978-1-931971-02-7. URL: https://www.usenix.org/conference/icac13/technical-sessions/presentation/chiang.

[16] A. Melekhova. "Machine Learning in Virtualization: Estimate a Virtual Machine's Working Set Size". In: *2013 IEEE Sixth International Conference on Cloud Computing*. 2013, pp. 863–870. DOI: 10.1109/CLOUD.2013.91.

[17] Tudor-Ioan Salomie et al. "Application Level Ballooning for Efficient Server Consolidation". In: *Proceedings of the 8th ACM European Conference on Computer Systems*. EuroSys '13. Prague, Czech Republic: ACM, 2013, pp. 337–350. ISBN: 978-1-4503-1994-2. DOI: 10.1145/2465351.2465384. URL: http://doi.acm.org/10.1145/2465351.2465384.

[18] Aparna Mandke Dani, Bharadwaj Amrutur, and Y. N. Srikant. "Toward a Scalable Working Set Size Estimation Method and Its Application for Chip Multiprocessors". In: *IEEE Trans. Comput.* 63.6 (June 2014), pp. 1567–1579. ISSN: 0018-9340. DOI: 10.1109/TC.2012.291. URL: https://doi.org/10.1109/TC.2012.291.

[19] Kishore Kumar Pusukuri. "Working Set Model for Multithreaded Programs". In: *Proceedings of the 2014 International Conference on Timely Results in Operating Systems*. TRIOS'14. Broomfield, CO: USENIX Association, 2014, pp. 1–1. URL: http://dl.acm.org/citation.cfm?id=2750315.2750316.

[20] Jui-Hao Chiang, Tzi-cker Chiueh, and Han-Lin Li. "Memory Reclamation and Compression Using Accurate Working Set Size Estimation". In: *8th IEEE International Conference on Cloud Computing, CLOUD 2015, New York City, NY, USA, June 27 - July 2, 2015*. 2015, pp. 187–194. DOI: 10.1109/CLOUD.2015.34. URL: https://doi.org/10.1109/CLOUD.2015.34.

[21] Jinchun Kim et al. "Dynamic Memory Pressure Aware Ballooning". In: *Proceedings of the 2015 International Symposium on Memory Systems*. MEMSYS '15. Washington DC, DC, USA: ACM, 2015, pp. 103–112.

ISBN: 978-1-4503-3604-8. DOI: 10.1145/2818950.2818967. URL: http://doi.acm.org/10.1145/2818950.2818967.

[22] Zhigang Wang et al. "Dynamic Memory Balancing for Virtualization". In: *ACM Trans. Archit. Code Optim.* 13.1 (Mar. 2016), 2:1–2:25. ISSN: 1544-3566. DOI: 10.1145/2851501. URL: http://doi.acm.org/10.1145/2851501.

[23] Q. Zhang et al. "iBalloon: Efficient VM Memory Balancing as a Service". In: *2016 IEEE International Conference on Web Services (ICWS)*. 2016, pp. 33–40. DOI: 10.1109/ICWS.2016.14.

[24] Brendan D. Gregg. *Working Set Size Estimation*. http://www.brendangregg.com/wss.html. Last Updated: 04-Feb-2018. 2018.

[25] Vlad Nitu et al. "Working Set Size Estimation Techniques in Virtualized Environments: One Size Does Not Fit All". In: *Proc. ACM Meas. Anal. Comput. Syst.* 2.1 (Apr. 2018), 19:1–19:22. ISSN: 2476-1249. DOI: 10.1145/3179422. URL: http://doi.acm.org/10.1145/3179422.

[26] *Idle Page Tracking*. https://www.kernel.org/doc/html/latest/admin-guide/mm/idle_page_tracking.html.

[27] *Proactively reclaiming idle memory*. https://lwn.net/Articles/787611/.

[28] *SPEC CPU 2017*. https://www.spec.org/cpu2017/.

[29] *The challenges of managing memory in a virtual environment*. https://blog.turbonomic.com/blog/on-technology/memory-101-the-challenges-of-managing-memory-in-a-virtual-environment.

[30] *Understanding Full Virtualization, Paravirtualization, and Hardware Assisted Virtualization*. https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/VMware_paravirtualization.pdf.

[31] *Working Set*. https://en.wikipedia.org/wiki/Working_set.