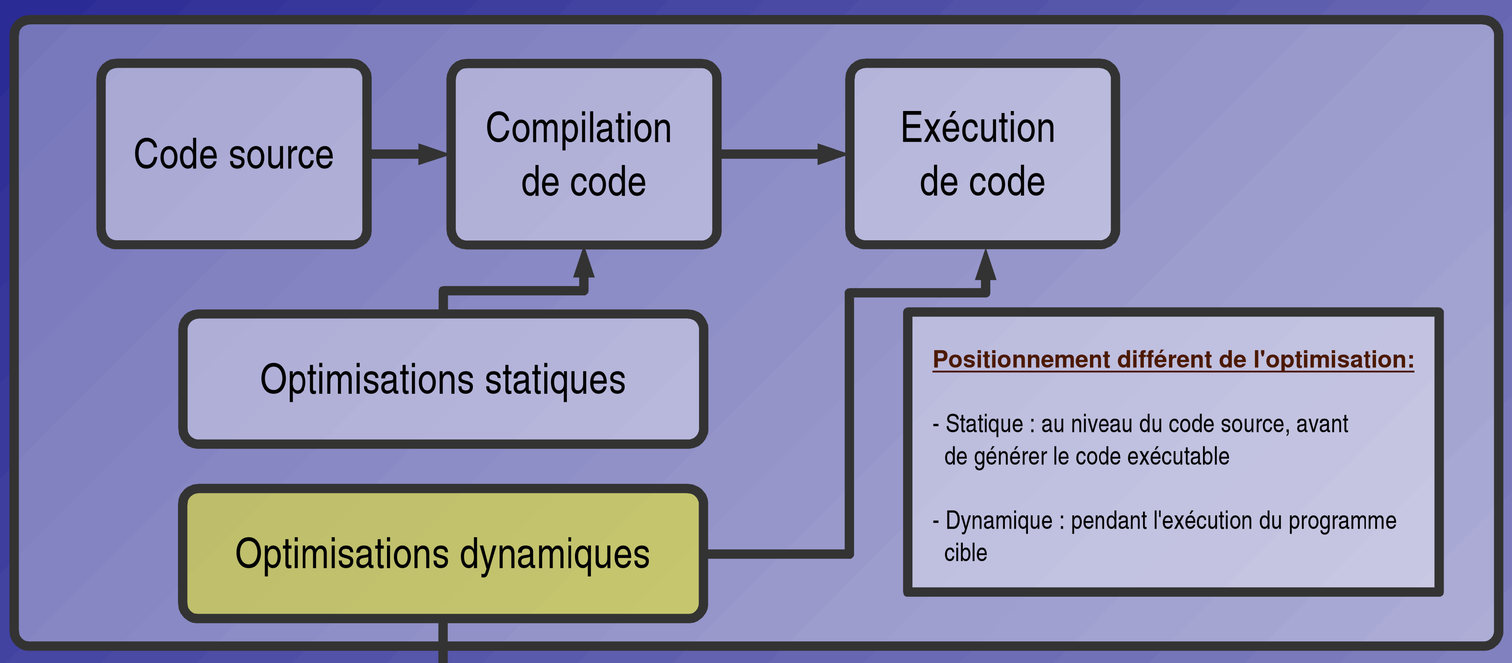
# Optimisation dynamique des accès mémoires

Jean Christophe Beyler
LSIIT/ICPS
beyler@icps.u-strasbg.fr



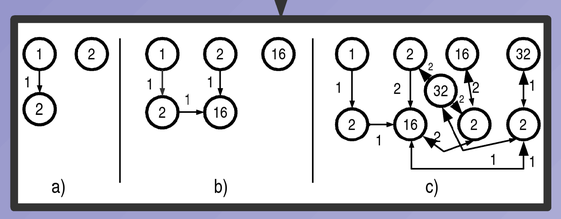

## Construction du modèle Markovien

- Les noeuds représentent les sauts mémoire effectués
- Les arcs représentent une précédence entre deux sauts
- Les étiquettes des arcs représentent le nombre de fois que cet arc a été suivi

Suite de sauts mémoires à étudier

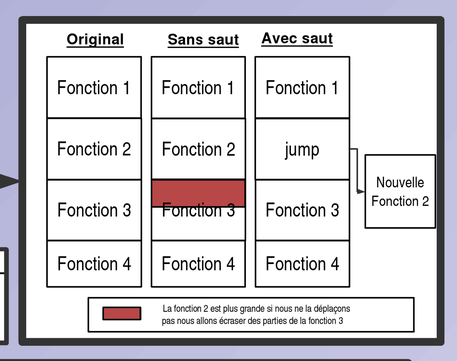
1 2 16 2 32 2 16 32 ...

Modéle Markovien




## Optimiseur de code dynamique

| Code original | Code binaire | Nouveau code | Code final |
|---|---|---|---|
| mov r36 = 4;; | 0a 20 11 00 00 24 | 0a 20 19 00 00 24 | mov r36 = 6;; |
| ld8 r35 = [r35] | 30 02 8c 30 20 00 | 30 02 8c 30 20 00 | ld8 r35 = [r35] |
| nop.i 0 | 00 00 04 00 | 00 00 04 00 | nop.i 0 |




## Modification dynamique du code

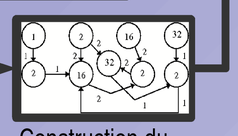
- Décoder le code binaire
- Arrêter le programme cible : utilisation de threads par exemple
- Changer le code “non optimisé” : faire attention au problème de corruption mémoire
- Relancer le programme cible

## Programme cible

Récursivité :
```
void f(void * t, int a)
{
//optimisé l'accès t[a]
u[c] = t[a} + 4;
...
f(u,b);
}
```

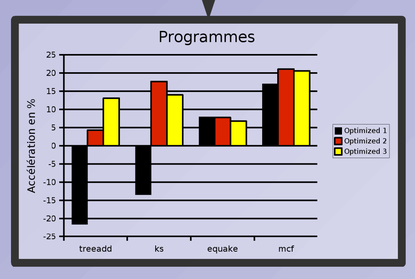
Boucle :
```
for(i=0;i<n;i++)
{
j = f(i);
//optimiser l'accès u[j]
t[i] = u[j];
}
```

Liste chaînée :
```
while(tmp)
{
s += tmp->val;
//optimiser l'accès tmp
tmp = tmp->next;
}
```



Construction du graphe

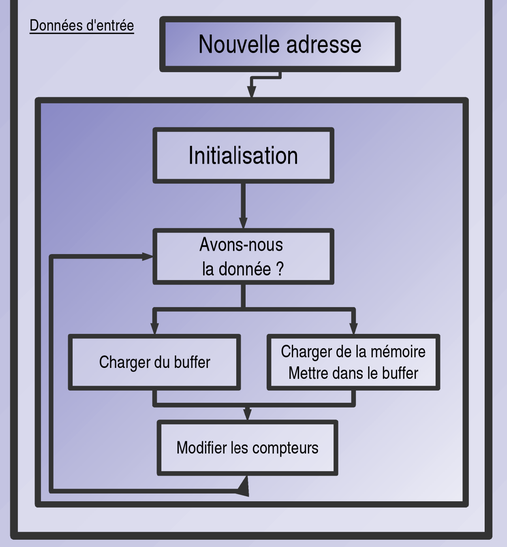
Programme optimisé




## Résultats obtenus sur différents benchmarks

- Le temps du programme optimisé dépend du choix des paramètres
- Différents paramètres ont été utilisés : la profondeur du graphe, le temps de construction et la distance de préchargement

Données d'entrée




## Un exemple concret

DORM: *Dynamic Optimization by Reorganizing Data*

- Un simple automate sert à vérifier si la donnée est en mémoire ou dans le buffer
- Le buffer sert à améliorer la localité spatiale du programme
- La diminution des défauts cache accélèrera le programme cible

## Optimisation dynamique:

- Modifier le comportement du programme *pendant* l'exécution pour améliorer certaines caractéristiques : diminution des défauts de cache et/ou de la consommation d'énergie...
- La solution peut être entièrement dynamique, utiliser l'aide d'un compilateur, ou modifier directement le code binaire du programme ciblé.
- Le dynamisme de ce procédé est un complément intéressant pour l'optimisation statique car il permet d'évaluer des informations inconnues au moment de la compilation.