

# Σχεδιασμός και υλοποίηση μηχανισμών pipe και fork σε unikernels

Μάινας Χαράλαμπος

Επιβλέπων: Γεώργιος Γκούμας

Εργαστήριο Υπολογιστικών Συστημάτων  
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών  
Εθνικό Μετσόβιο Πολυτεχνείο

WWW: <http://cslab.ece.ntua.gr/research/>



Μάρτιος 2019

# Επισκόπηση

## 1 Εισαγωγή

- Unikernels
- Unikernel frameworks
- Βασική ιδέα

## 2 Μηχανισμός pipe

- Γενική εικόνα
- Πρώτο στάδιο υλοποίησης
- Δεύτερο στάδιο υλοποίησης
- Τρίτο στάδιο υλοποίησης

## 3 Μηχανισμός fork

- Γενική εικόνα
- Βήματα
- Αξιολόγηση



# Εισαγωγή

## Unikernels

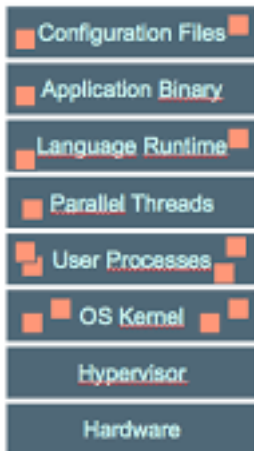
Εξειδικευμένες εικόνες μηχανές, με ένα μοναδικό χώρο διευθύνσεων, τα οποία κατασκευάζονται χρησιμοποιώντας library operating systems

## Αναλύοντας τον ορισμό

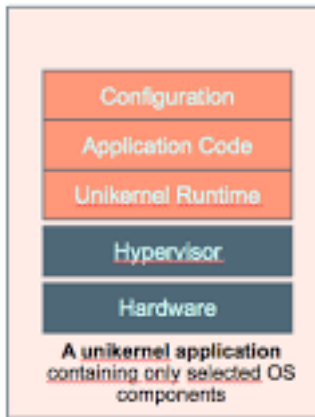
- Εξειδικευμένες
- Μοναδικός χώρος διευθύνσεων
- library operating systems



# Typical OS vs unikernel



Typical application  
running above an OS



A unikernel application  
containing only selected OS  
components

# Χαρακτηριστικά των unikernels

## Πλεονεκτήματα

- Γρήγοροι χρόνοι εκκίνησης
- Μικρό memory footprint
- Περισσότερη ασφάλεια

## Μειονεκτήματα

- Porting
- Μικρότερη υποστήριξη



# Επισκόπηση

## 1 Εισαγωγή

- Unikernels
- Unikernel frameworks
- Βασική ιδέα

## 2 Μηχανισμός pipe

- Γενική εικόνα
- Πρώτο στάδιο υλοποίησης
- Δεύτερο στάδιο υλοποίησης
- Τρίτο στάδιο υλοποίησης

## 3 Μηχανισμός fork

- Γενική εικόνα
- Βήματα
- Αξιολόγηση



# Unikernel frameworks

## Rumprun

- Βασισμένο στο NetBSD και συγκεκριμένα στα rump kernels
- POSIX friendly
- Υποστήριξη για threads, filesystem
- Υποστηρίζει πολλές γλώσσες
- Πολλές εφαρμογές έτοιμες να εκτελεστούν σε αυτό



# Unikernel frameworks

## OSv

- Φτιαγμένο από την αρχή, με στόχο την εκτέλεση στο cloud
- POSIX compatible
- Υποστήριξη για threads, filesystem
- Υποστηρίζει πολλές γλώσσες
- Πολλές εφαρμογές έτοιμες να εκτελεστούν σε αυτό
- Από τα πιο ενεργά projects





# Unikernel frameworks

## IncludeOS

- Φτιαγμένο από την αρχή
- Μερικώς POSIX compatible
- Single threaded
- Ταχύτατα αναπτυσσόμενο project
- Υποστήριξη μόνο για εφαρμογές σε C++



# Unikernel frameworks

## MirageOS

- Φτιαγμένο από την αρχή χρησιμοποιώντας OCaml
- Υποστήριξη μόνο για εφαρμογές σε OCaml
- Από τα πιο παλιά unikernel frameworks



# Unikernel frameworks

## MirageOS

- Φτιαγμένο από την αρχή χρησιμοποιώντας OCaml
- Υποστήριξη μόνο για εφαρμογές σε OCaml
- Από τα πιο παλιά unikernel frameworks

## Πολλά ακόμα

- ClickOS
- LKL
- Mini-OS, Unikraft
- HalVM, LING



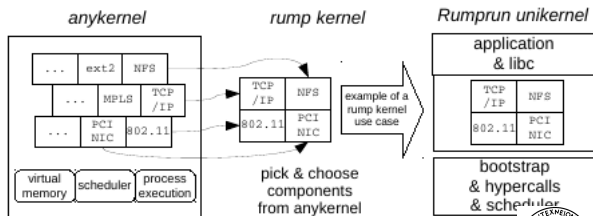
# Rumprun

## Anykernels

Codebase (πυρήνα) όπου οι οδηγοί μπορούν να εξαχθούν και να ενσωματωθούν σε οποιοδήποτε μοντέλο ΛΣ

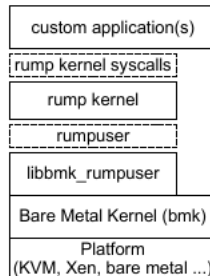
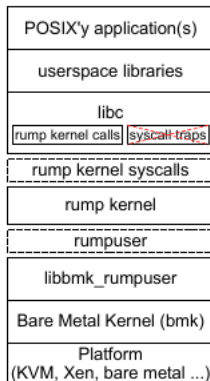
## Rump kernels

παρέχουν drivers του NetBSD ως φορητά εξαρτήματα με τα οποία μπορούμε να εκτελέσουμε εφαρμογές χωρίς να ναι απαραίτητη η ύπαρξη ΛΣ



# Rumprun stack

same threads  
throughout entire stack



# Επισκόπηση

## 1 Εισαγωγή

- Unikernels
- Unikernel frameworks
- Βασική ιδέα

## 2 Μηχανισμός pipe

- Γενική εικόνα
- Πρώτο στάδιο υλοποίησης
- Δεύτερο στάδιο υλοποίησης
- Τρίτο στάδιο υλοποίησης

## 3 Μηχανισμός fork

- Γενική εικόνα
- Βήματα
- Αξιολόγηση



# Βασική ιδέα

## single process

- Βασικό κοινό χαρακτηριστικό όλων των unikernels: single process
- Δεν υποστηρίζεται η κλήση fork

## Ιδέα

- POSIX friendly
- Διατήρηση βασικών χαρακτηριστικών unikernels
- Unikernels ως διεργασίες και hypervisor ως λειτουργικό σύστημα



# Αντικείμενο διπλωματικής

## Μηχανισμός fork

- Αντί για δημιουργία νέας διεργασίας στο υπάρχον unikernel
- Δημιουργία νέου unikernel
- Από το επίπεδο των διεργασιών στο επίπεδο των εικονικών μηχανών

## Μηχανισμός pipe

- Επικοινωνία μεταξύ δύο διεργασιών
- Συνήθης πρακτική, pipe
- Κατασκευή ενός μηχανισμού pipe σε επίπεδο εικονικών μηχανών.





# Επισκόπηση

## 1 Εισαγωγή

- Unikernels
- Unikernel frameworks
- Βασική ιδέα

## 2 Μηχανισμός pipe

- Γενική εικόνα
- Πρώτο στάδιο υλοποίησης
- Δεύτερο στάδιο υλοποίησης
- Τρίτο στάδιο υλοποίησης

## 3 Μηχανισμός fork

- Γενική εικόνα
- Βήματα
- Αξιολόγηση



# Γενική εικόνα

## Pipe system call

```
int pipe(int fildes[2]);
```

Pipe semantics:

- Αν το pipe είναι άδειο, η κλήση ανάγνωσης μπλοκάρει μέχρι να προκύψουν δεδομένα
- Αν το pipe είναι γεμάτο, η κλήση εγγραφής μπλοκάρει μέχρι να ελευθερωθεί χώρος
- Αν δεν υπάρχουν ανοιχτά άκρα εγγραφής, τότε η ανάγνωση στο pipe επιστρέφει 0, end of file
- Αν δεν υπάρχουν ανοιχτά άκρα ανάγνωσης, τότε εγγραφή στο pipe επιστρέφει το error EPIPE



# Γενική εικόνα

## Pipe σε unikernels

Χρήση του pipe για unikernels, όπως και στις διεργασίες ενός συμβατικού ΛΣ.

## Τρία στάδια υλοποίησης

- 1 function call (sockets)
- 2 system call (sockets)
- 3 system call (κοινή μνήμη)



# Επισκόπηση

## 1 Εισαγωγή

- Unikernels
- Unikernel frameworks
- Βασική ιδέα

## 2 Μηχανισμός pipe

- Γενική εικόνα
- Πρώτο στάδιο υλοποίησης
- Δεύτερο στάδιο υλοποίησης
- Τρίτο στάδιο υλοποίησης

## 3 Μηχανισμός fork

- Γενική εικόνα
- Βήματα
- Αξιολόγηση



# Πρώτο στάδιο υλοποίησης

## function call

- Στον κώδικα της εφαρμογής
- Δημιουργία δύο sockets (TCP)
- Επιστροφή των δύο sockets
- Διεύθυνση παραλήπτη
- Δεν υλοποιήθηκαν όλα τα semantics του pipe



# Επισκόπηση

## 1 Εισαγωγή

- Unikernels
- Unikernel frameworks
- Βασική ιδέα

## 2 Μηχανισμός pipe

- Γενική εικόνα
- Πρώτο στάδιο υλοποίησης
- Δεύτερο στάδιο υλοποίησης
- Τρίτο στάδιο υλοποίησης

## 3 Μηχανισμός fork

- Γενική εικόνα
- Βήματα
- Αξιολόγηση



# Δεύτερο στάδιο υλοποίησης

## system call

- Χρήση UDP sockets, αντί για TCP
- Δημιουργία system call για τον πυρήνα του NetBSD
- Δημιουργία δύο sockets
- Επιστρέφονται δύο file descriptors
- ioctl κλήση, για τη διεύθυνση του παραλήπτη
- Δεν υλοποιήθηκαν όλα τα semantics του pipe



# Επισκόπηση

## 1 Εισαγωγή

- Unikernels
- Unikernel frameworks
- Βασική ιδέα

## 2 Μηχανισμός pipe

- Γενική εικόνα
- Πρώτο στάδιο υλοποίησης
- Δεύτερο στάδιο υλοποίησης
- Τρίτο στάδιο υλοποίησης

## 3 Μηχανισμός fork

- Γενική εικόνα
- Βήματα
- Αξιολόγηση





# Τρίτο στάδιο υλοποίησης

## system call

- Χρήση κοινής μνήμης μεταξύ των εικονικών μηχανών (invshmem - nahanni)
- Επιστρέφονται δύο file descriptors
- Ίδιος host
- Υλοποιήθηκαν όλα τα semantics του pipe



## Τρίτο στάδιο υλοποίησης

### ivshmem

- Χρήση του μηχανισμού ivshmem - nahanni
- Κοινή μνήμη μεταξύ δύο εικονικών μηχανών σε QEMU
- PCI driver

### Υλοποίηση

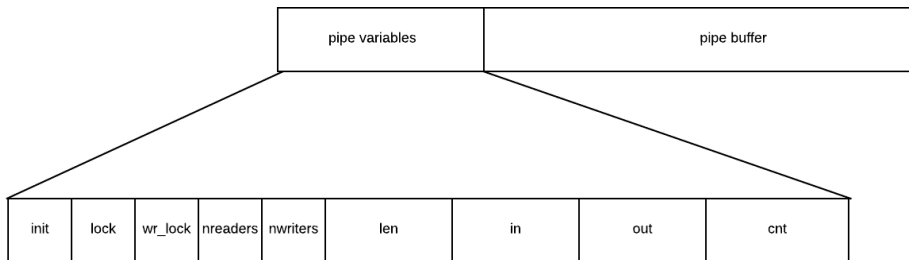
- PCI driver
- Pipe system call



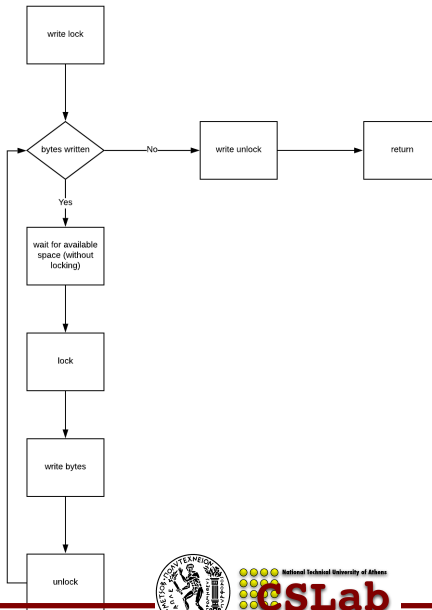
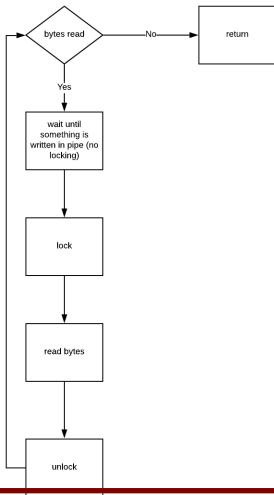
# Τρίτο στάδιο υλοποίησης

## Υλοποίηση

- PCI driver
- Pipe system call



# Τρίτο στάδιο υλοποίησης



# Επισκόπηση

## 1 Εισαγωγή

- Unikernels
- Unikernel frameworks
- Βασική ιδέα

## 2 Μηχανισμός pipe

- Γενική εικόνα
- Πρώτο στάδιο υλοποίησης
- Δεύτερο στάδιο υλοποίησης
- Τρίτο στάδιο υλοποίησης

## 3 Μηχανισμός fork

- Γενική εικόνα
- Βήματα
- Αξιολόγηση



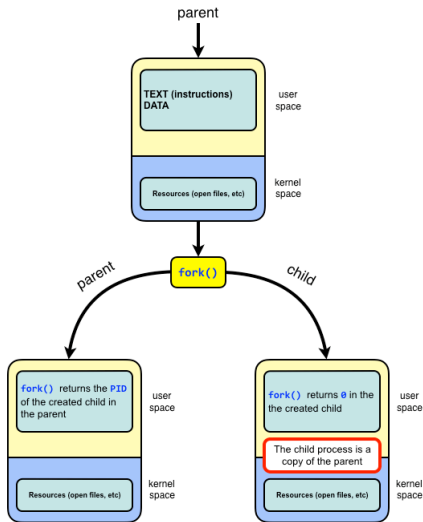
# Γενική εικόνα

## fork system call

- Δημιουργία νέας διεργασίας
- Η νέα διεργασία είναι ίδια με τη διεργασία γονέα
- Διαχωρισμός των δύο διεργασιών με τιμή επιστροφής από την κλήση συστήματος
- Διατήρηση ανοιχτών file descriptors στη διεργασία παιδί.



# fork σε συμβατικά ΛΣ



# Γενική εικόνα

## fork system call

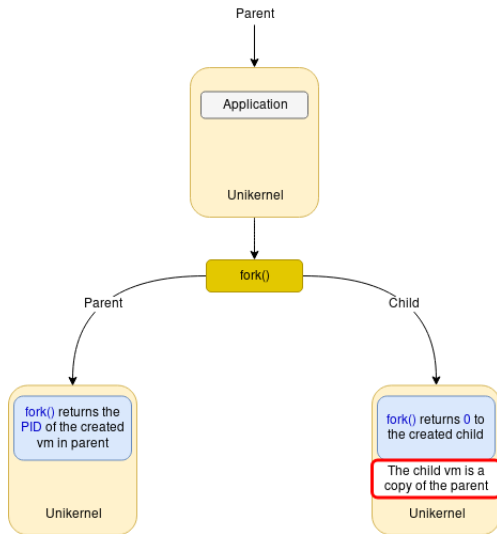
- Δημιουργία νέας διεργασίας
- Η νέα διεργασία είναι ίδια με τη διεργασία γονέα
- Διαχωρισμός των δύο διεργασιών με τιμή επιστροφής από την κλήση συστήματος
- Διατήρηση ανοιχτών file descriptors στη διεργασία παιδί.

## Στόχος

Μεταφορά της ίδιας λειτουργίας σε επίπεδο εικονικών μηχανών



# unikernel fork



# Γενική εικόνα

## Υλοποίηση

- Μηχανισμός migration του QEMU
- hypercalls

## Βήματα

- Ενημέρωση των κοινών μεταβλητών του pipe (αν υπάρχει pipe)
- Εκκίνηση διαδικασίας migration
- Αναμονή για migration
- Εκκίνηση νέας εικονικής μηχανής με βάση το migration file
- Συγχρονισμός κοινής μνήμης



# Επισκόπηση

## 1 Εισαγωγή

- Unikernels
- Unikernel frameworks
- Βασική ιδέα

## 2 Μηχανισμός pipe

- Γενική εικόνα
- Πρώτο στάδιο υλοποίησης
- Δεύτερο στάδιο υλοποίησης
- Τρίτο στάδιο υλοποίησης

## 3 Μηχανισμός fork

- Γενική εικόνα
- Βήματα
- Αξιολόγηση



# Ενημέρωση κοινών μεταβλητών στο pipe

- Αν υπάρχει pipe θα πρέπει να ενημερωθούν οι κοινές μεταβλητές
- Έλεγχος ύπαρξης pipe
- Αύξηση των μετρητών ανοιχτών άκρων
- Πρόσβαση στην κοινή μνήμη



# Εκκίνηση δημιουργίας migration file

## Από τη μεριά του rumpun

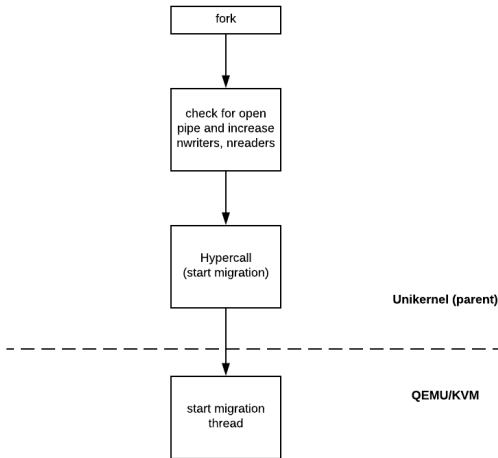
Hypercall για την εκκίνηση δημιουργίας migration file

## Από τη μεριά του QEMU

- Χρήση του μηχανισμού migration
- Migration thread
- Λειτουργία της εικονικής μηχανής γονέας



# Εκκίνηση δημιουργίας migration file



# Αναμονή για την περάτωση του migration

## Από τη μεριά του rumpun

- Busy wait
- Σημείο εκκίνησης του unikernel παιδί.

## Από τη μεριά του QEMU

- Κατάσταση του migration thread
- Hypercalls counter



# Αναμονή για την περάτωση του migration

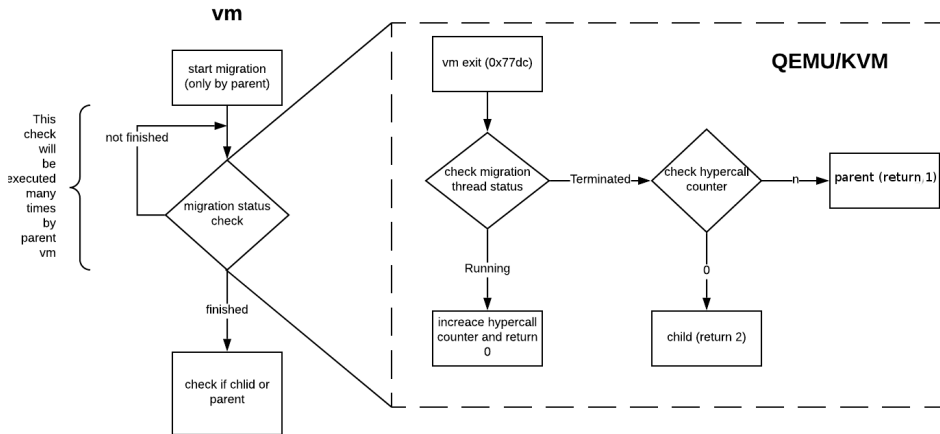
## Hypercalls counter

- Διάκριση παιδιού, γονέα
- Διαφορετικοί counters για κάθε vm
- Migration, χρονοβόρα διαδικασία
- Ένα ακριβώς hypercall από το παιδί





# Αναμονή για την περάτωση του migration



# Δημιουργία νέας εικονικής μηχανής

Από τη μεριά του rumpun

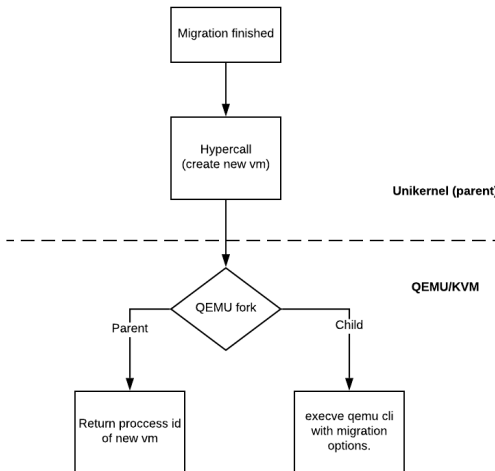
Hypercall για την εκκίνηση νέας εικονικής μηχανής

Από τη μεριά του QEMU

- fork
- Επιστροφή process id
- Εκκίνηση νέας εικονικής μηχανής



# Εκκίνηση δημιουργίας migration file

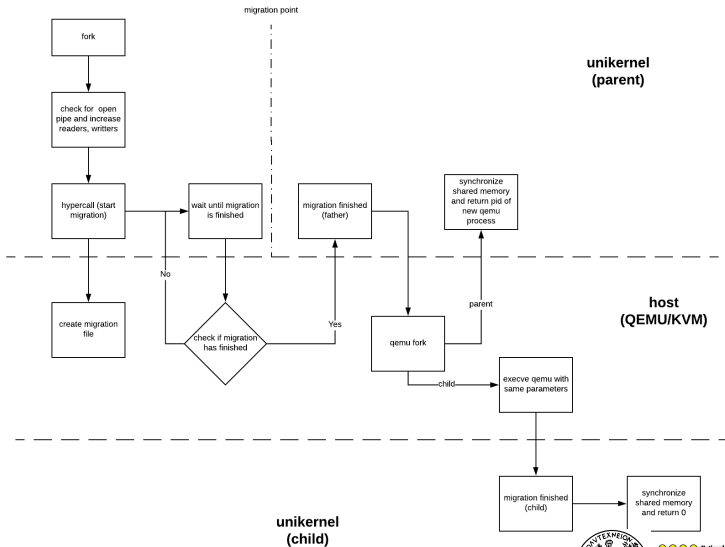


# Συγχρονισμός εικονικής μνήμης

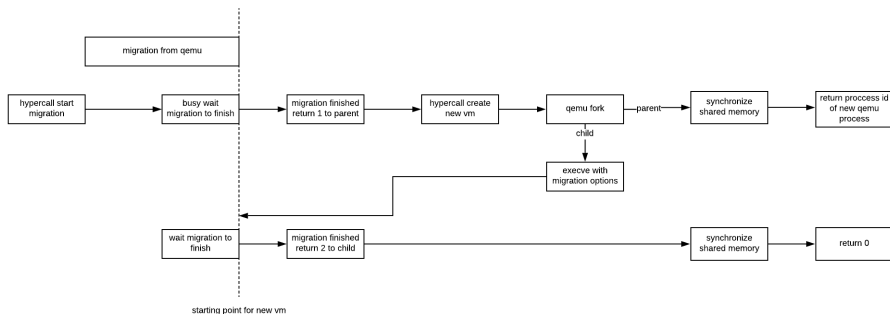
- Μόνο στην περίπτωση χρήσης pipe
- Bug(?) κοινής μνήμης
- Busy wait από το γονιό
- Εγγραφή στην κοινή μνήμη από το παιδί



# Όλα τα βήματα



# Όλα τα βήματα



# Επισκόπηση

## 1 Εισαγωγή

- Unikernels
- Unikernel frameworks
- Βασική ιδέα

## 2 Μηχανισμός pipe

- Γενική εικόνα
- Πρώτο στάδιο υλοποίησης
- Δεύτερο στάδιο υλοποίησης
- Τρίτο στάδιο υλοποίησης

## 3 Μηχανισμός fork

- Γενική εικόνα
- Βήματα
- Αξιολόγηση



# Μετρήσεις

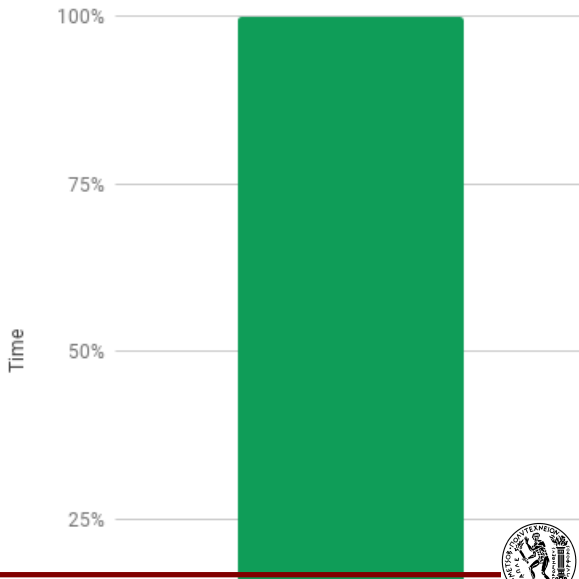
## Σύγκριση fork

- Σύγκριση fork σε linux πυρήνα και του δικού μας fork
- Σε ίδια έκδοση QEMU (2.11.2)
- Linux 4.9.0-7-amd64 #1 SMP Debian 4.9.110-3+deb9u2
- Linux: 0.541ms, Unikernel: 137,423ms
- Διαφορά





# Time for each step in fork



# Σύνοψη

## Unikernels:

- Ενδιαφέρουσα τεχνολογία
- Διαφορετικοί στόχοι
- POSIX compatible
- Single process

## Αντιμετώπιση των unikernels ως διεργασίες

- Μηχανισμός pipe για unikernels σε ίδιο και διαφορετικό host
- Μηχανισμός fork για unikernels σε ίδιο host



# Μελλοντικές Επεκτάσεις

- Χρήση σημάτων, για τη βελτίωση του μηχανισμού pipe
- Επέκταση inter-unikernel μηχανισμών επικοινωνίας
- Βελτίωση μηχανισμού fork
- Επέκταση του μηχανισμού και σε άλλες πλατφόρμες.



# Ευχαριστώ!

## Ερωτήσεις;





# Backup

