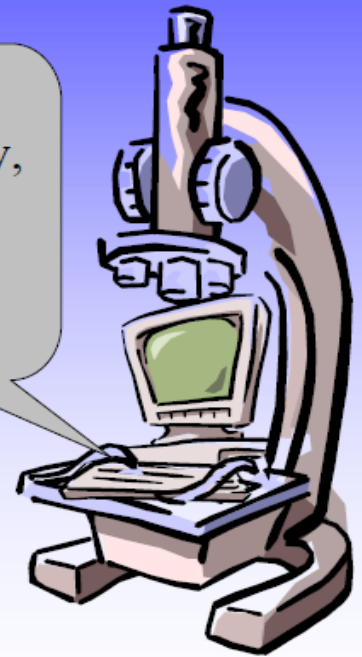


ΠΡΟΣΟΜΟΙΩΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

Εισαγωγή

Complex,
code-heavy,
high-end
digital
system



- Θέματα Ενδιαφέροντος Αρχιτεκτονικής Υπολογιστών
 - Σχεδίαση επεξεργαστή
 - » 1 thread → pipeline, branch prediction
 - » n threads → SMT resource allocation, threads scheduling
 - Ετερογενείς αρχιτεκτονικές
 - » Cell, Intel Sandy Bridge, AMD Fusion, Intel Stellarton (Atom E600C)
 - Ιεραρχία μνήμης
 - » cache sharing, coherence protocols, NUMA
 - Παράλληλα συστήματα
 - » Coherence, Δίκτυα διασύνδεσης
 - » Compilers, automatic parallelization
 - » Programming Models, synchronization costs, locks, computation and communication overheads

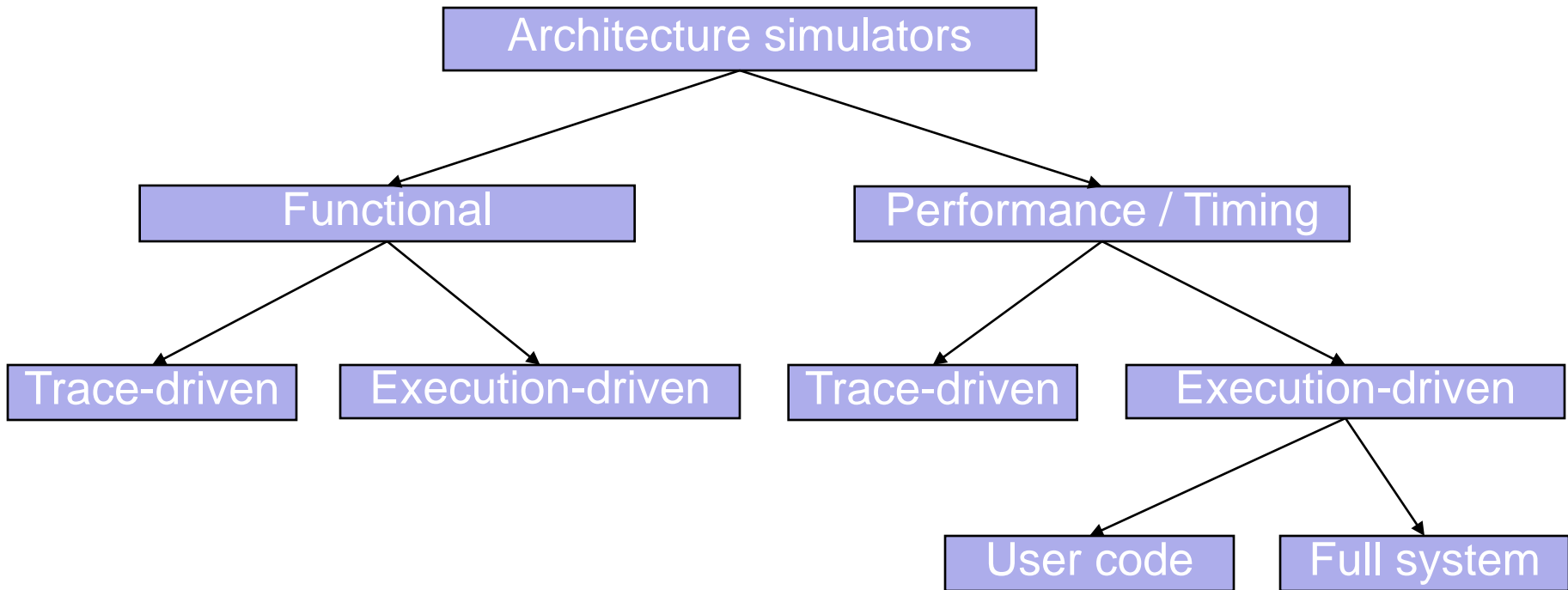
Τρόποι Υλοποίησης

- Χρήση υπαρχόντων μηχανημάτων
 - Μεγάλο κόστος
 - » Sun SPARC Enterprise T5120 server (64 threads, 128GB mem) ~ 16K \$ (2012)
 - » Intel Westmere X5680 (2 * 6 cores, 24 threads) ~7K \$
 - Αδυναμία παρέμβασης στο υλικό τους
 - » pipeline, caches, interconnection network
 - Περιορισμένη δυνατότητα παρακολούθησης και μετρήσεων
 - » performance counters → λίγοι, μικρό documentation
 - Περιορισμός στο σήμερα
 - » μελλοντικές αρχιτεκτονικές (π.χ. chip με 100 ή 1000 threads) ;
- Λύση : **Simulation** (προσομοίωση)

Τι είναι ο simulator;

- Ένα εργαλείο που αναπαράγει τη «συμπεριφορά» ενός υπολογιστικού συστήματος.
- Γιατί να χρησιμοποιήσουμε ένα simulator;
 - Πληροφορίες σχετικά με την εσωτερική λειτουργία
 - » Performance Analysis
 - Δυνατότητα ανάπτυξης λογισμικού για μη διαθέσιμες (ή και μη υπαρκτές) πλατφόρμες
 - Προβλέψεις απόδοσης για διαφορετικές αρχιτεκτονικές.

Simulator taxonomy



Functional vs. Timing Simulators

- Functional Simulators
 - Visible architectural state
 - Προσομοίωση της λειτουργικότητας των εντολών (instructions semantics and functionality), μεταβολή του state (registers, memory)
 - Σωστό program output
 - Κύριος σκοπός: Software development and/or emulation
- Timing Simulators
 - Microarchitecture details
 - Λεπτομερής υλοποίηση των διαφορετικών δομών (pipeline, branch predictors, interconnection networks, memory hierachy, etc)
 - Χρονισμός γεγονότων, προκειμένου να υπολογισθεί ο χρόνος εκτέλεσης του προγράμματος
- **Functional simulation πολύ πιο γρήγορο**

Trace vs. Execution-driven Simulators

- Trace Simulators
 - Εκτέλεση της εφαρμογής σε πραγματική πλατφόρμα → trace (instruction, address, ...)
 - Τα traces χρησιμοποιούνται σαν inputs του simulator
- Execution-driven Simulators
 - Ο simulator εκτελεί την εφαρμογή
 - Διατήρηση application state και architecture state
- Trace-driven simulation συνήθως πιο γρήγορο
 - Διατήρηση μόνο του architecture state, δεν εκτελούνται όλοι οι υπολογισμοί
- Τα traces επιτρέπουν την προσομοίωση proprietary applications & input sets.
- Πρόβλημα : Τα traces δεν μπορούν να συλλάβουν/φανερώσουν την δυναμική συμπεριφορά της εφαρμογής

User code vs. Full system simulators

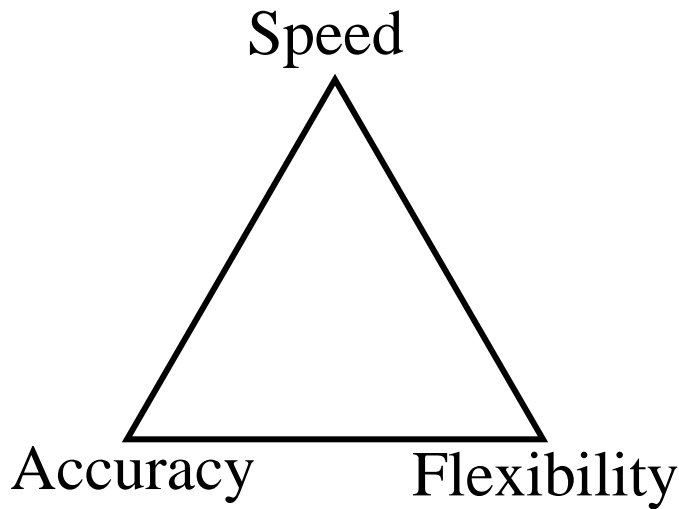
- User code Simulators

- Προσομοίωση μόνο του κώδικα της εφαρμογής
- System calls και I/O εκτελούνται με functional simulation
- Συνήθως το functional emulation πραγματοποιείται από το host OS
 - » host OS = target OS

- Full system Simulators

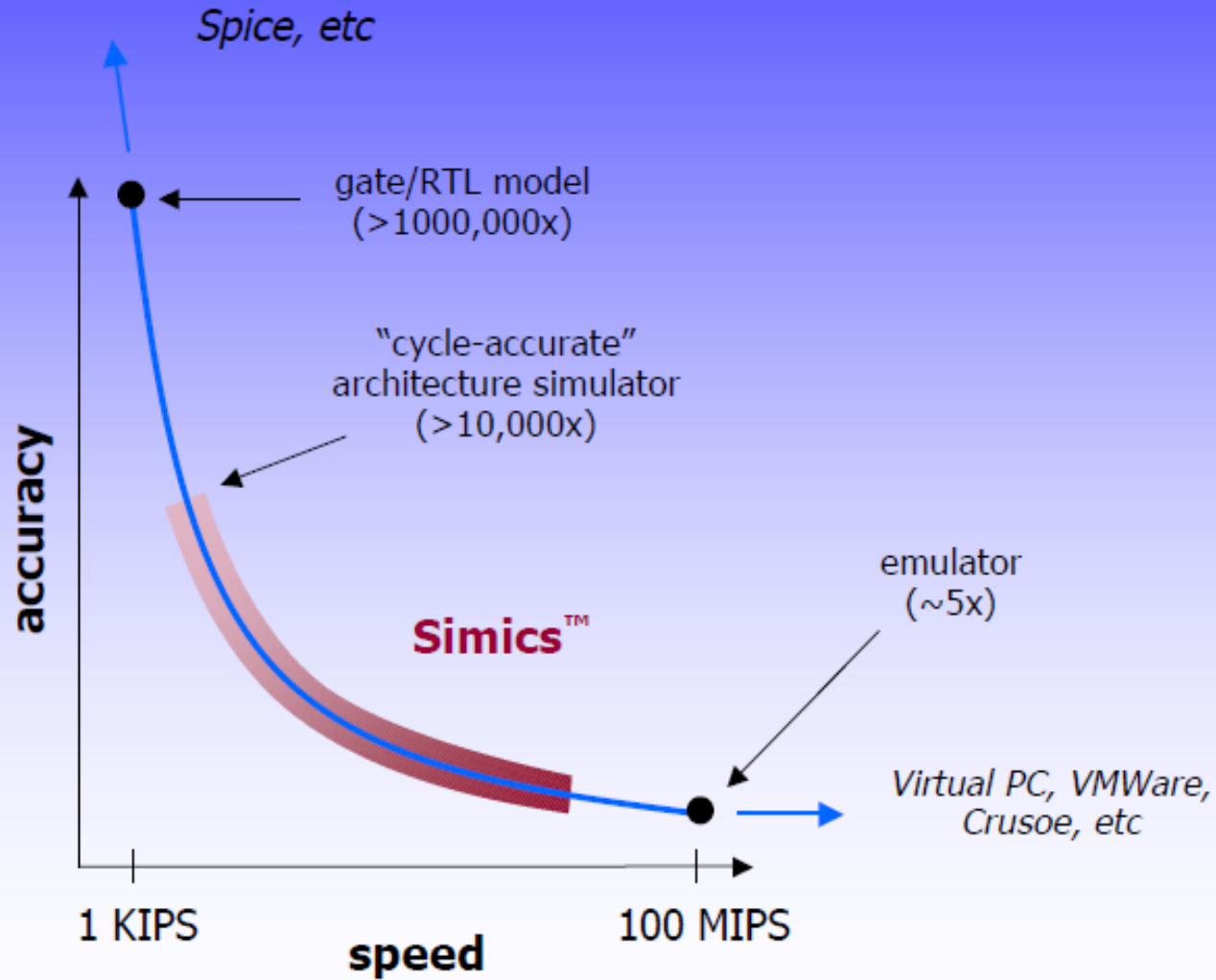
- Προσομοίωση της εφαρμογής
- Προσομοίωση του OS
- Προσομοίωση των devices (disks, network, etc.)

The Zen of architecture simulators



- Δεν υπάρχει ο τέλειος simulator
- FPGA prototypes
 - Γρήγορα, ακριβή, αλλά έλλειψη ευελιξίας
- Λεπτομερή software μοντέλα
 - Ακριβή, ευέλικτα, αλλά αργά
- Αφηρημένα software μοντέλα
 - Γρήγορα, ευέλικτα αλλά όχι ακριβή

Speed vs. Accuracy



Παράδειγμα χρόνων προσομοίωσης (1)

- spec2k with gcc and small inputs

| | Time | Ratio to Native | Ratio to Functional |
|---------------|-------------|-----------------|---------------------|
| Native | 1,054s | -- | -- |
| sim-fast | 2m 47s | 158x | 1 |
| sim-outorder | 1h 11m 07s | 4,029x | 25x |
| simics | 7m 41s | 437x | 1 |
| w/Ruby | 11h 27m 25s | 39,131x | 89x |
| w/Ruby + Opal | 43h 13m 41s | 147,648x | 338x |

Παράδειγμα χρόνων προσομοίωσης (2)

- Οι πραγματικές εφαρμογές παίρνουν ώρες σε πραγματικά μηχανήματα
- Χρειαζόμαστε αρκετά μεγάλες ταχύτητες για να τρέξουμε ένα σημαντικό κομμάτι αυτών των εφαρμογών.

| | Number of Ops (B) |
|---------------------|-------------------|
| Windows XP Boot | 5 |
| Linux RH 6.0 Boot | 4 |
| Windows XP Install | 361 |
| SPECint2000 (train) | 279 |

Προσομοίωση Αρχιτεκτονικής (1)

- Απαιτήσεις

- Γενικότητα (Generality)

- » Μπορεί το εργαλείο να αναλύσει τα workloads?

- » Parallel Systems, Multithreading, Multiple address spaces, OS code, Network Systems, etc.

- Πρακτικότητα (Practicality)

- » Μπορεί το εργαλείο να χρησιμοποιηθεί αποδοτικά;

- » Host assumptions, compiler assumptions, OS modifications, workload language assumptions

- Εφαρμοσιμότητα (Applicability)

- » Μπορεί το εργαλείο να απαντήσει στα ερωτήματά μας;

- » Restricted state that can be monitored, restrictions on parameter visibility, restricted length of observations.

Προσομοίωση Αρχιτεκτονικής (2)

- Πλεονεκτήματα
 - Early availability
 - Ευκολία χρήσης
 - » Πλήρης διαφάνεια και ευκολία παρακολούθησης και μετρήσεων
 - » Διαφορετικά επίπεδα λεπτομέρειας και ακρίβειας
 - Pipelines, caches, branch predictors, ...
 - Hardware devices (timer, drives, cards, ...)
 - » Έλεγχος καινοτόμων προτάσεων/ιδεών
 - Κόστος
 - » Open source (Free)
 - » Academic licenses (Free ή μικρό κόστος για support)

Προσομοίωση Αρχιτεκτονικής (3)

- Προκλήσεις
 - Χρόνος ανάπτυξης των μοντέλων (modeling time)
 - Έλεγχος ορθότητας μοντέλων (validation)
 - Ταχύτητα προσομοίωσης
- Ενεργό ερευνητικό πεδίο
- Πληθώρα επιλογών
 - WindRiver Simics (x86, SPARC, MIPS, Leon, ARM...)
 - AMD SimNow (x86)
 - SimpleScalar (Alpha, PISA, ARM, x86)
 - SimFlex
 - MARSSx86
 - SniperSim

Επιλογή περιβάλλοντος προσομοίωσης

- Κριτήρια Επιλογής
 - Modularity simulator
 - Extensibility simulator
 - Επίπεδο ακρίβειας simulator
 - Ταχύτητα simulator
 - Μέγεθος του design space που θέλουμε να μελετήσουμε
 - Επιλογή κατάλληλων benchmarks

Στατιστικά Προσομοίωσης(1)

- Ο σκοπός ενός timing simulation είναι η συγκέντρωση πληροφοριών και μέτρηση διαφόρων μεγεθών
 - IPC
 - Memory access cycles
 - On-chip network contention
- Τα προγράμματα παρουσιάζουν διαφορετικές φάσεις
 - Initialization phase
 - Main phase
 - Wrap-up phase
- Πότε παίρνουμε τα στατιστικά που μας ενδιαφέρουν;

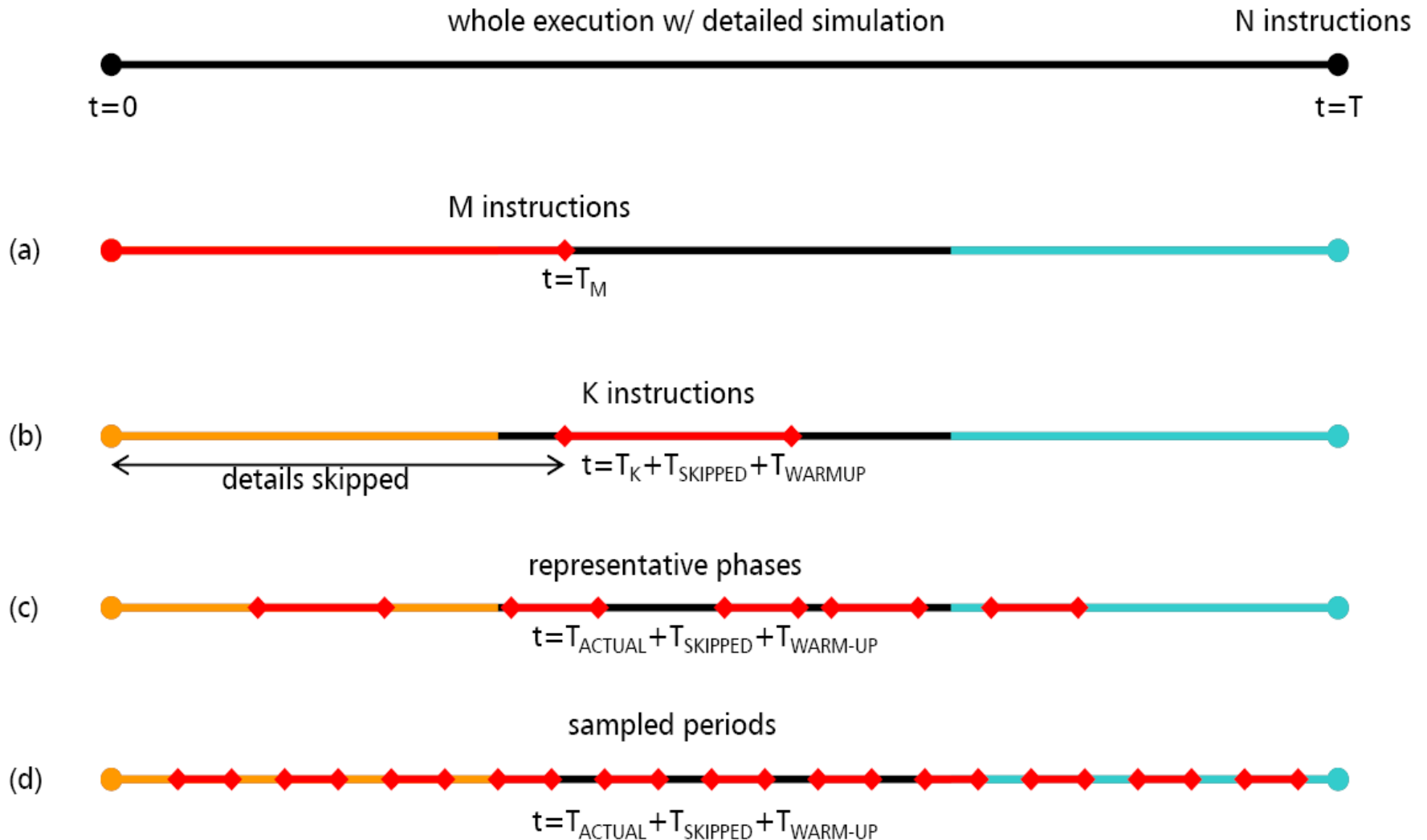
Στατιστικά Προσομοίωσης(2)

- Η προσομοίωση είναι ένα single-thread process
 - Ακόμα και αν προσομοιώνουμε ένα παράλληλο σύστημα
- Η ταχύτητα προσομοίωσης δε βελτιώνεται πια με την τεχνολογία
 - Παράλληλα συστήματα τρέχουν πολλαπλές προσομοιώσεις ταυτόχρονα.
- **Δεν μπορούμε να φτιάξουμε γρήγορους simulators**
 - **Δεν προσομοιώνουμε ολόκληρο το σύστημα**
 - **Δεν προσομοιώνουμε ολόκληρη την εφαρμογή**

Στατιστικά Προσομοίωσης(3)

- Προσομοίωση μικρότερου συστήματος
 - π.χ. 16 cores αντί για 1024
 - Δεν εκθέτει θέματα κλιμακωσιμότητας (scalability)
 - » Ανταγωνισμός για shared resources
 - » Conflicts
 - » Race conditions
- Προσομοίωση μικρότερης εφαρμογής
 - π.χ. Matrix multiply 1K x 1K αντί για 1M x 1M
 - Δεν εξετάζει τα όρια του hardware
 - » Το working set χωράει στις caches
 - » Capacity/conflict issues
 - » Λιγότερη επαναχρησιμοποίηση data/code
 - » Initialization vs. steady state

Στατιστικά Προσομοίωσης

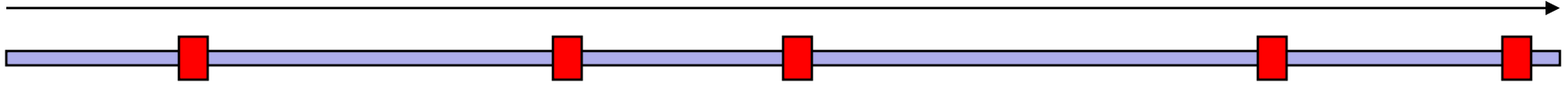


Τεχνικές Προσομοίωσης

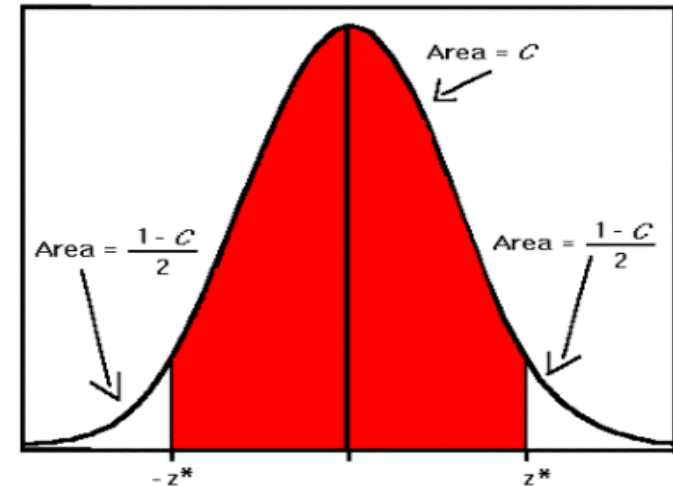
- Sampling
 - Προσομοίωση τυχαίων σημείων της εφαρμογής
- Fast-forwarding
 - Μέχρι να φτάσουμε στο sample point
- Warm-up
 - Γρήγορη προσομοίωση πριν τη φάση των μετρήσεων
- Checkpointing
 - Αποθήκευση architectural state πριν το sample
- Phase detection
 - Επιλογή των samples μετά από ανάλυση της εφαρμογής

Simulation sampling

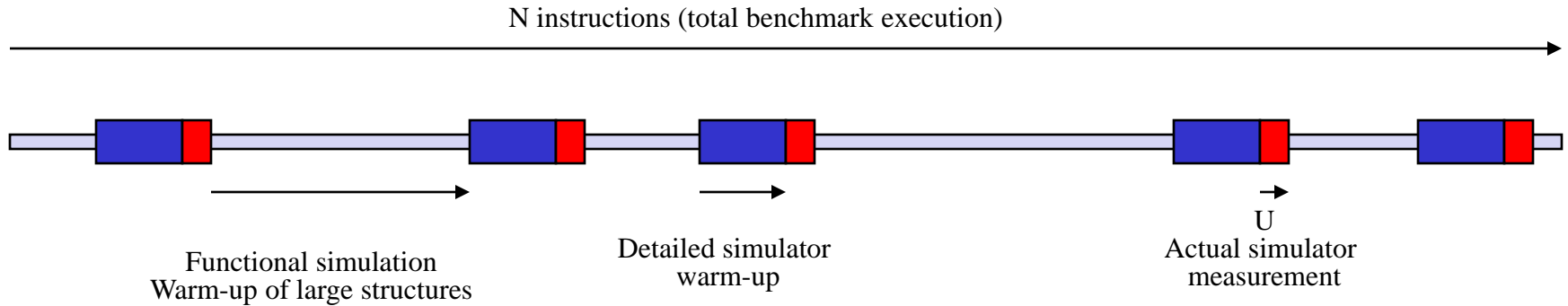
N instructions (total benchmark execution)



- Στατιστική προσέγγιση
 - Εξετάσουμε αντιπροσωπευτικά δείγματα
- Μαθηματική προσέγγιση
 - confidence margin (eg. 95%)
 - confidence interval (eg. +/- 2.5)
- Δυο προσεγγίσεις σχετικά με την επιλογή δειγμάτων
 - Systematic sampling
 - » Sample every N instructions
 - Random sampling

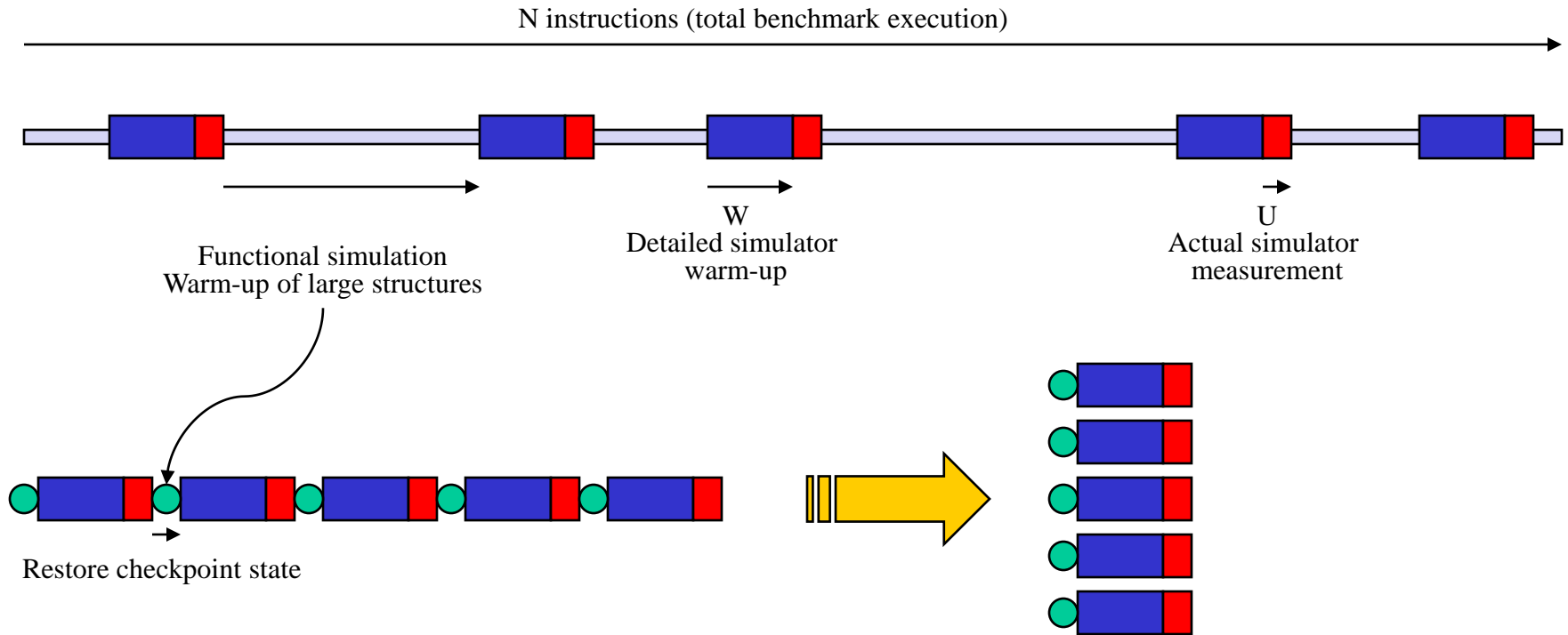


Simulation warm-up



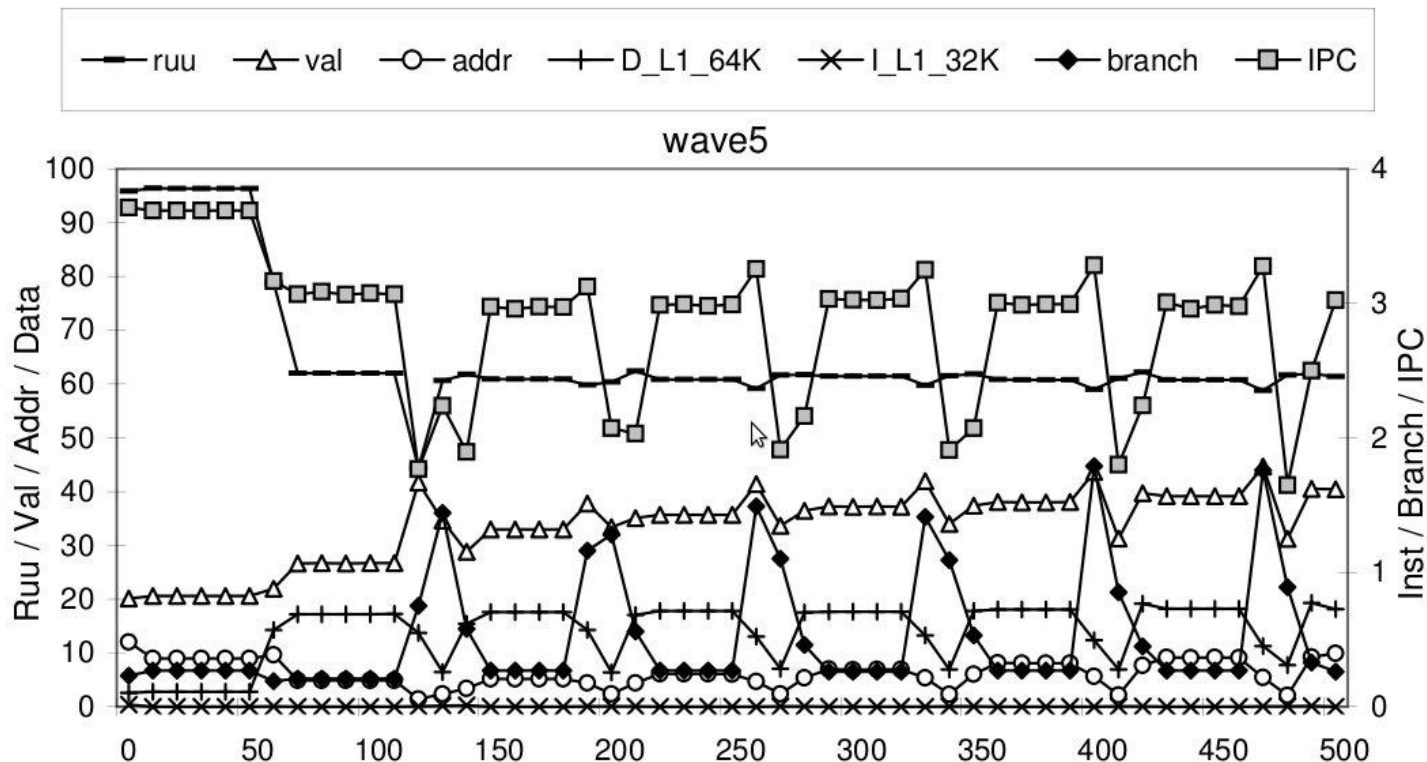
- Δεν μπορούμε να προσομοιώσουμε σωστά αν έχουμε *empty architecture state*
 - Caches' invalid state
 - Branch predictor, TLB, ...
 - Operating system state
 - » Files open / close, read / write pointers, ...
- Χρειάζεται κάποιος χρόνος για warm-up
 - Όσο μεγαλύτερη η δομή, τόσο μεγαλύτερος ο χρόνος

Checkpointing and Sampling



- Αποθήκευση warmed-up state πριν από κάθε sample
 - functional simulation overhead time vs more detailed simulations
 - Παράλληλη προσομοίωση όλων των samples

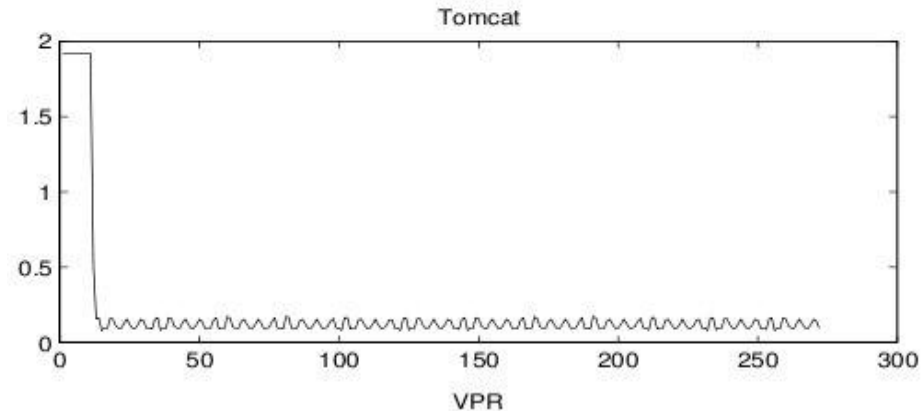
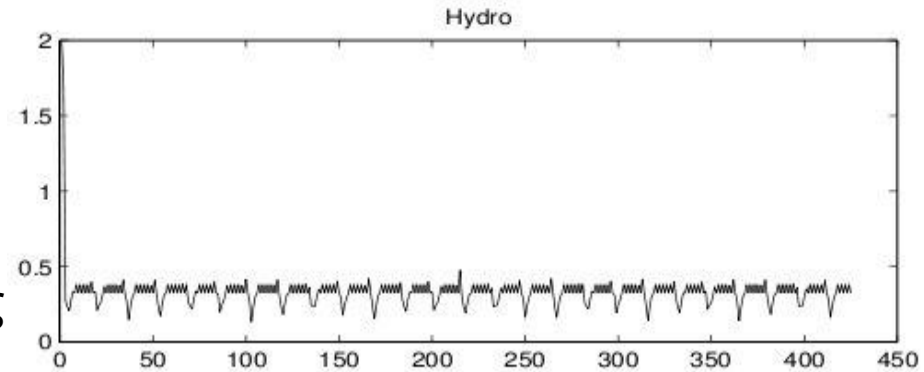
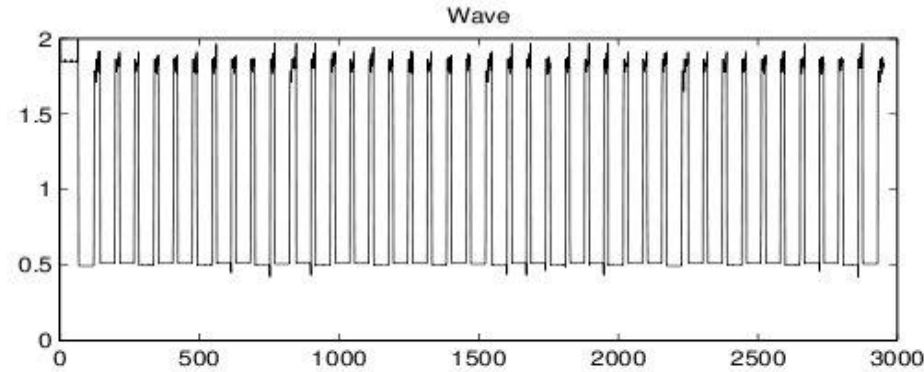
Non-random sampling



- Η κάθε εφαρμογή εμφανίζει φάσεις οι οποίες πολλές φορές επαναλαμβάνονται
 - Επιλογή σωστών samples
- Υποθέτουμε ότι η συμπεριφορά της εφαρμογής εξαρτάται άμεσα από την εκτέλεση του static code.
 - Κάθε φάση αντιστοιχίζεται σε ένα static section του κώδικα

Simpoints - Basic Block Vectors

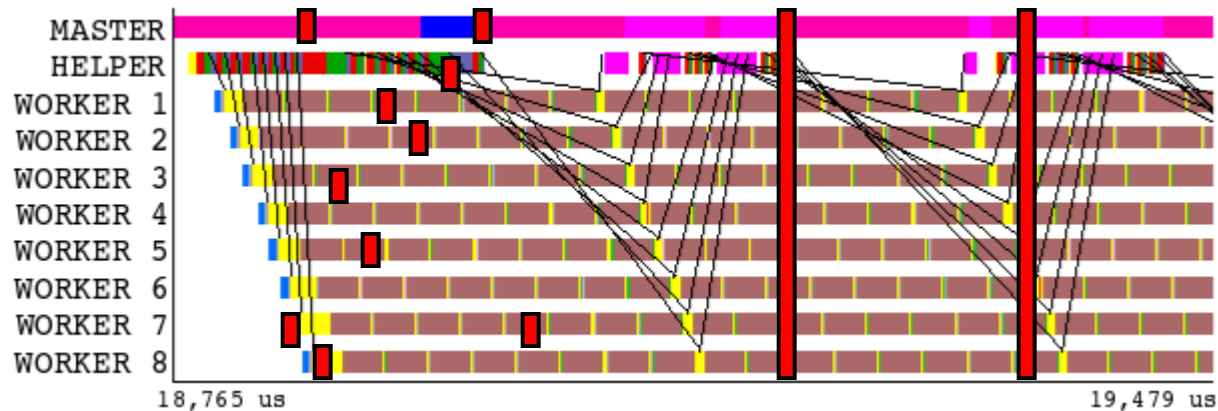
- Μετράμε τις εκτελέσεις για κάθε basic block
 - Για ολόκληρη την εκτέλεση
 - Για την εκτέλεση N εντολών (π.χ. N = 100 million)
- Συγκρίνουμε το sample BBV με το global BBV
 - Manhattan distance
 - Euclidean distance
- Η σύγκριση των BBV αποκαλύπτει επίσης τις περιοδικότητες στην εκτέλεση μιας εφαρμογής
 - Initialization phase
 - Repetitive intervals



Simpoints

- Επιλογή Simpoinths
- Το πιο αντιπροσωπευτικό BBV μπορεί να είναι μακριά
 - Μεγάλος χρόνος fast-forward
 - Επιλογή κάποιου sample νωρίτερα
- Μοναδικό sample;
 - Ένα από κάθε phase
- Τα samples είναι architecture dependent
 - Ο ίδιος κώδικας με διαφορετικό compiler έχει το ίδιο BBV.

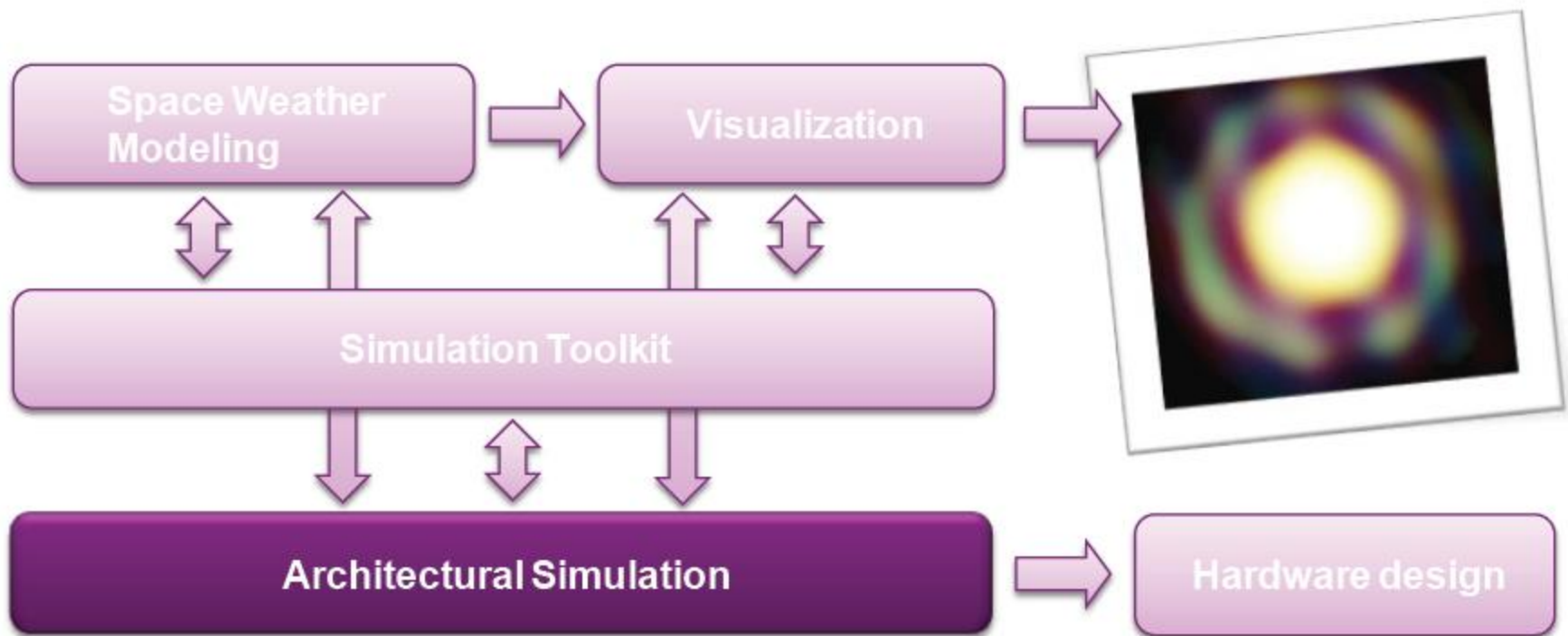
Multithreaded simulation



- Τα τυχαία samples από πολλαπλά threads μπορεί να μη συμπίπτουν χρονικά
- Το «κάθετο» sample δεν εγγυάται σωστή «ευθυγράμμιση»
 - Η ταχύτητα του κάθε thread διαφέρει κατά τη διάρκεια:
 - » Functional simulation
 - » Warm-up
 - » Simulation
 - Το Fast-forward μετριέται σε instructions όχι **χρόνοT**

Sniper Multi-core Simulator

- Συνεργασία μεταξύ Intel (Intel Exascience Lab), IMEC και 5 βελγικών πανεπιστημίων
- Σκοπός : Μελέτη του καιρού του διαστήματος ως HPC workload.



Sniper : A fast and accurate simulator

- Hybrid simulation
 - Analytical interval core model
 - Micro-architecture structure simulation (branch predictors, caches, NoC)
- Multi/Many-core systems running multithreaded & multiprogrammed workloads
- Παράλληλος simulator
- <http://snipersim.org>

Sniper : Top Features

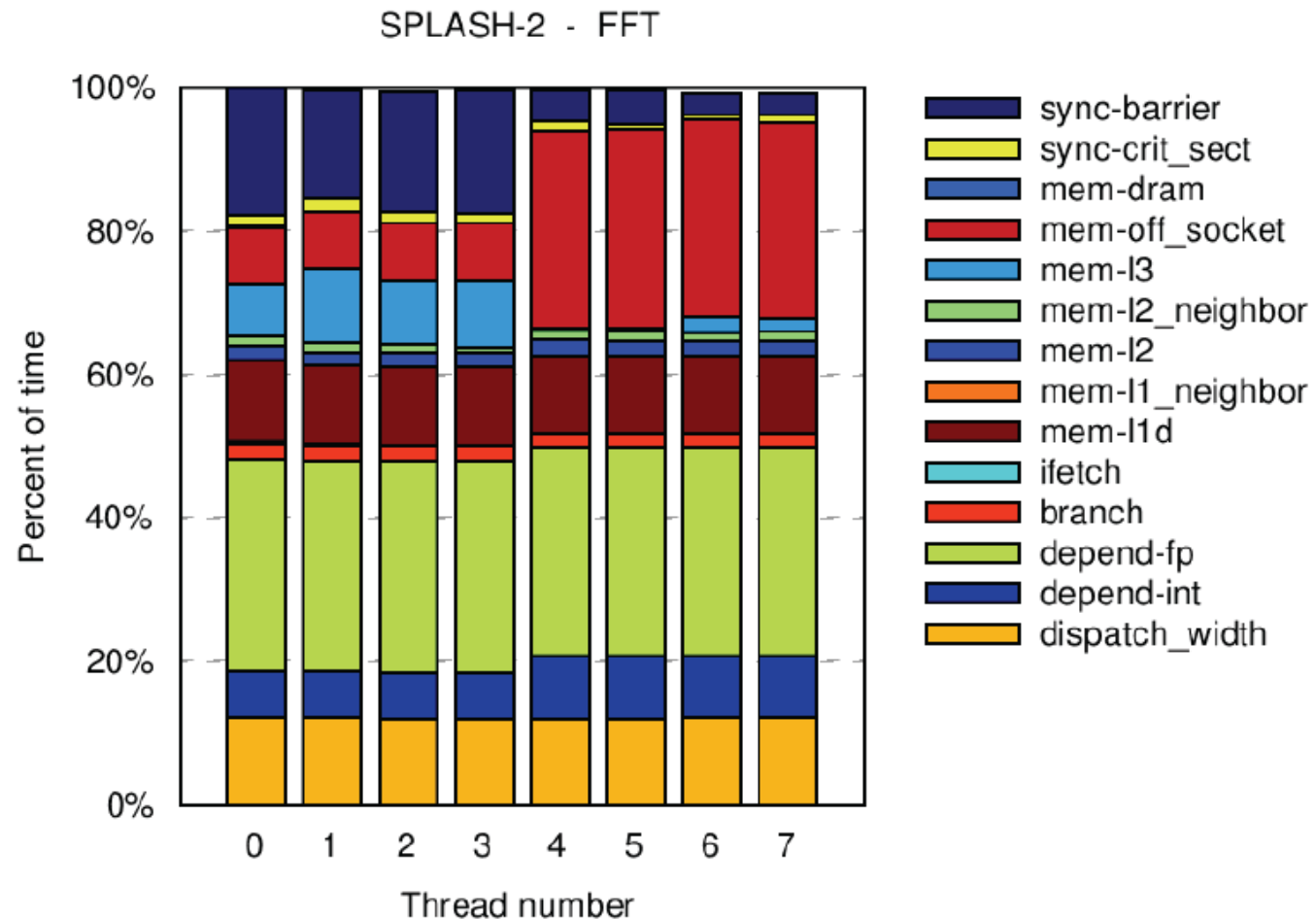
- Interval Model
 - Superscalar OOO, ILP, MLP
- CPI Stacks & Visualization
- Parallel simulation
- X86_64, SSE2 support
- Thread scheduling & migration
- DVFS support
- Modern branch predictors
- Pthreads, OpenMP, TBB, OpenCL, MPI

Sniper : Limitations

- User-level
 - Ιδανικός για HPC
 - Όχι τόσο για workloads με αρκετό OS interaction
- High-abstraction core model
 - Όχι κατάλληλος για τη μοντελοποίηση όλων των συνεπειών εξαιτίας αλλαγών στο επίπεδο του core
 - Ιδανικός για memory subsystem ή NoC
- Μόνο x86

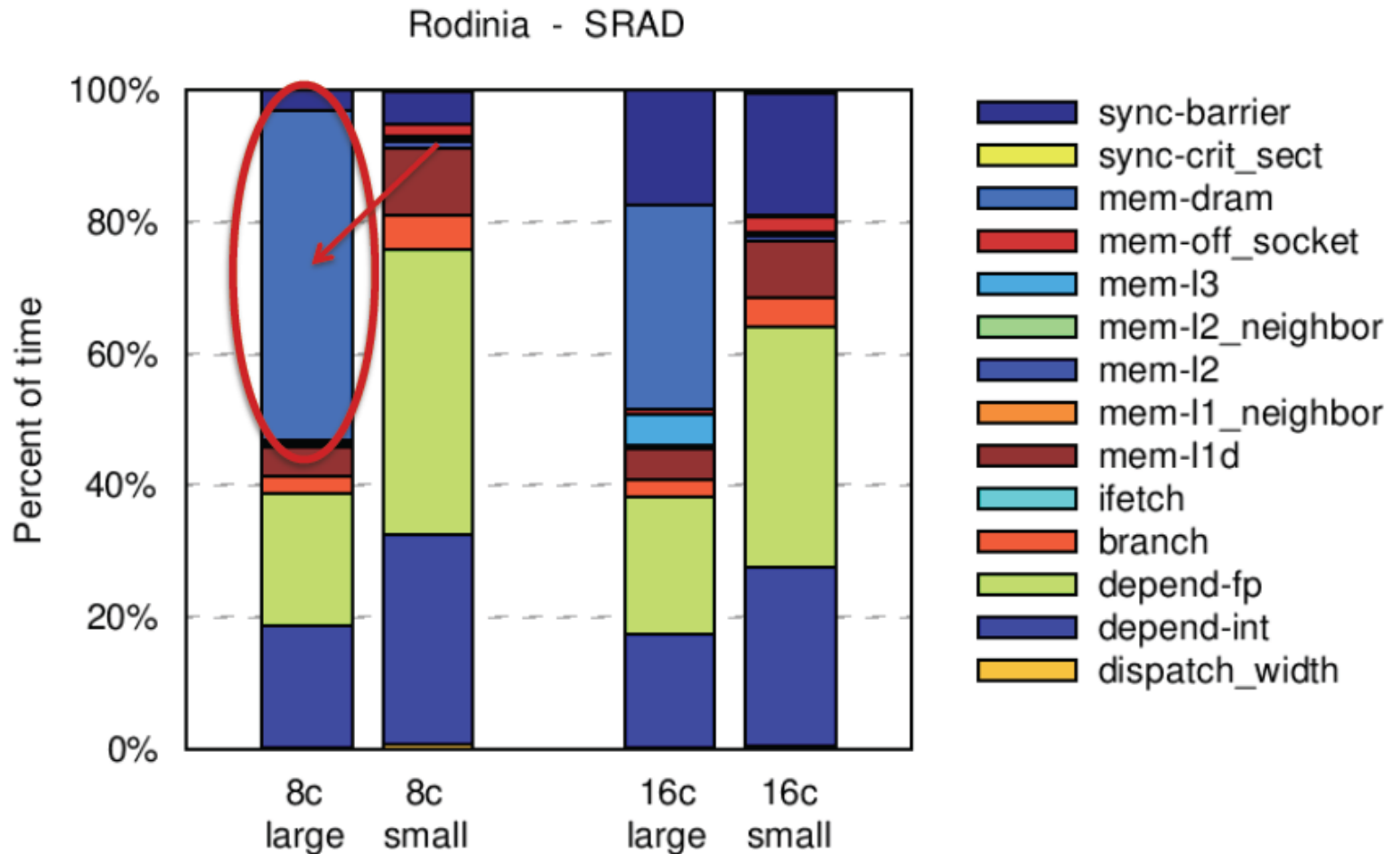
Sniper : Cycle stacks for parallel applications

- Ετερογενείς συμπεριφορές των threads μιας ομογενούς εφαρμογής;



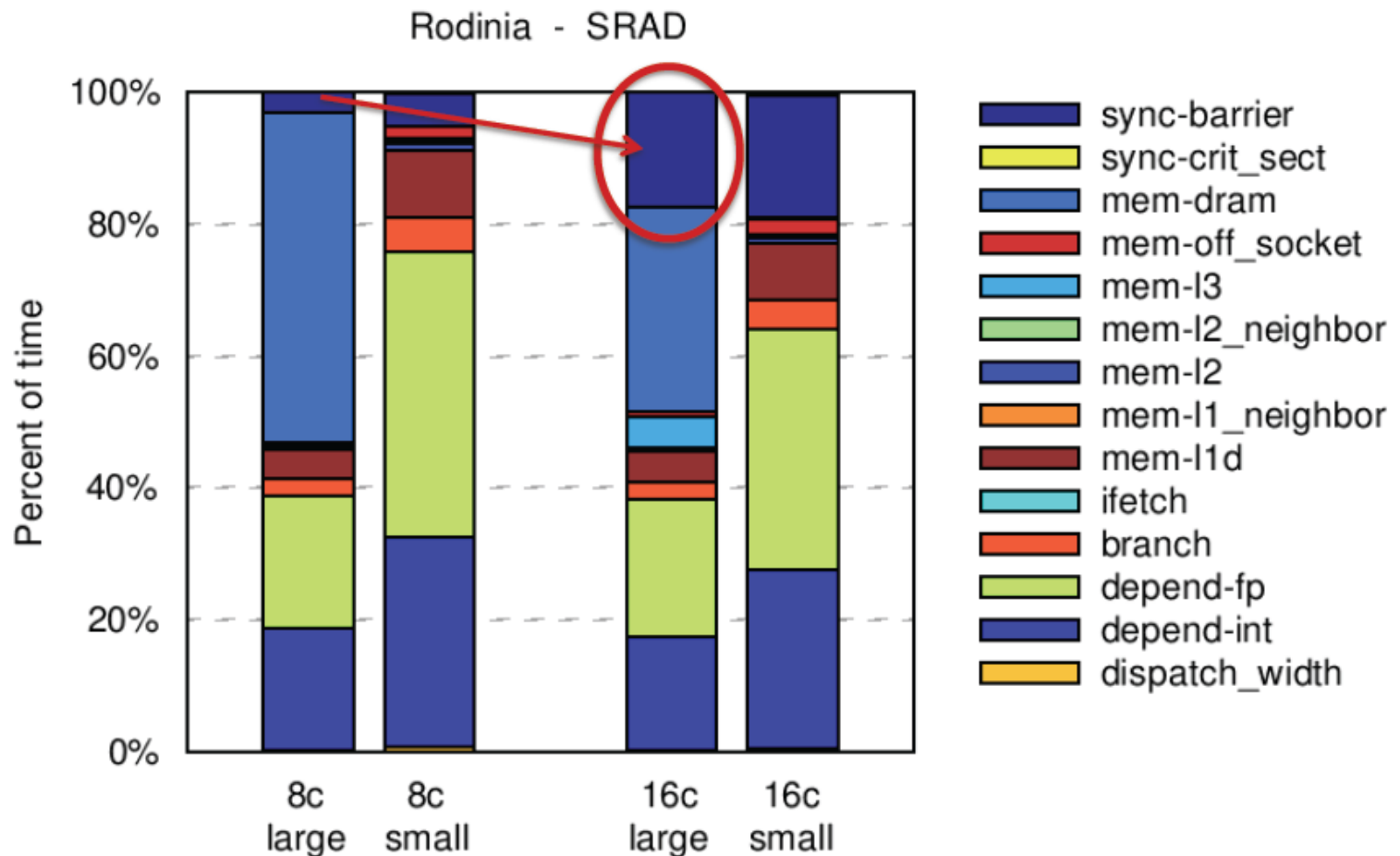
Sniper : Cycle stacks for scaling behavior

- DRAM bound



Sniper : Cycle stacks for scaling behavior

- DRAM bound
- sync increases by 20%



Snipersim Demo