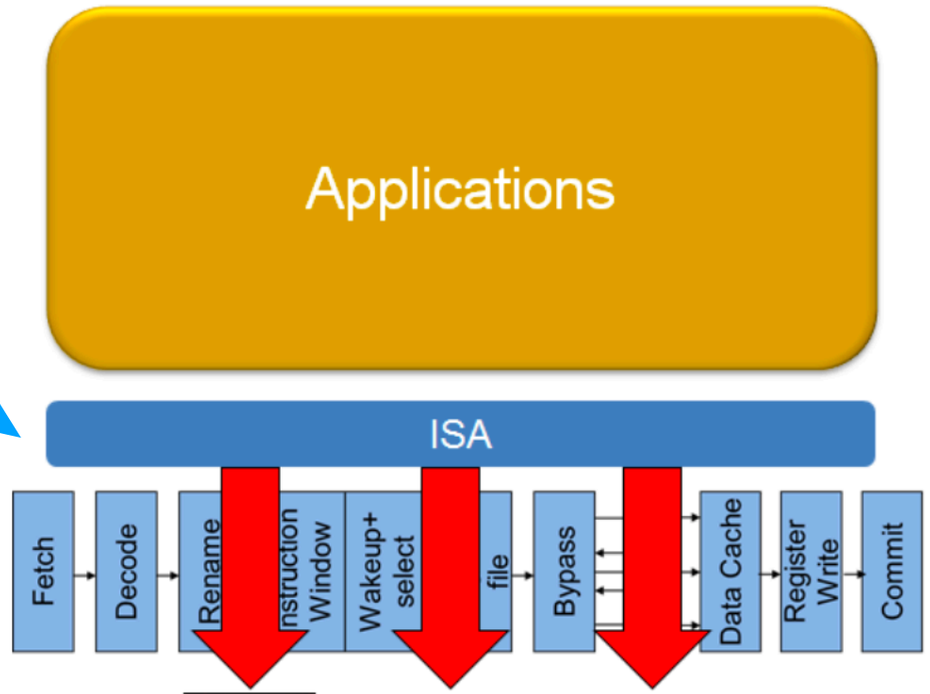
Runtime-Aware Architectures

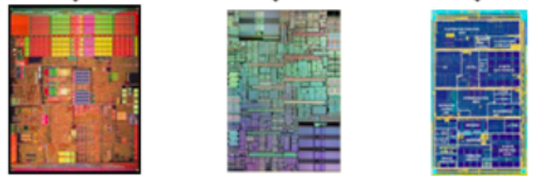
Τεχνολογία Λογισμικού ΣΗΜΜΥ 2017-2018

Αλέξανδρος Καλόμοιρος 03113514

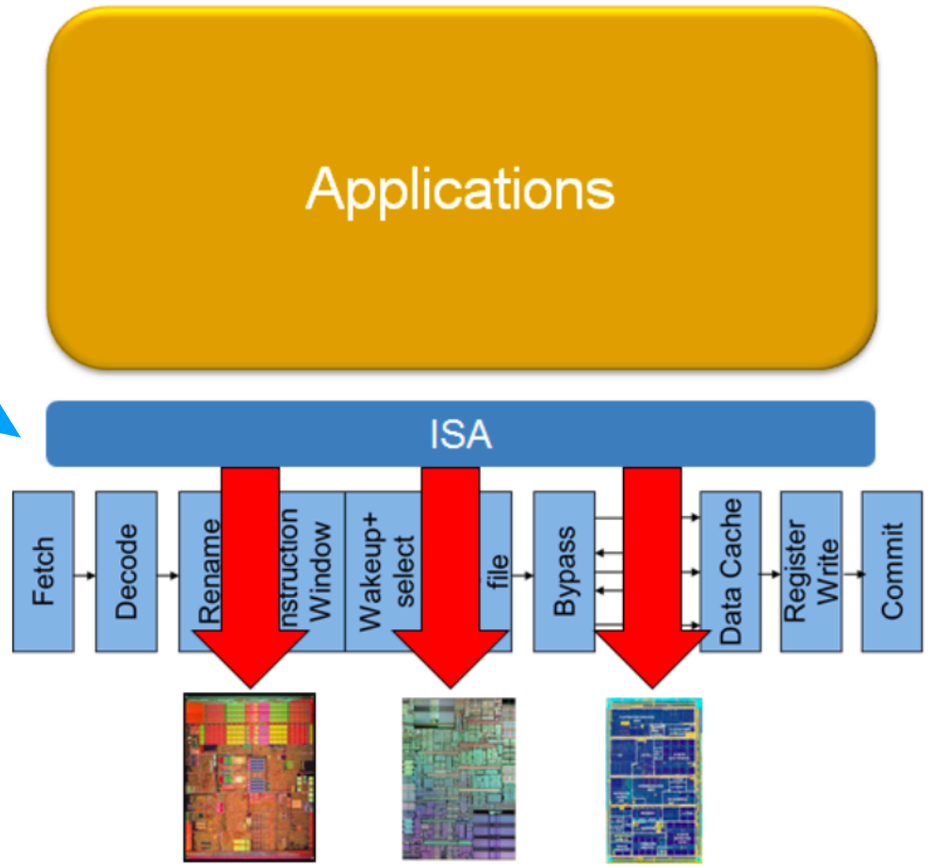
Η χρυσή εποχή των μονοπύρηνων συστημάτων

Διαχωρίζει το λογισμικό και το υλικό



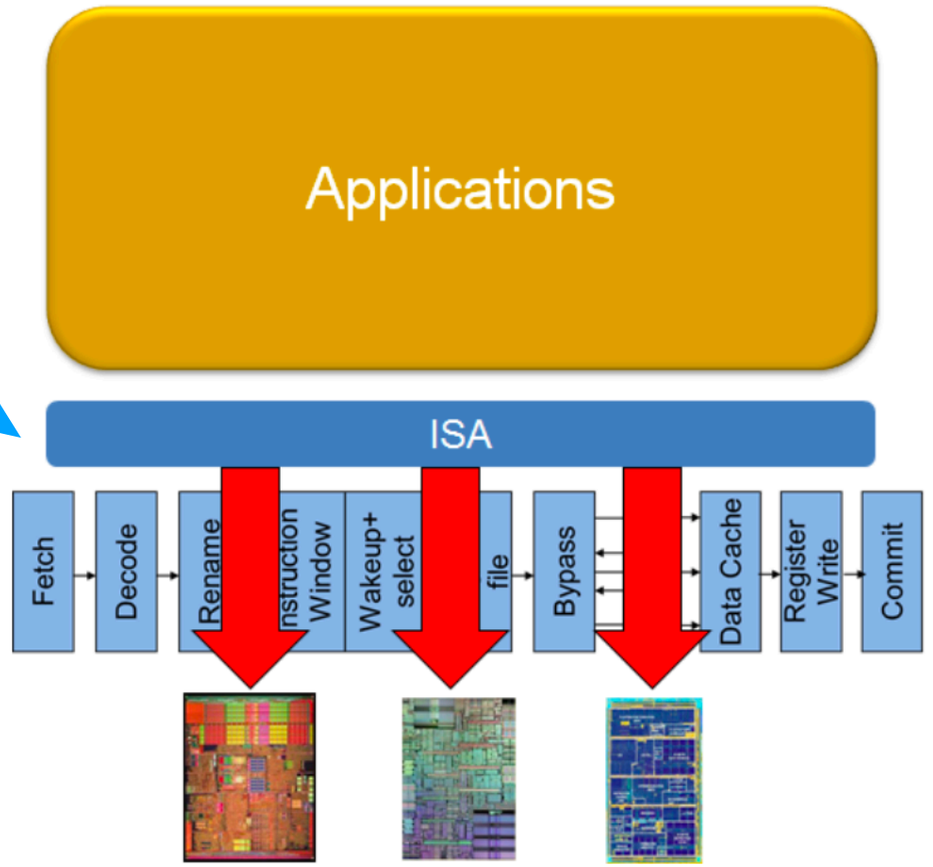


Διαχωρίζει το λογισμικό και το υλικό



Το λογισμικό γράφεται ανεξάρτητα του υλικού

Διαχωρίζει το λογισμικό και το υλικό



Το λογισμικό γράφεται ανεξάρτητα του υλικού

Βελτιώσεις στο επίπεδο του pipeline → βελτίωναν την επίδοση του λογισμικού

Instruction Level Parallelism (ILP)

Οι δύο βασικές προσεγγίσεις για την εκμετάλευση του ILP

Instruction Level Parallelism (ILP)

Οι δύο βασικές προσεγγίσεις για την εκμετάλευση του ILP

- **Superscalar processors**

Instruction Level Parallelism (ILP)

Οι δύο βασικές προσεγγίσεις για την εκμετάλευση του ILP

- **Superscalar processors**
- **Very Long Instruction Word (VLIW) processors**

VLIW processors

Προϋποθέτουν κατά την ώρα μεταγλώττισης

- **ανάλυση εξαρτήσεων μεταξύ των εντολών**
- **και τον χρονοπρογραμματισμό τους**

VLIW processors

Προϋποθέτουν κατά την ώρα μεταγλώττισης

- **ανάλυση εξαρτήσεων μεταξύ των εντολών**
- **και τον χρονοπρογραμματισμό τους**

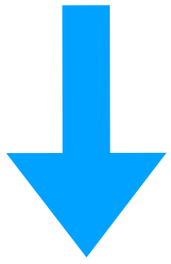
Στη γενική περίπτωση δεν είναι εφικτό να βρεθεί βέλτιστος χρονοπρογραμματισμός

VLIW processors

Προϋποθέτουν κατά την ώρα μεταγλώττισης

- **ανάλυση εξαρτήσεων μεταξύ των εντολών**
- **και τον χρονοπρογραμματισμό τους**

Στη γενική περίπτωση δεν είναι εφικτό να βρεθεί βέλτιστος χρονοπρογραμματισμός

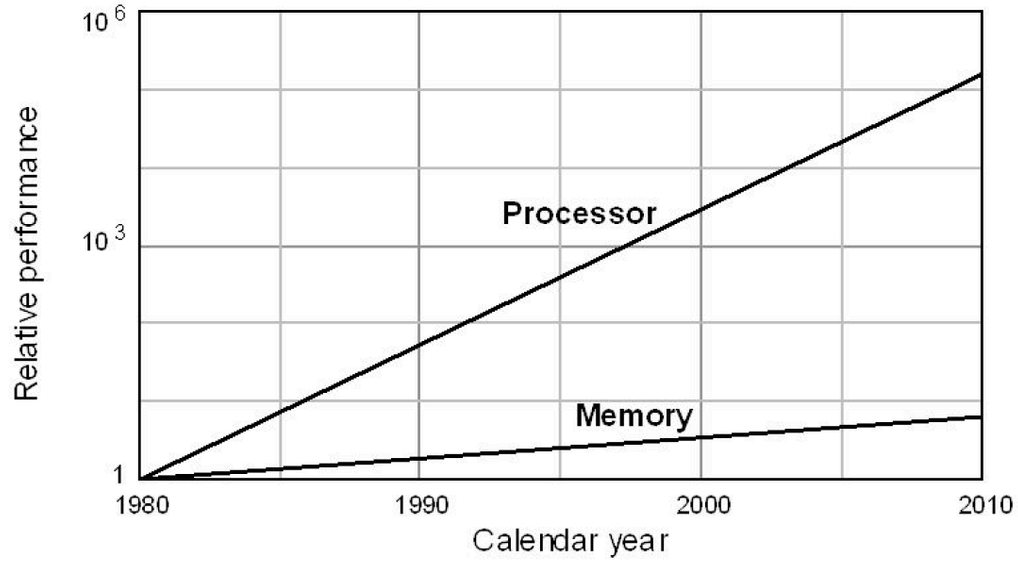


VLIW δεν εκμεταλεύονται στο μέγιστο το ILP

Superscalar processors

Superscalar processors

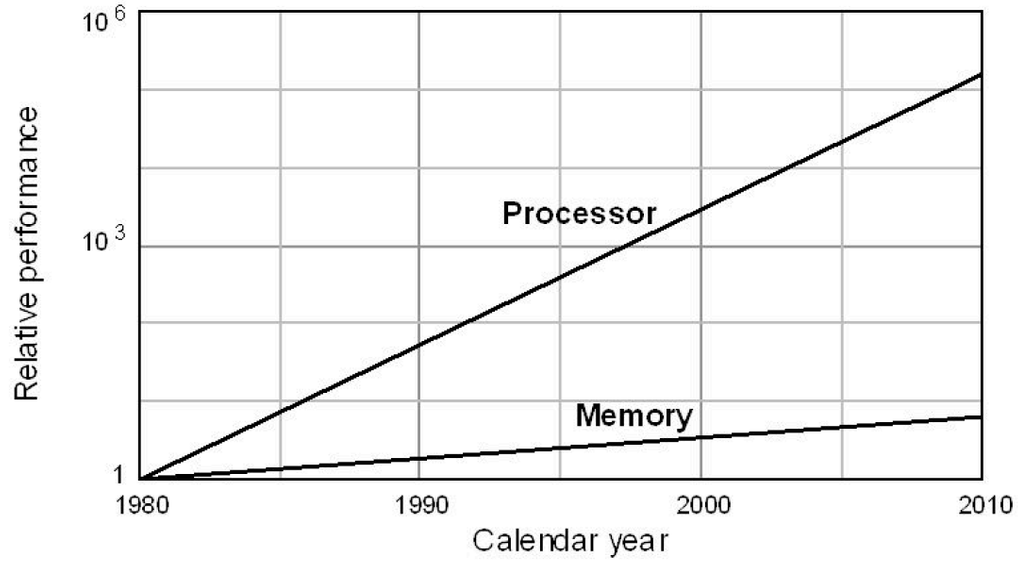
Οι superscalar αρχιτεκτονικές χρησιμοποιούν για να αντιμετωπίσουν το “Memory wall”



Superscalar processors

Οι superscalar αρχιτεκτονικές χρησιμοποιούν για να αντιμετωπίσουν το “Memory wall”

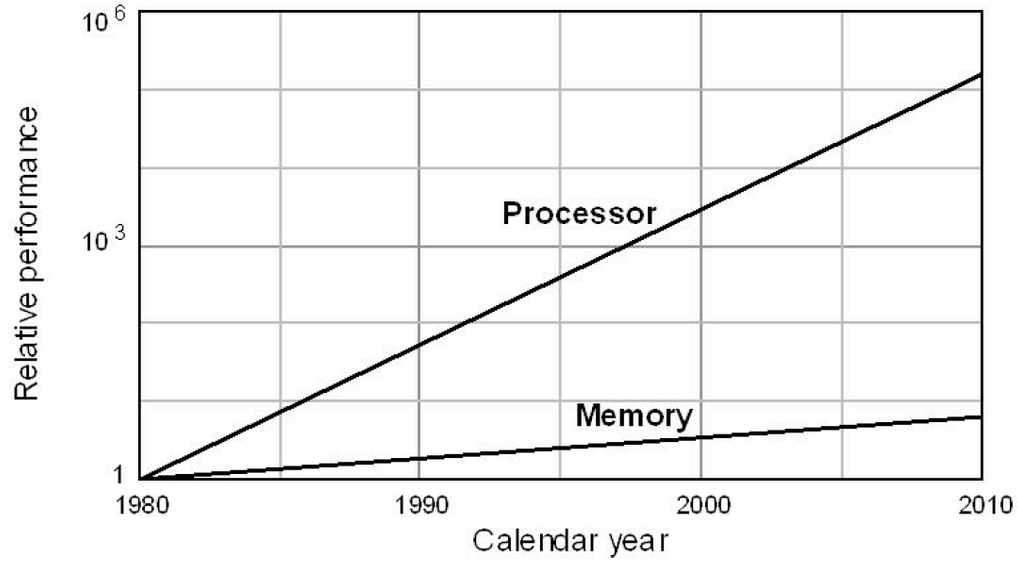
- **Out of Order (OoO)**



Superscalar processors

Οι superscalar αρχιτεκτονικές χρησιμοποιούν για να αντιμετωπίσουν το “Memory wall”

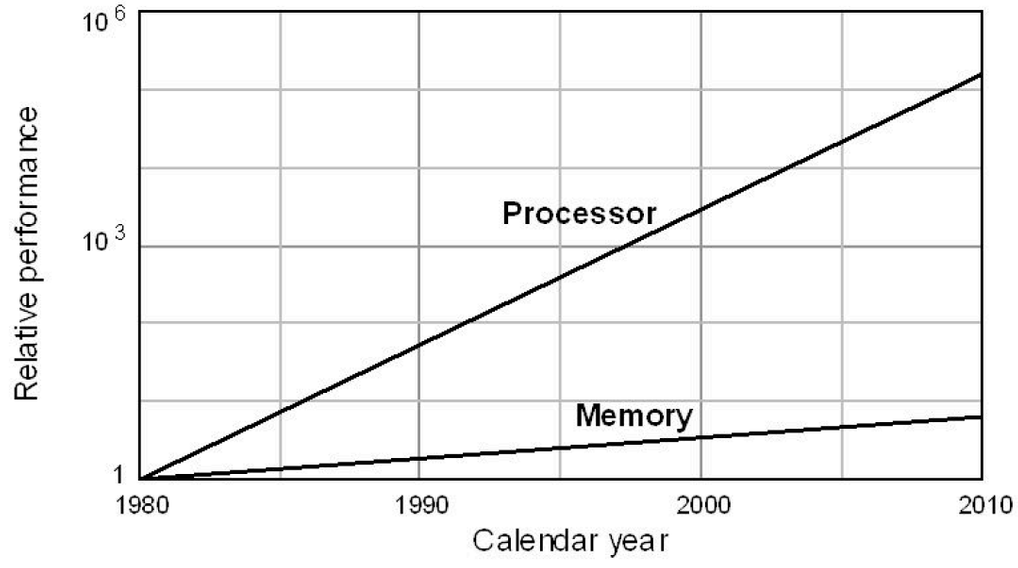
- **Out of Order (OoO)**
- **and speculative execution**



Superscalar processors

Οι superscalar αρχιτεκτονικές χρησιμοποιούν για να αντιμετωπίσουν το “Memory wall”

- **Out of Order (OoO)**
- **and speculative execution**



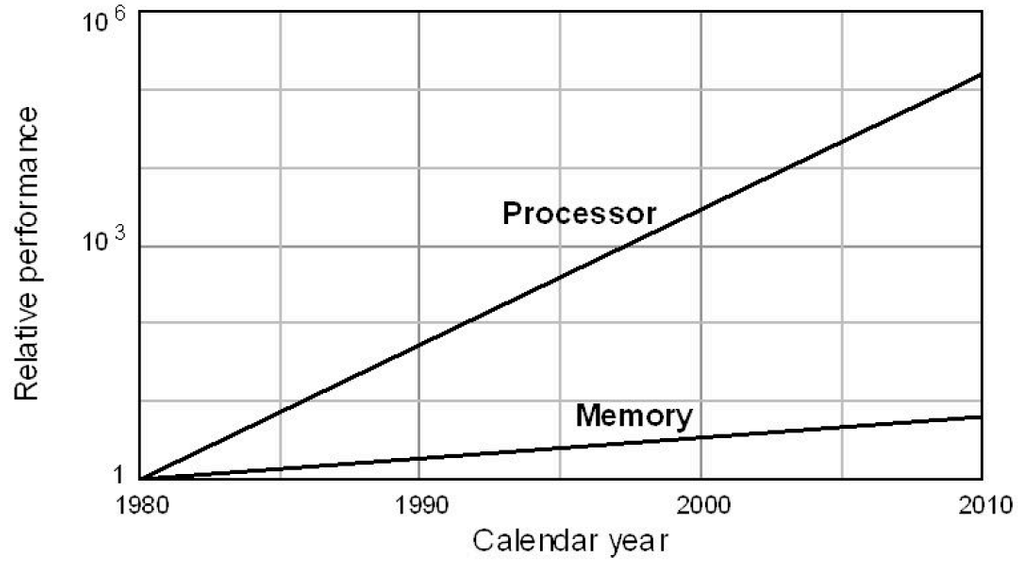
Ακόμα χρησιμοποιούν τεχνικές όπως

- **Prefetching**

Superscalar processors

Οι superscalar αρχιτεκτονικές χρησιμοποιούν για να αντιμετωπίσουν το “Memory wall”

- **Out of Order (OoO)**
- **and speculative execution**



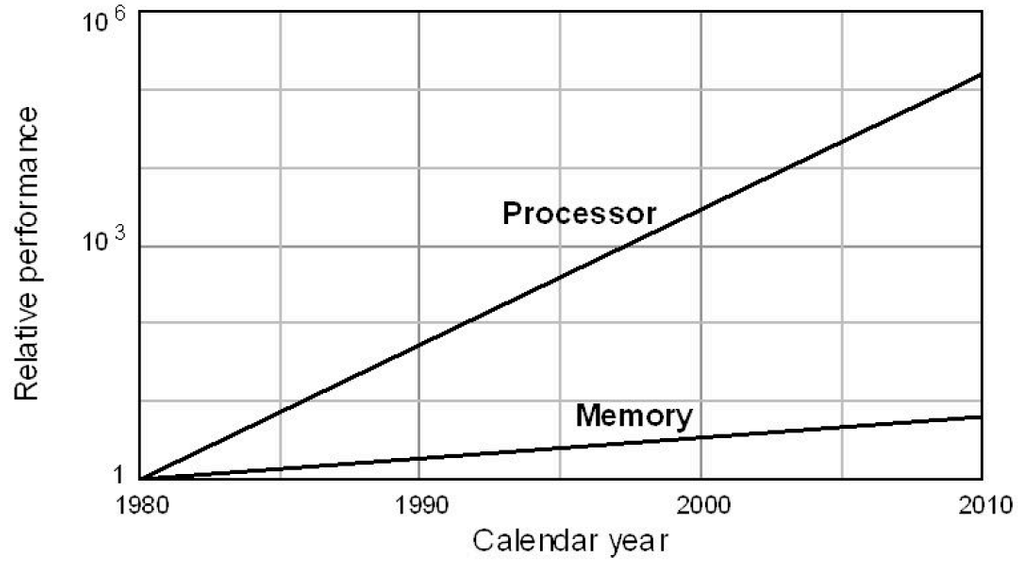
Ακόμα χρησιμοποιούν τεχνικές όπως

- **Prefetching**
- **Deep memory hierarchies**

Superscalar processors

Οι superscalar αρχιτεκτονικές χρησιμοποιούν για να αντιμετωπίσουν το “Memory wall”

- **Out of Order (OoO)**
- **and speculative execution**



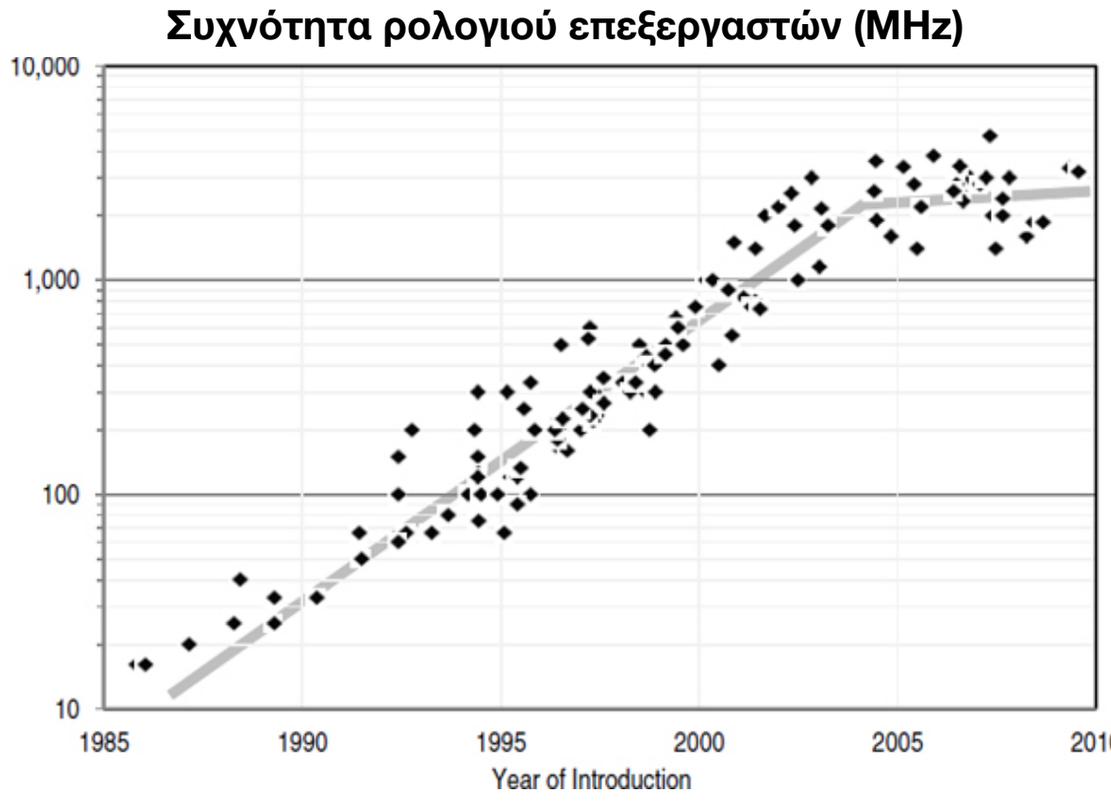
Ακόμα χρησιμοποιούν τεχνικές όπως

- **Prefetching**
- **Deep memory hierarchies**
- **And large reorder buffers**

Τα τελευταία χρόνια, οι παραδοσιακοί τρόποι, για να αυξήσουμε την επίδοση του hardware, σύμφωνα με τις προβλέψεις του νόμου του Moore, εξαφανίστηκαν.

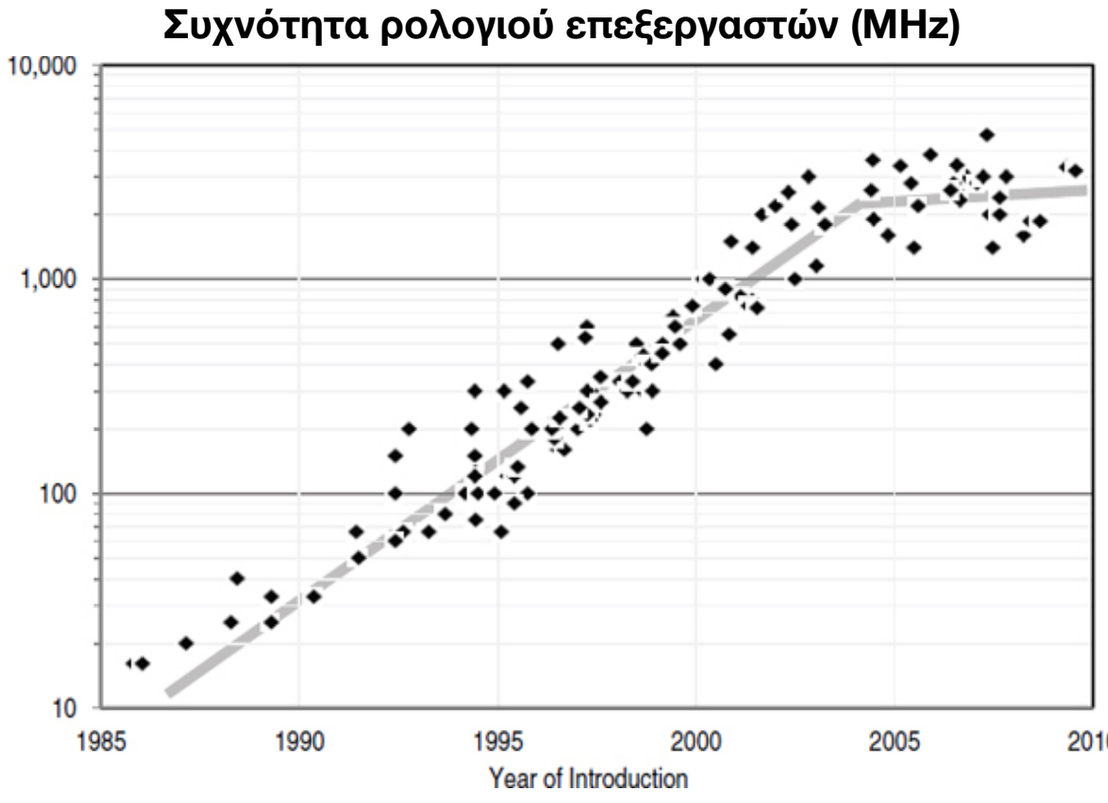
Τα τελευταία χρόνια, οι παραδοσιακοί τρόποι, για να αυξήσουμε την επίδοση του hardware, σύμφωνα με τις προβλέψεις του νόμου του Moore, εξαφανίστηκαν.

Όταν η συχνότητα ρολογιού του επεξεργαστή περάσει ένα όριο, η ισχύς ανά μονάδα επιφάνειας (πυκνότητα ισχύος) δεν μπορεί να απομακρυνθεί.



Τα τελευταία χρόνια, οι παραδοσιακοί τρόποι, για να αυξήσουμε την επίδοση του hardware, σύμφωνα με τις προβλέψεις του νόμου του Moore, εξαφανίστηκαν.

Όταν η συχνότητα ρολογιού του επεξεργαστή περάσει ένα όριο, η ισχύς ανά μονάδα επιφάνειας (πυκνότητα ισχύος) δεν μπορεί να απομακρυνθεί.



Η συχνότητα των ρολογιών των επεξεργαστών σταθεροποιήθηκε

Power Wall + Memory Wall + ILP Wall = Free lunch is over

(ILP Wall : ασήμαντες απολαβές από επιπλέον ILP HW)

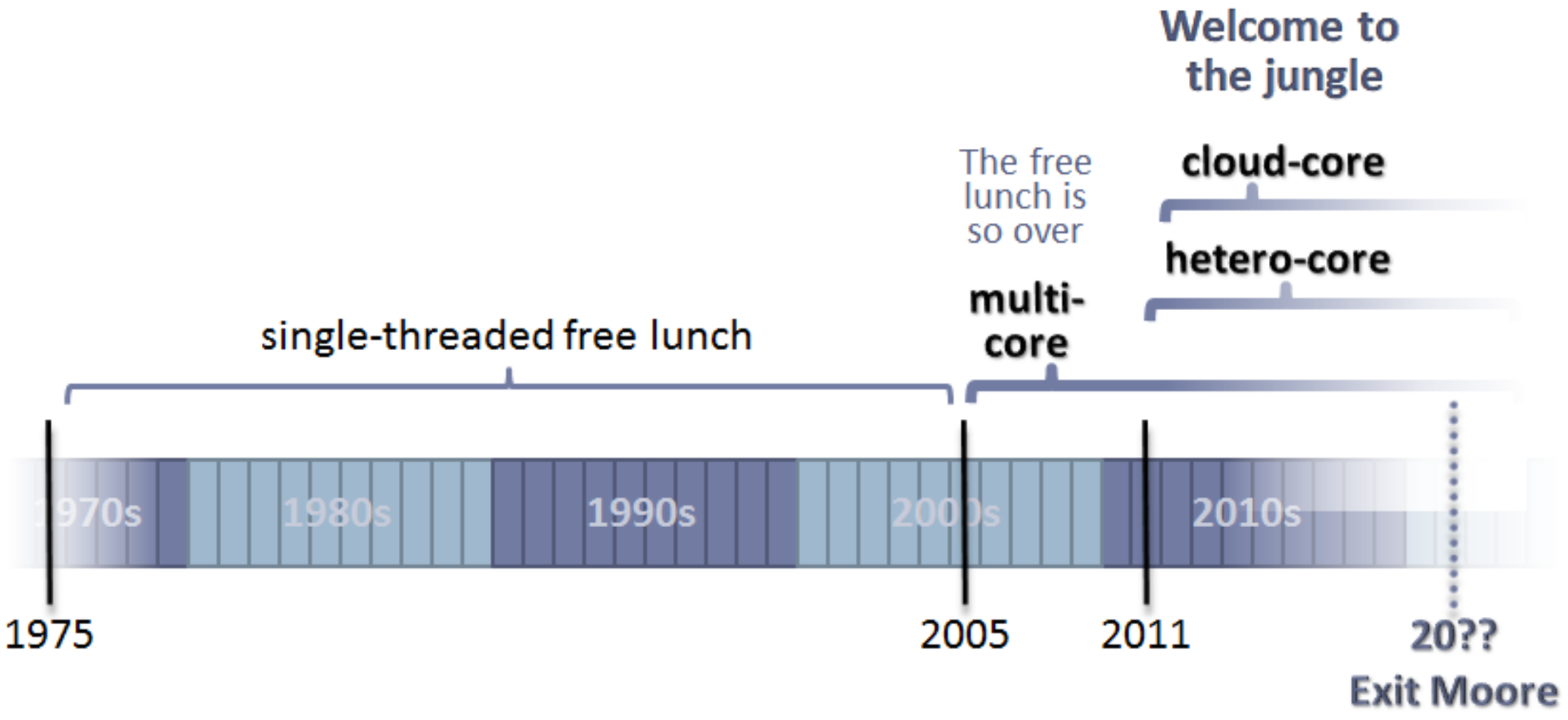
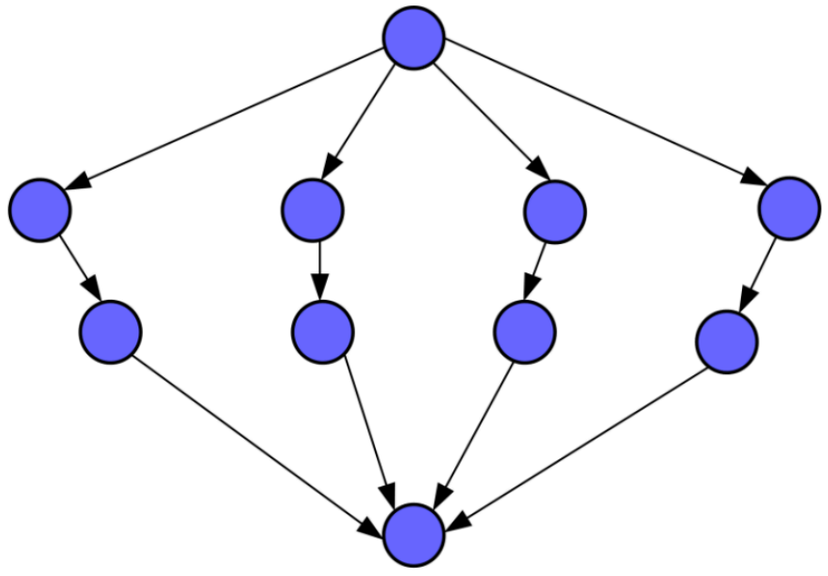


image by Herb Sutter

Multicore processors

- **Μπορούν να εκμεταλευτούν παραλληλία σε επίπεδο task (TLP)**



Αλλά…..

Multicore processors

Οι λόγοι

- **Cache size / operation**
- **Memory bandwidth / operation**

μένουν σταθεροί ή μειώνονται στις πολυπύρηνες αρχιτεκτονικές, κάνοντας αρκετά δύσκολη την πλήρη εκμετάλευση του throughput που έχουν οι πολυπύρηνες αρχιτεκτονικές

- Το memory wall μεγάλωσε

Multicore processors

Οι λόγοι

- **Cache size / operation**
- **Memory bandwidth / operation**

μένουν σταθεροί ή μειώνονται στις πολυπύρηνες αρχιτεκτονικές, κάνοντας αρκετά δύσκολη την πλήρη εκμετάλευση του throughput που έχουν οι πολυπύρηνες αρχιτεκτονικές

- Το memory wall μεγάλωσε

Ακόμα η κατανάλωση ισχύος, αν συνεχίσει να αυξάνεται με τους τωρινούς ρυθμούς, το όνειρο για exascale supercomputers και petaflop mobiles φαίνεται ακατόρθωτο.

Περιορισμοί σχετικά με την κατανάλωση ισχύος, αποτελούν ένα νέο power wall.

Multicore processors

Επιπλέον πολυπύρηνα συστήματα μπορεί να έχουν

Multicore processors

Επιπλέον πολυπύρηνα συστήματα μπορεί να έχουν

- **Ετερογενής επεξεργαστές**

Multicore processors

Επιπλέον πολυπύρηνα συστήματα μπορεί να έχουν

- **Ετερογενής επεξεργαστές**
- **Με διαφορετικά ISA**

Multicore processors

Επιπλέον πολυπύρηνα συστήματα μπορεί να έχουν

- **Ετερογενής επεξεργαστές**
- **Με διαφορετικά ISA**
- **Συνδεδεμένους μέσω πολλαπλών επιπέδων μοιραζόμενων resources,**

Multicore processors

Επιπλέον πολυπύρηνα συστήματα μπορεί να έχουν

- **Ετερογενής επεξεργαστές**
- **Με διαφορετικά ISA**
- **Συνδεδεμένους μέσω πολλαπλών επιπέδων μοιραζόμενων resources,**
- **Με διαφορετικούς χρόνους πρόσβασης**

Multicore processors

Επιπλέον πολυπύρηνα συστήματα μπορεί να έχουν

- **Ετερογενής επεξεργαστές**
- **Με διαφορετικά ISA**
- **Συνδεδεμένους μέσω πολλαπλών επιπέδων μοιραζόμενων resources,**
- **Με διαφορετικούς χρόνους πρόσβασης**
- **Και κατανεμημένες περιοχές μνήμης**

Multicore processors

Επιπλέον πολυπύρηνα συστήματα μπορεί να έχουν

- **Ετερογενής επεξεργαστές**
- **Με διαφορετικά ISA**
- **Συνδεδεμένους μέσω πολλαπλών επιπέδων μοιραζόμενων resources,**
- **Με διαφορετικούς χρόνους πρόσβασης**
- **Και κατανεμημένες περιοχές μνήμης**

Όλα αυτά κάνουν την σωστή διαχείρηση των δεδομένων αρκετά μεγάλη πρόκληση, γνωστή και ως Programmability Wall.

Multicore processors

Reliability Wall

Multicore processors

Reliability Wall

Σε επίπεδο Λογισμικού

Η αδυναμία των παράλληλων προγραμμάτων να κλιμακώσουν καλά, αν σε αυτά εισάγουμε μηχανισμούς αξιοπιστίας (πχ checkpointing)

Multicore processors

Reliability Wall

Σε επίπεδο Λογισμικού

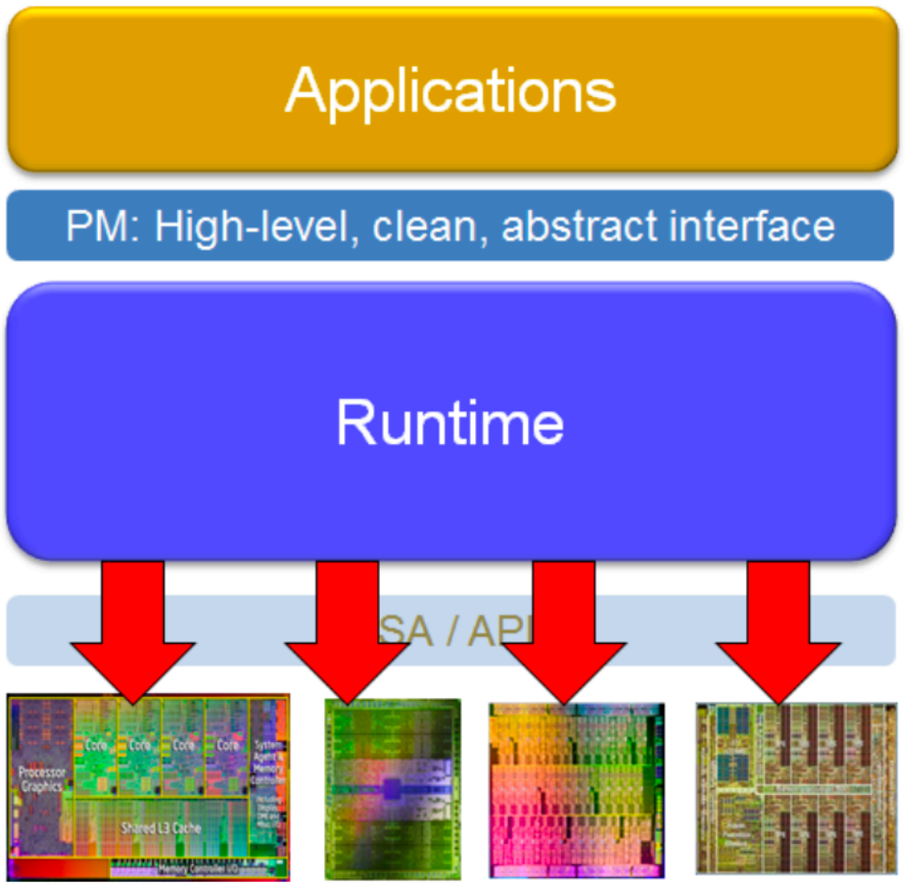
Η αδυναμία των παράλληλων προγραμμάτων να κλιμακώσουν καλά, αν σε αυτά εισάγουμε μηχανισμούς αξιοπιστίας (πχ checkpointing)

Σε επίπεδο Υλικού

Το επιπλέον κόστος για την υλοποίηση μηχανισμών αξιοπιστίας, όπως (Error Correction Codes), αλλά και το επιπλέον κόστος που σχετίζεται με την αποθήκευση της έξτρα πληροφορίας που χρειάζονται οι μηχανισμοί αυτοί.

Πώς προχωράμε από δω και πέρα;

Ο μόνος τρόπος να κρατήσουμε την συνεχώς αυξανόμενη δυσκολία προγραμματισμού σε λογικά πλαίσια, είναι η συνεργασία του ετερογενούς παράλληλου υλικού με το σύστημα χρόνου εκτέλεσης.



Runtime-Aware Architectures

Το runtime παρέχει πλέον την αφαίρεση από το πολύπλοκο υλικό, ενώ αυτό σχεδιάζεται λαμβάνοντας υπόψιν του το runtime, προκειμένου να αξιοποιηθεί η γνώση που έχει το runtime για τα διάφορα tasks.

Runtime-Aware Architectures

Το runtime παρέχει πλέον την αφαίρεση από το πολύπλοκο υλικό, ενώ αυτό σχεδιάζεται λαμβάνοντας υπόψιν του το runtime, προκειμένου να αξιοποιηθεί η γνώση που έχει το runtime για τα διάφορα tasks.

Η ιδέα είναι να έχουμε task-based representations των παράλληλων προγραμμάτων και να χειριζόμαστε τα διάφορα tasks με τον ίδιο τρόπο που οι superscalar επεξεργαστές χειρίζονται τον ILP, καθώς τα tasks έχουν εξαρτήσεις μεταξύ τους και ο γράφος των εξαρτήσεων μπορεί να δημιουργηθεί δυναμικά κατά την ώρα εκτέλεσης ή στατικά. Το υλικό λοιπόν πρέπει να υποστιρίζει τις ενέργειες του runtime.

Σύγκριση οπτικής superscalar και runtime

Superscalar	**Task-based Runtime**
Instructions	Tasks
Functional Units	Cores
Fetch and Decode Units	Cores
Registers (name space)	Main Memory
Registers (storage)	Local Memory
Out-of-Order Execution	
Pipelined Execution	
Speculative Execution	

Σύγκριση οπτικής superscalar και runtime

Superscalar	Task-based Runtime
Instructions	Tasks
Functional Units	Cores
Fetch and Decode Units	Cores
Registers (name space)	Main Memory
Registers (storage)	Local Memory
Out-of-Order Execution	
Pipelined Execution	
Speculative Execution	

Ο προγραμματιστής όταν ορίζει τα tasks, ορίζει και το input, output του task

Σύγκριση οπτικής superscalar και runtime

Superscalar	Task-based Runtime
Instructions	Tasks
Functional Units	Cores
Fetch and Decode Units	Cores
Registers (name space)	Main Memory
Registers (storage)	Local Memory
Out-of-Order Execution	
Pipelined Execution	
Speculative Execution	

Ο προγραμματιστής όταν ορίζει τα tasks, ορίζει και το input, output του task

Η πληροφορία αυτή χρησιμοποιείται για την παραγωγή του γράφου των εξαρτήσεων

Σύγκριση οπτικής superscalar και runtime

Superscalar	Task-based Runtime
Instructions	Tasks
Functional Units	Cores
Fetch and Decode Units	Cores
Registers (name space)	Main Memory
Registers (storage)	Local Memory
Out-of-Order Execution	
Pipelined Execution	
Speculative Execution	

Ο προγραμματιστής όταν ορίζει τα tasks, ορίζει και το input, output του task

Η πληροφορία αυτή χρησιμοποιείται για την παραγωγή του γράφου των εξαρτήσεων

Ο οποίος αποτελεί την βάση για την στενή συνεργασία με το υλικό, για να γίνεται χρονοπρογραμματισμός των tasks και να αξιοποιηθούν δυνατότητες διαχείρισης της μετακίνησης των δεδομένων στην αρχιτεκτονική.

Memory Wall

Prefetching: Στις RAAs, μπορούμε να αξιοποιήσουμε την γνώση του runtime για μελλοντικές μεταφορές δεδομένων, που βρίσκεται κωδικοποιημένη στο γράφο των εξαρτήσεων. Μαζί με κατάλληλη υποστήριξη από το υλικό για να χειριστούμε αυτές τις μεταφορές, μπορούμε να συνδιάσουμε data movements με computation.

Memory Wall

Prefetching: Στις RAAs, μπορούμε να αξιοποιήσουμε την γνώση του runtime για μελλοντικές μεταφορές δεδομένων, που βρίσκεται κωδικοποιημένη στο γράφο των εξαρτήσεων. Μαζί με κατάλληλη υποστήριξη από το υλικό για να χειριστούμε αυτές τις μεταφορές, μπορούμε να συνδιάσουμε data movements με computation.

Η ελαχιστοποίηση της μεταφοράς δεδομένων είναι πολύ συμαντική για την τελική επίδοση του συστήματος. Η αξιοποίηση της τοπικότητας μέσω επαναχρησιμοποίησης δεδομένων ή prefetching δεν είναι αρκετή για τα μελλοντικά multi-core συστήματα.

Memory Wall

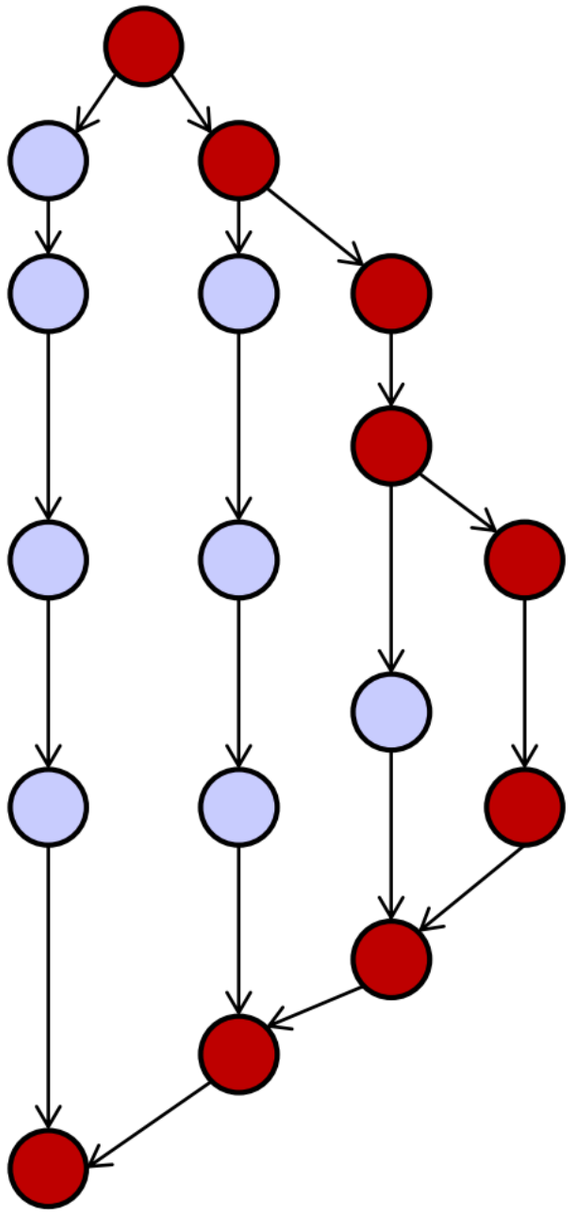
Prefetching: Στις RAAs, μπορούμε να αξιοποιήσουμε την γνώση του runtime για μελλοντικές μεταφορές δεδομένων, που βρίσκεται κωδικοποιημένη στο γράφο των εξαρτήσεων. Μαζί με κατάλληλη υποστήριξη από το υλικό για να χειριστούμε αυτές τις μεταφορές, μπορούμε να συνδιάσουμε data movements με computation.

Η ελαχιστοποίηση της μεταφοράς δεδομένων είναι πολύ συμαντική για την τελική επίδοση του συστήματος. Η αξιοποίηση της τοπικότητας μέσω επαναχρησιμοποίησης δεδομένων ή prefetching δεν είναι αρκετή για τα μελλοντικά multi-core συστήματα.

Απλοποιημένα πρωτόκολλα συνάφειας της κρυφής μνήμης οδηγούμενα από το runtime system, μπορούν να μειώσουν την κίνηση για τη διατήρηση της συνάφειας.

Power Wall

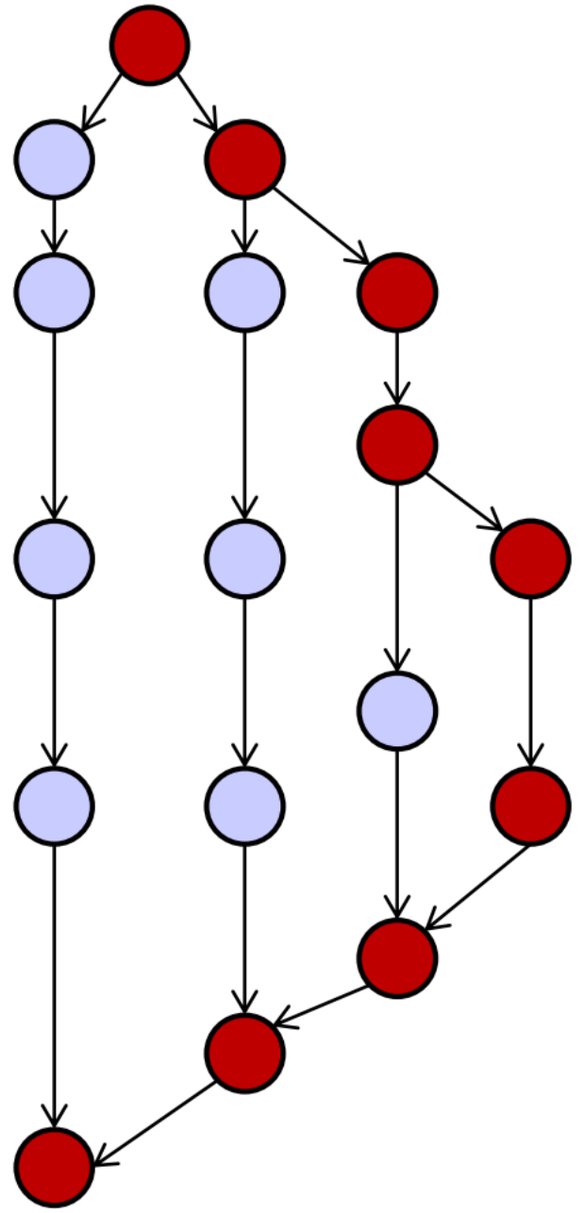
Η τοποθέτηση tasks που βρίσκονται πάνω στο κρίσιμο μονοπάτι, σε γρήγορα cores, είναι εξαιρετικά σημαντική για τελική επίδοση.



Power Wall

Η τοποθέτηση tasks που βρίσκονται πάνω στο κρίσιμο μονοπάτι, σε γρήγορα cores, είναι εξαιρετικά σημαντική για τελική επίδοση.

Ακόμα tasks που δεν βρίσκονται πάνω στο κρίσιμο μονοπάτι είναι latency tolerant, σε κάποιο βαθμό, και μπορούν να χρονοδρομολογηθούν σε πιο αργούς, και energy efficient, επεξεργαστές, καθώς και memory accesses από αυτά τα tasks μπορούν να πραγματοποιηθούν από μνήμες υψηλότερης καθυστέρησης, και μικρότερης κατανάλωσης ισχύος.



Reliability Wall

Η μείωση της μετάδοσης λαθών είναι σημαντική για την αντιμετώπιση του reliability wall.

Reliability Wall

Η μείωση της μετάδοσης λαθών είναι σημαντική για την αντιμετώπιση του reliability wall.

Το υλικό μπορεί να παρέχει υποστήριξη

- **παρέχοντας την δυνατότητα γρήγορων επαναϋπολογισμών, χρησιμοποιώντας δεδομένα που βρίσκονται ήδη στην cache**

Reliability Wall

Η μείωση της μετάδοσης λαθών είναι σημαντική για την αντιμετώπιση του reliability wall.

Το υλικό μπορεί να παρέχει υποστήριξη

- **παρέχοντας την δυνατότητα γρήγορων επαναϋπολογισμών, χρησιμοποιώντας δεδομένα που βρίσκονται ήδη στην cache**
- **υποστηρίζοντας γρήγορες μεταφορές δεδομένων**

Reliability Wall

Η μείωση της μετάδοσης λαθών είναι σημαντική για την αντιμετώπιση του reliability wall.

Το υλικό μπορεί να παρέχει υποστήριξη

- **παρέχοντας την δυνατότητα γρήγορων επαναϋπολογισμών, χρησιμοποιώντας δεδομένα που βρίσκονται ήδη στην cache**
- **υποστηρίζοντας γρήγορες μεταφορές δεδομένων**
- **έχοντας μερικά cores, αποκλειστικά για να τρέχουν αλγοριθμικούς ελέγχους ανίχνευσης και διόρθωσης διεφθαρμένων δεδομένων**

Programmability Wall

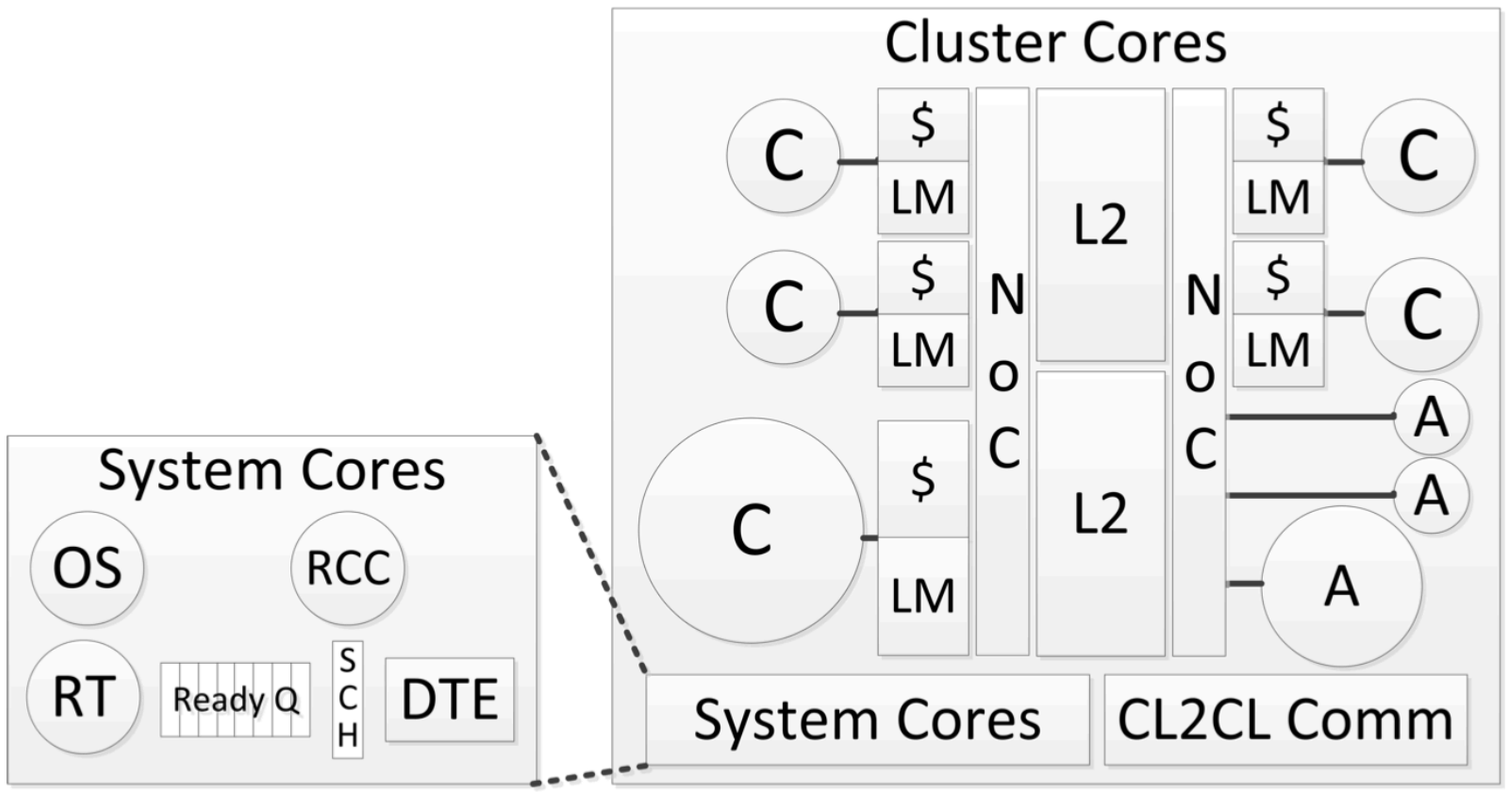
Στα τωρινά runtime systems, το granularity των tasks πρέπει να είναι αρκετά μεγάλο, προκειμένου να μπορούμε να αγνοήσουμε το overhead του runtime. Σαν αποτέλεσμα η ελάχιστη διάρκεια ενός task πρέπει να είναι της τάξης δεκάδες-εκατοντάδες milliseconds.

Programmability Wall

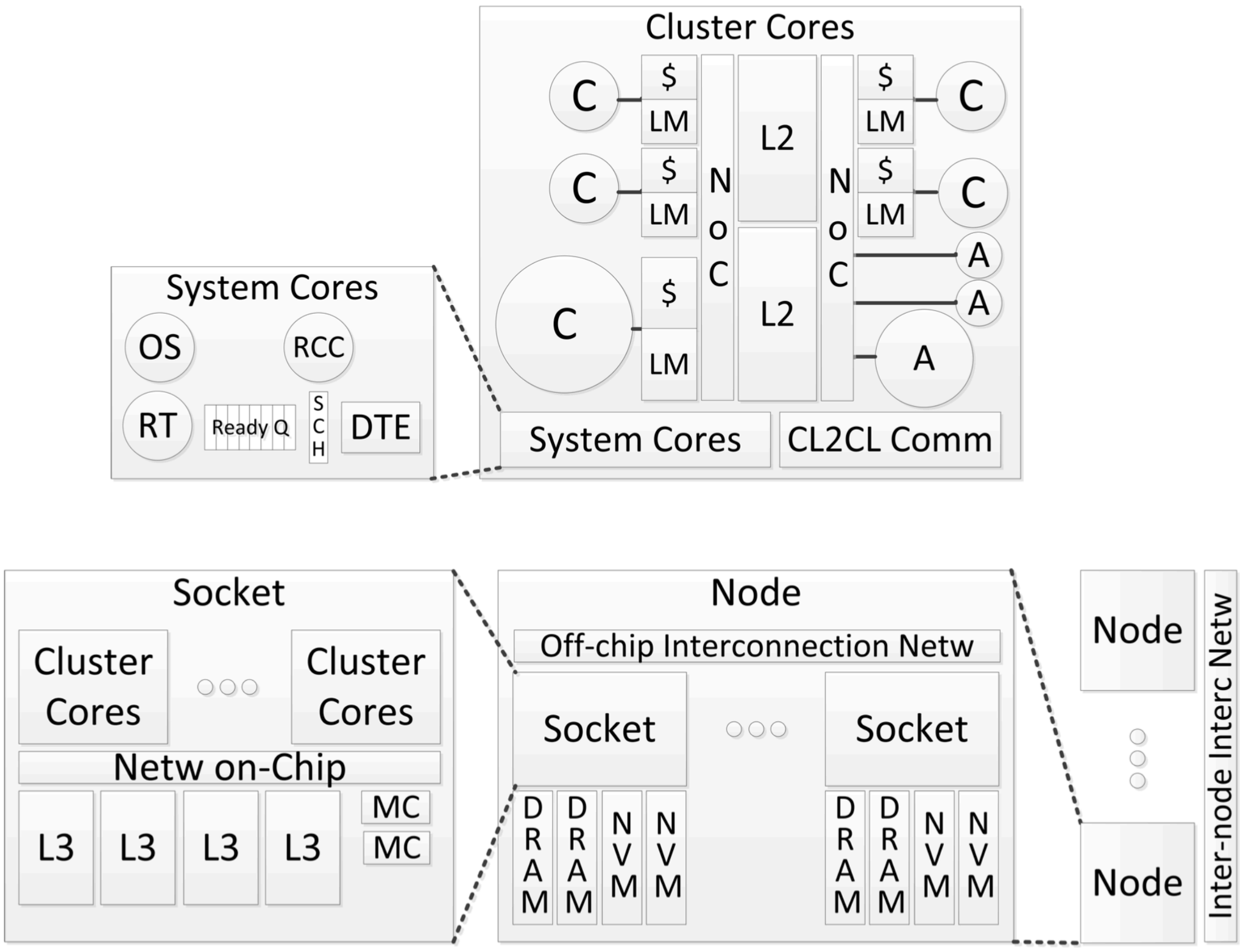
Στα τωρινά runtime systems, το granularity των tasks πρέπει να είναι αρκετά μεγάλο, προκειμένου να μπορούμε να αγνοήσουμε το overhead του runtime. Σαν αποτέλεσμα η ελάχιστη διάρκεια ενός task πρέπει να είναι της τάξης δεκάδες-εκατοντάδες milliseconds.

Κάποιοι αλγόριθμοι χρειάζονται μικρότερο granularity για να περιγράψουν την παραλληλία που έχουν. Για το λόγο αυτό χρειάζεται υποστήριξη από το υλικό για να επιταχυνθούν εργασίες του runtime, όπως η κατασκευή του γράφου των εξαρτήσεων.

Γενικό σχήμα συστατικών RAA



Γενικό σχήμα συστατικών RAA



References

1. Runtime-Aware Architectures: A First Approach (2014)

Mateo Valero, Miquel Moreto, Marc Casas, Eduard Ayguade, Jesus Labarta

https://pdfs.semanticscholar.org/cfd3/4380711f505e58289a524e6d154dc44355a1.pdf

2. Runtime-Aware Architectures (2015)

Marc Casas, Miquel Moreto, Lluc Alvarez, Emilio Castillo, Dimitrios Chasapis, Timothy Hayes, Luc Jaulmes, Oscar Palomar, Osman Unsal, Adrian Cristal, Eduard Ayguade, Jesus Labarta and Mateo Calero

http://www.springer.com/cda/content/document/cda_downloaddocument/9783662480953-c2.pdf?SGWID=0-0-45-1518982-p177634027

3.The Reliability Wall for Exascale Supercomputing (2012)

Zhiyuan Wang, Yun Zhou, Jingling Xue

https://www.researchgate.net/publication/254058691_The_Reliability_Wall_for_Exascale_Supercomputing