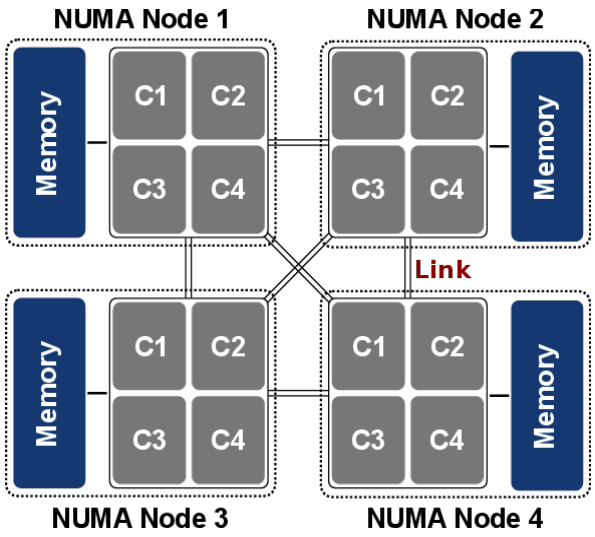
Détection automatique d'anomalies de performance

Mohamed Said Mosli Bouksiaa, François Trahay, Gaël Thomas

Introduction

- Matériel complexe : NUMA, caches hiérarchiques, GPU, etc
- Logiciels complexes : MPI+OpenMP, MPI+CUDA, etc



→ Outils d'analyse de performance

Analyse de performance : les outils de trace

Lancer l'application

- **Intercepter les événements intéressants (appels de fonctions, messages, etc)**

```
 4
 5  //code original
 6  f (a, b, c);
 7
 8  //remplacé par :
 9  //code instrumenté
10  _f_ (a, b, c) {
11    enregistrer_entree_f (a, b, c);
12    f (a, b, c);
13    enregistrer_sortie_f ();
14  }
15
```

- **Générer une trace d'exécution**

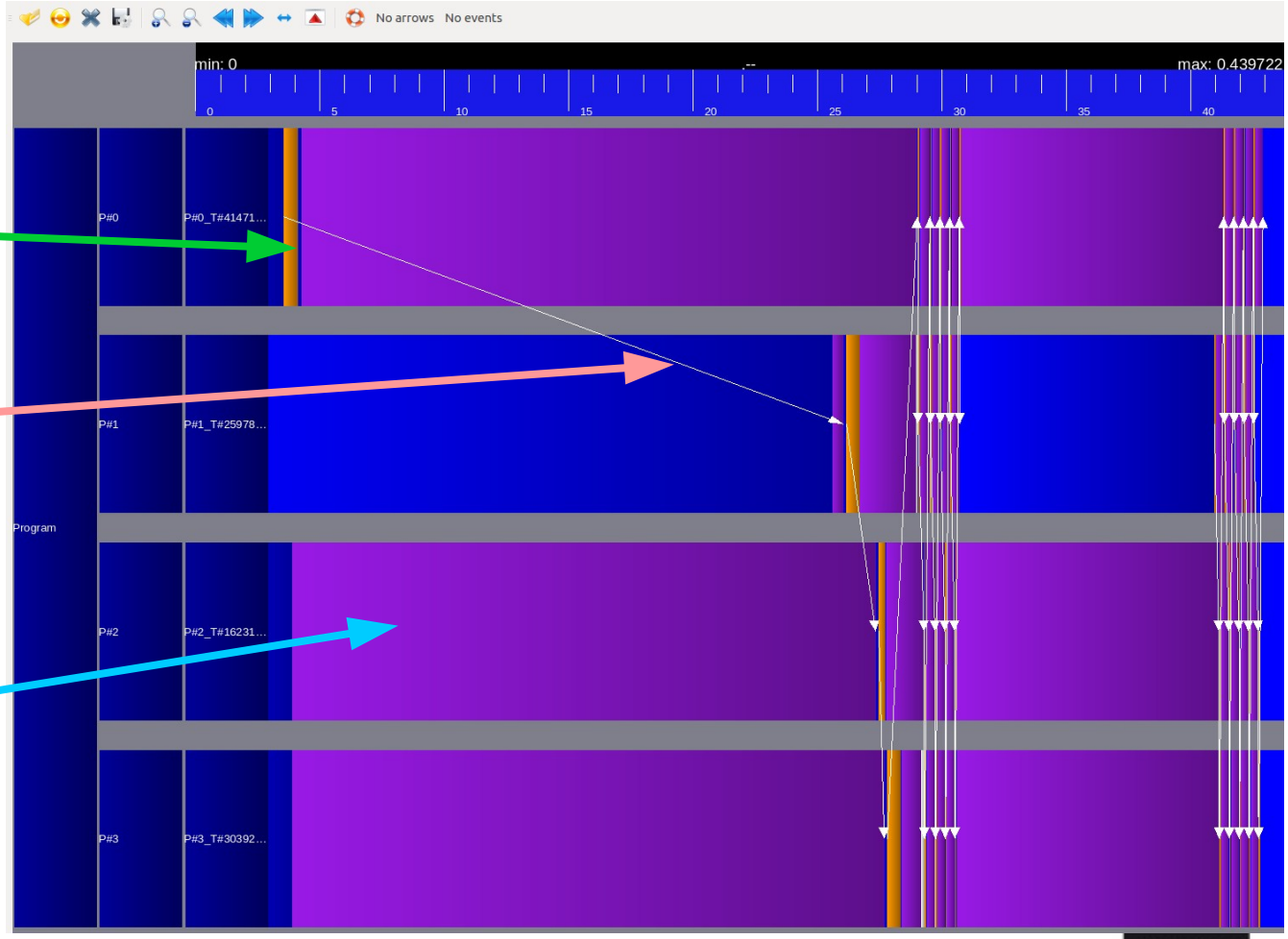
```
(#82)   5292 Enter: function 14, process 7, source 0
(#83)   5387 Leave: function 1, process 7, source 0
(#84)   5540 Enter: function 14, process 3, source 0
(#85)   5631 Leave: function 1, process 3, source 0
(#86)   5767 Enter: function 14, process 5, source 0
(#87)   5801 Leave: function 1, process 5, source 0
(#88)   5995 Counter: process 8, counter 1, value 14829
(#89)   6062 Counter: process 8, counter 1, value 14573
(#90)   6747 Enter: function 14, process 9, source 0
(#91)   6764 Counter: process 6, counter 1, value 14829
(#92)   6796 Leave: function 1, process 9, source 0
(#93)   6806 Counter: process 6, counter 1, value 14573
```

Visualisation des traces

État = Envoi de message

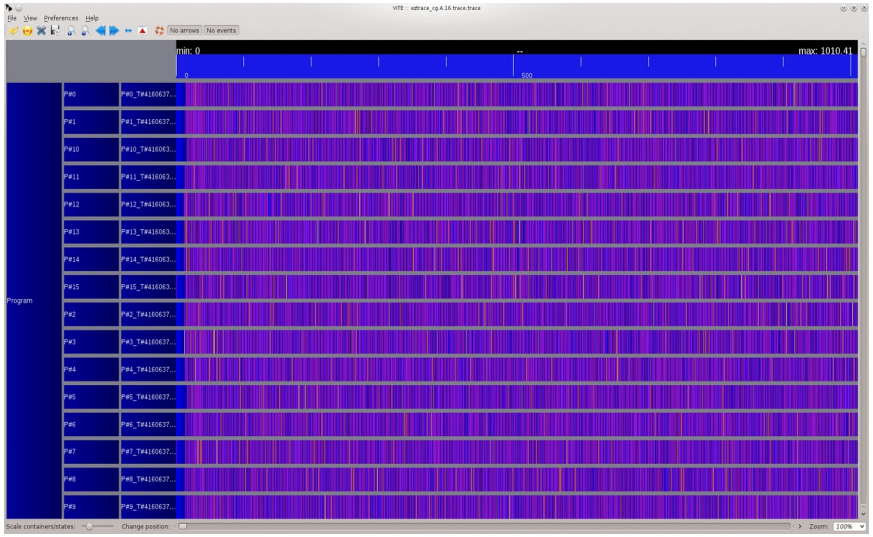
Flèche = un message

État = Réception de message



Visualisation de grosses traces

- **Visualiser une grosse trace est difficile**
  - Millions d'événements
- **Comment détecter les parties intéressantes d'une trace ?**

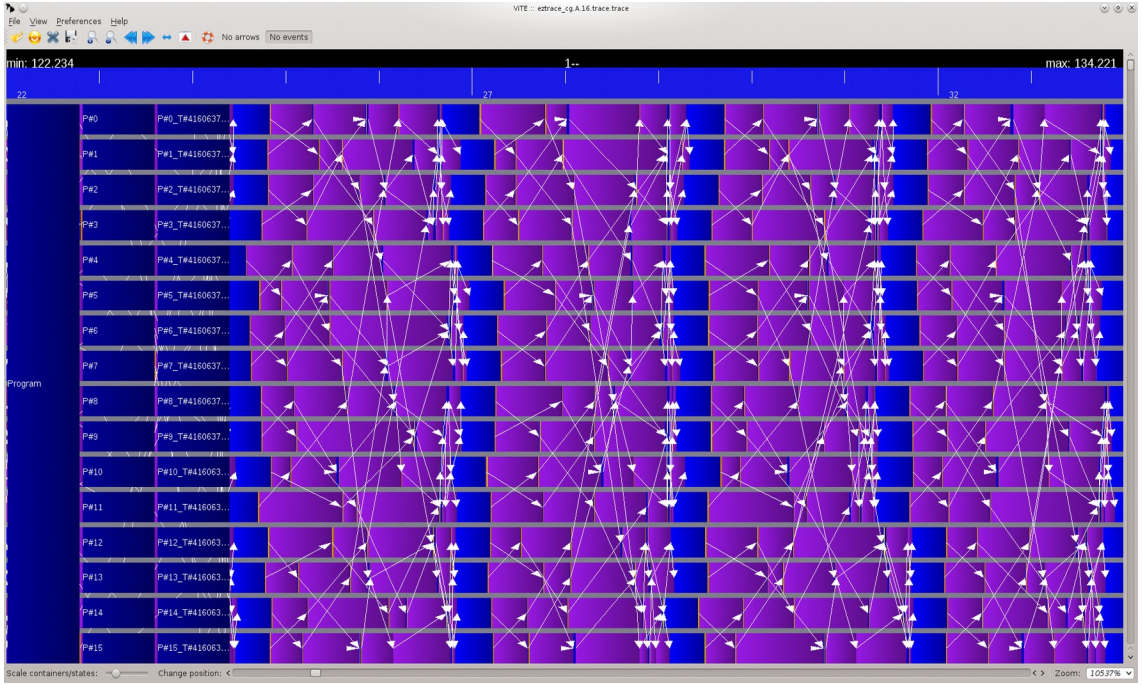


NPB CG classe A 16 Processus MPI – 426 000 événements

Visualisation de grosses traces

- **Une trace est souvent structurée**
  - Boucles
  - Fonctions

- **Beaucoup d'informations qui se ressemblent**



NPB CG classe A 16 Processus MPI – 426 000 événements

Proposition : Repérer les parties à examiner en priorité

- **Détecter les similarités dans une trace : les <u>motifs récurrents</u>**
  - Les phases de l'application qui se répètent

```
100 x {
     MPI_SEND (src=0   dest=1  len=16  tag=0)
     MPI_RECV (src=1   dest=0  len=16  tag=0)
 }
 MPI_Barrier
 10000 x {
     MPI_SEND (src=0   dest=1  len=16  tag=0)
     MPI_RECV (src=1   dest=0  len=16  tag=0)
 }
 MPI_Barrier
```

- **Sélectionner les points intéressants dans la trace**
  - Repérer <u>des instances représentatives</u>
  - Analyse comparative : repérer <u>des anomalies</u>

Récapitulation

- **Problème : inexploitabilité des grosses traces à l'état brut**

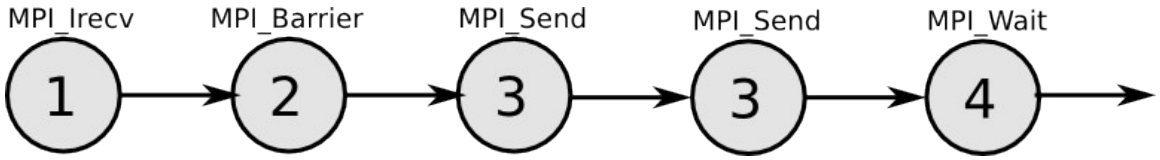
- **Idée : trouver les motifs récurrents ⇒ localiser des zones d'intérêt (de petite taille) que l'humain peut examiner**

- **Contributions :**
  - Détecter les motifs récurrents
  - Détecter des anomalies de performance
  - Détecter les relations de causalité entre anomalies

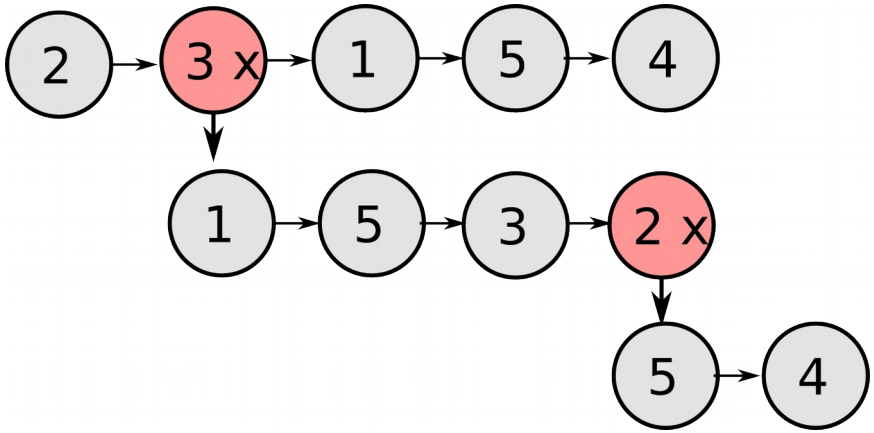
Détection des motifs récurrents

Représentation d'une trace

- **Une trace peut être représentée comme une liste d'événements**

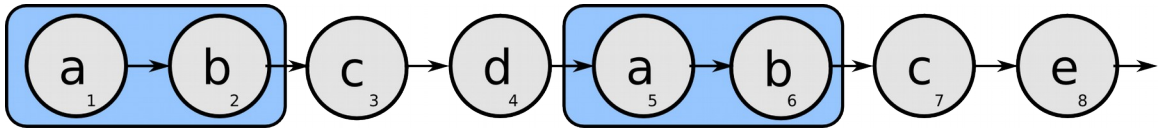


- **Objectif : trouver des motifs dans cette liste**
  - Peut être vu comme une factorisation



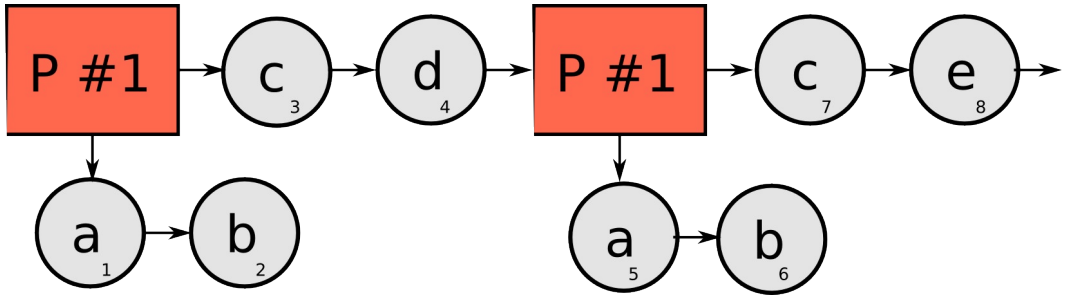
Algorithme de factorisation
Première étape : trouver des petits motifs

- **Trouver un couple d'événements (e1, e2) qui apparaît plusieurs fois**
  → motifs à 2 événements
- **Parcourir la liste des événements à la recherche d'un couple qui se répète**



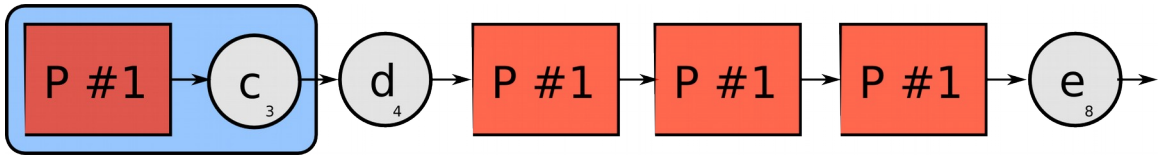
Algorithme de factorisation
Première étape : trouver des petits motifs

- **Trouver un couple d'événements (e1, e2) qui apparaît plusieurs fois**
  → motifs à 2 événements
- **Parcourir la liste des événements à la recherche d'un couple qui se répète**



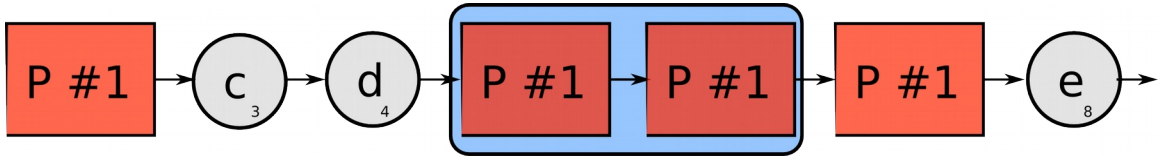
Algorithme de factorisation
Deuxième étape : trouver des boucles de ces motifs

- Une boucle est la concaténation de plusieurs occurrences d'un motif
  - Chaque itération a été détectée comme occurrence du motif
- Parcourir la liste des événements à la recherche d'occurrences consécutives du même motif



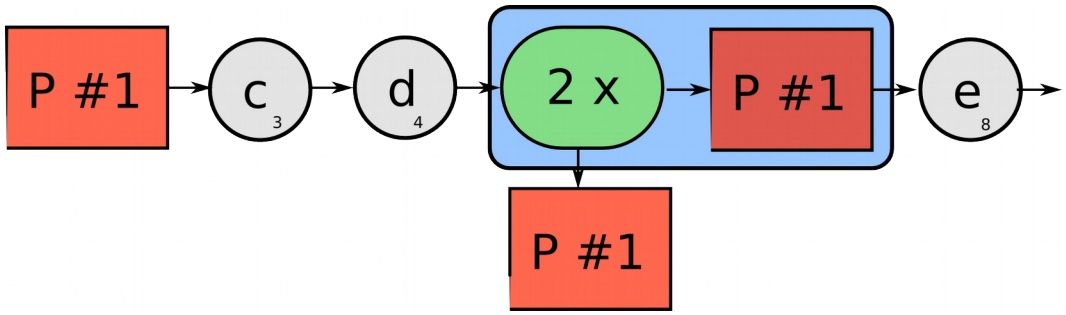
Algorithme de factorisation
Deuxième étape : trouver des boucles de ces motifs

- Une boucle est la concaténation de plusieurs occurrences d'un motif
  - Chaque itération a été détectée comme occurrence du motif
- Parcourir la liste des événements à la recherche d'occurrences consécutives du même motif



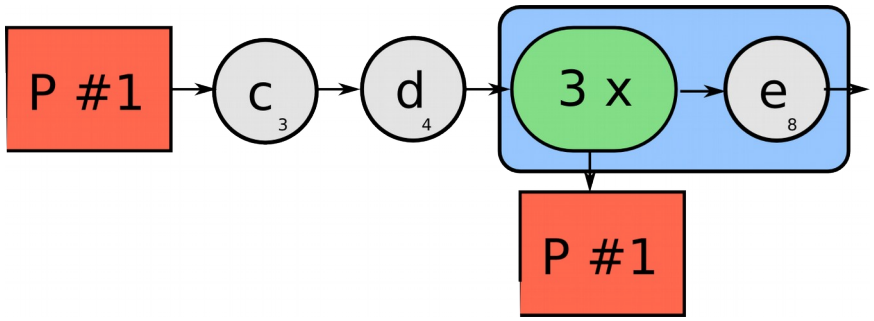
Algorithme de factorisation
Deuxième étape : trouver des boucles de ces motifs

- Une boucle est la concaténation de plusieurs occurrences d'un motif
  - Chaque itération a été détectée comme occurrence du motif
- Parcourir la liste des événements à la recherche d'occurrences consécutives du même motif



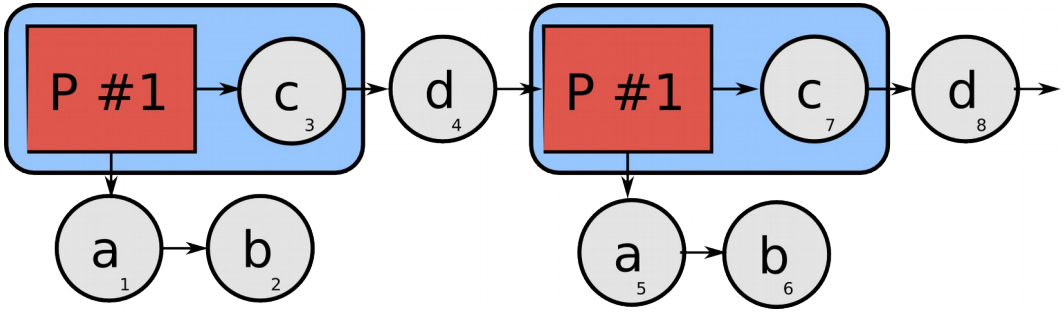
Algorithme de factorisation
Deuxième étape : trouver des boucles de ces motifs

- Une boucle est la concaténation de plusieurs occurrences d'un motif
  - Chaque itération a été détectée comme occurrence du motif
- Parcourir la liste des événements à la recherche d'occurrences consécutives du même motif



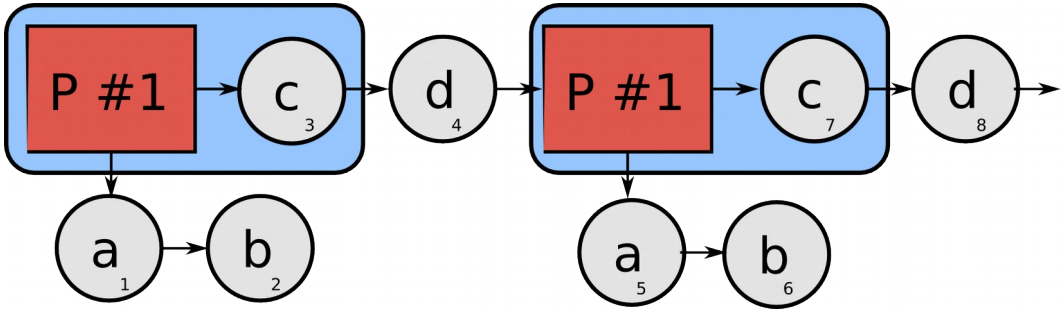
Algorithme de factorisation
Troisième étape : Essayer d'étendre les motifs

- Est-ce un motif à 2 événements ou à 3 événements ?



Algorithme de factorisation
Troisième étape : Essayer d'étendre les motifs

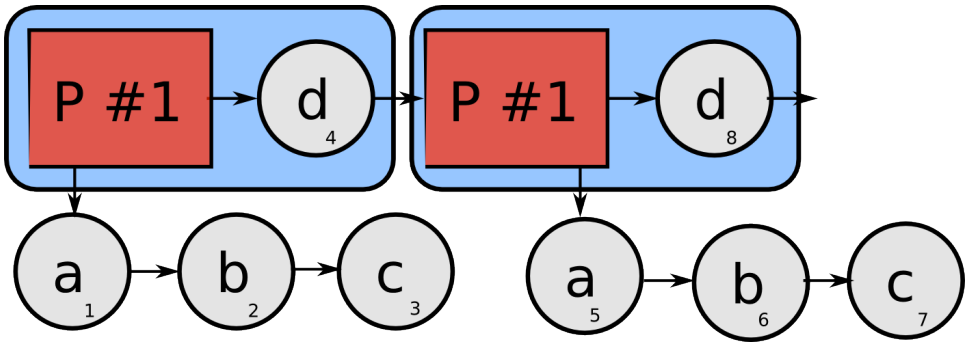
- Est-ce un motif à 2 événements ou à 3 événements ?
- 1er cas : le motif P#1 est toujours suivi par l'événement C



Algorithme de factorisation
Troisième étape : Essayer d'étendre les motifs

- Est-ce un motif à 2 événements ou à 3 événements ?
- 1er cas : le motif P#1 est toujours suivi par l'événement C

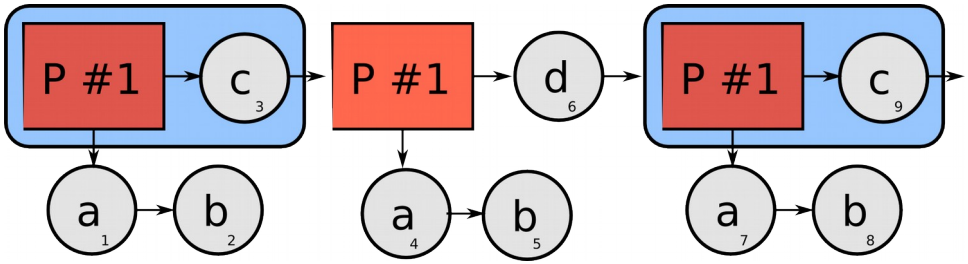
  → P#1 est un motif à 3 événements (au moins)



Algorithme de factorisation
Troisième étape : Essayer d'étendre les motifs

- Est-ce un motif à 2 événements ou à 3 événements ?
- 2ème cas : le motif P#1 n'est suivi par l'événement C que quelques fois

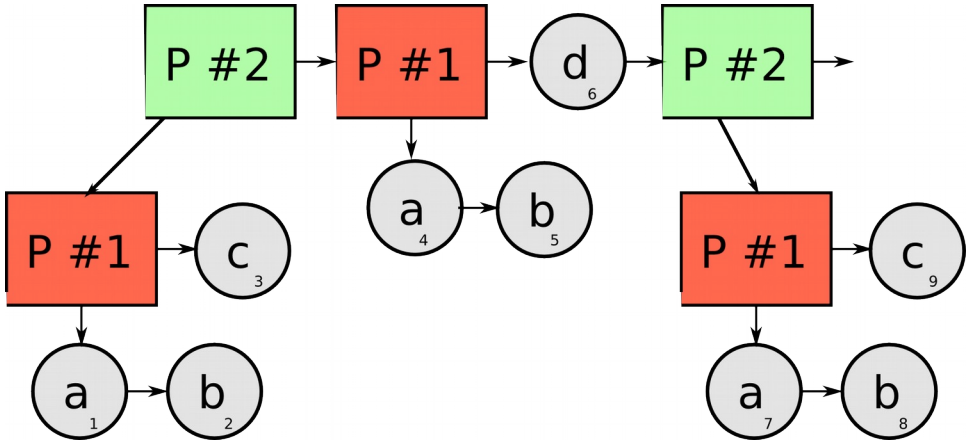
  → créer le motif P#2 qui englobe le motif P#1



Algorithme de factorisation
Troisième étape : Essayer d'étendre les motifs

- Est-ce un motif à 2 événements ou à 3 événements ?
- 2ème cas : le motif P#1 n'est suivi par l'événement C que quelques fois

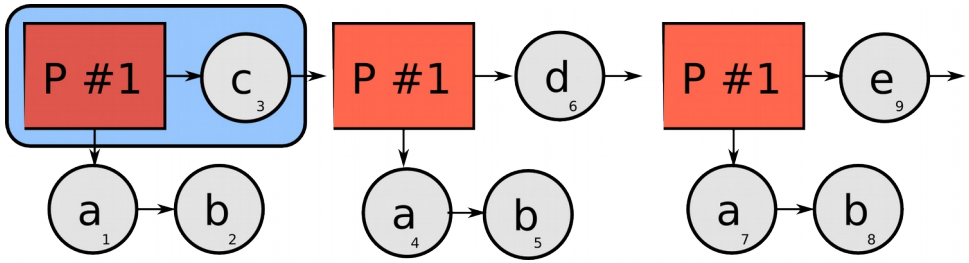
  → créer le motif P#2 qui englobe le motif P#1



Algorithme de factorisation
Troisième étape : Essayer d'étendre les motifs

- Est-ce un motif à 2 événements ou à 3 événements ?
- 3ème cas : le motif P#1 n'est suivi par l'événement C qu'une seule fois

  → ne rien faire



Motifs détectés

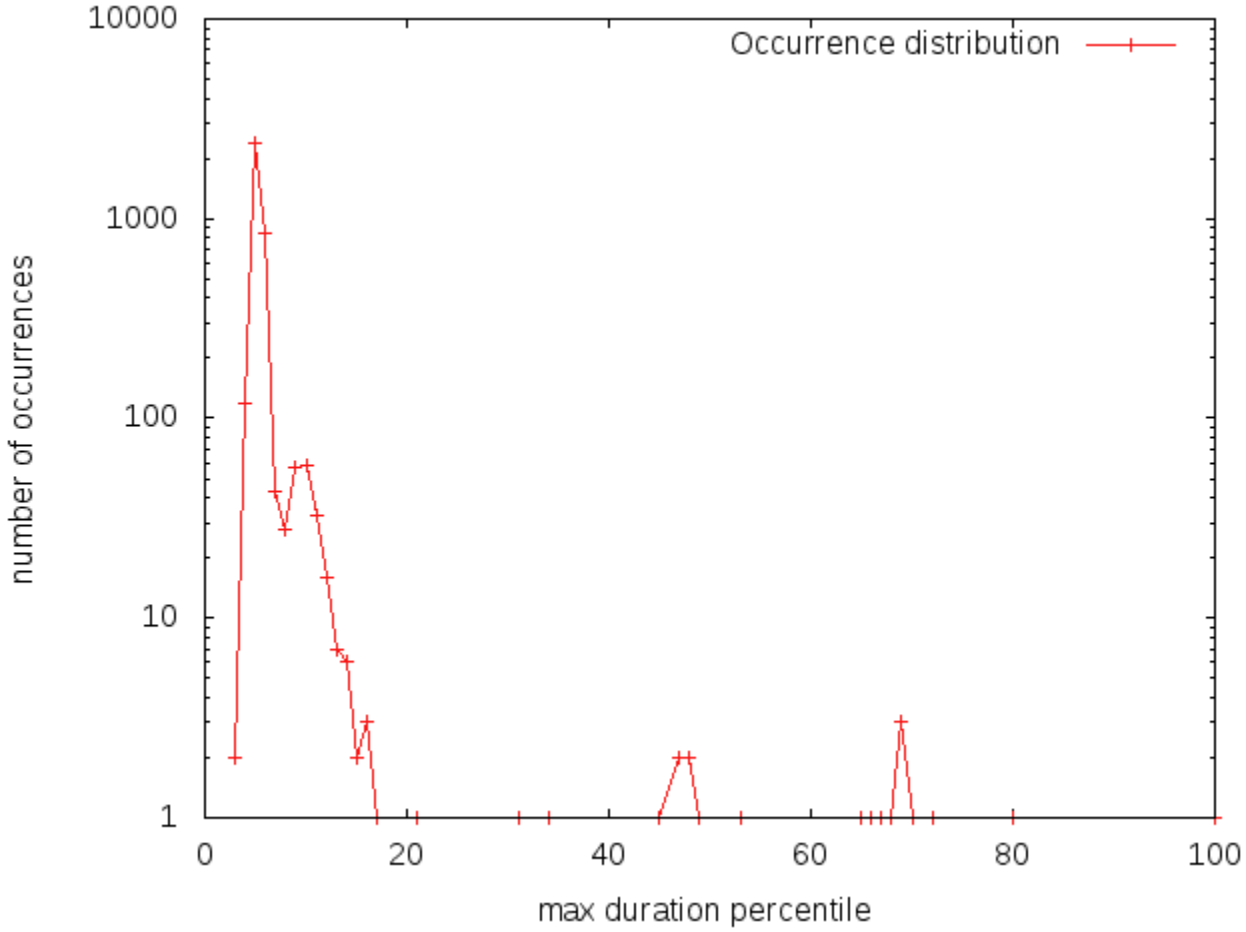
- **Définition :**

```
motif #1234 {
   séquence 1
   3 x séquence 2
   séquence 3
}
```

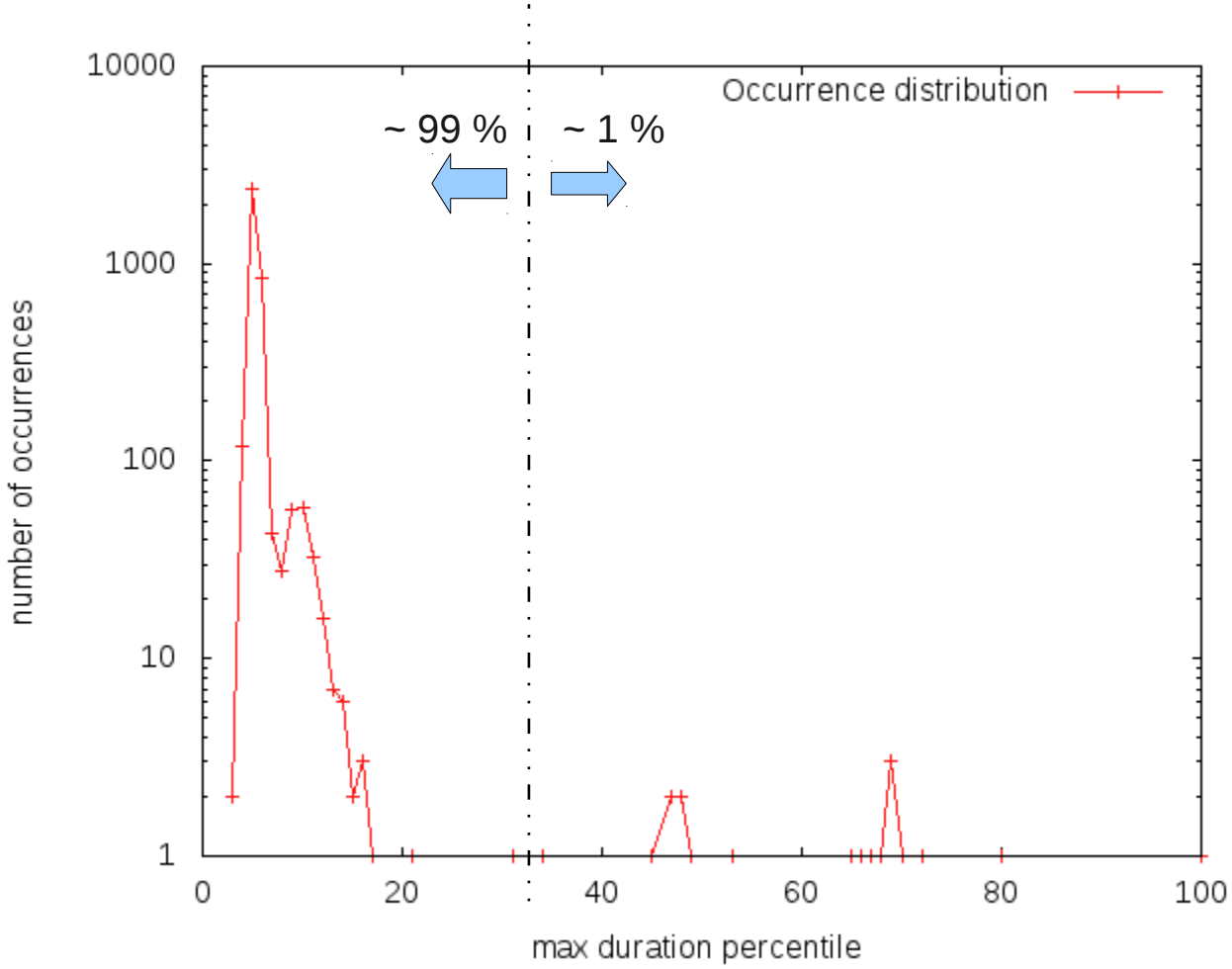
- **Nombre d'occurrences, durées des occurrences, etc**

Détection des anomalies

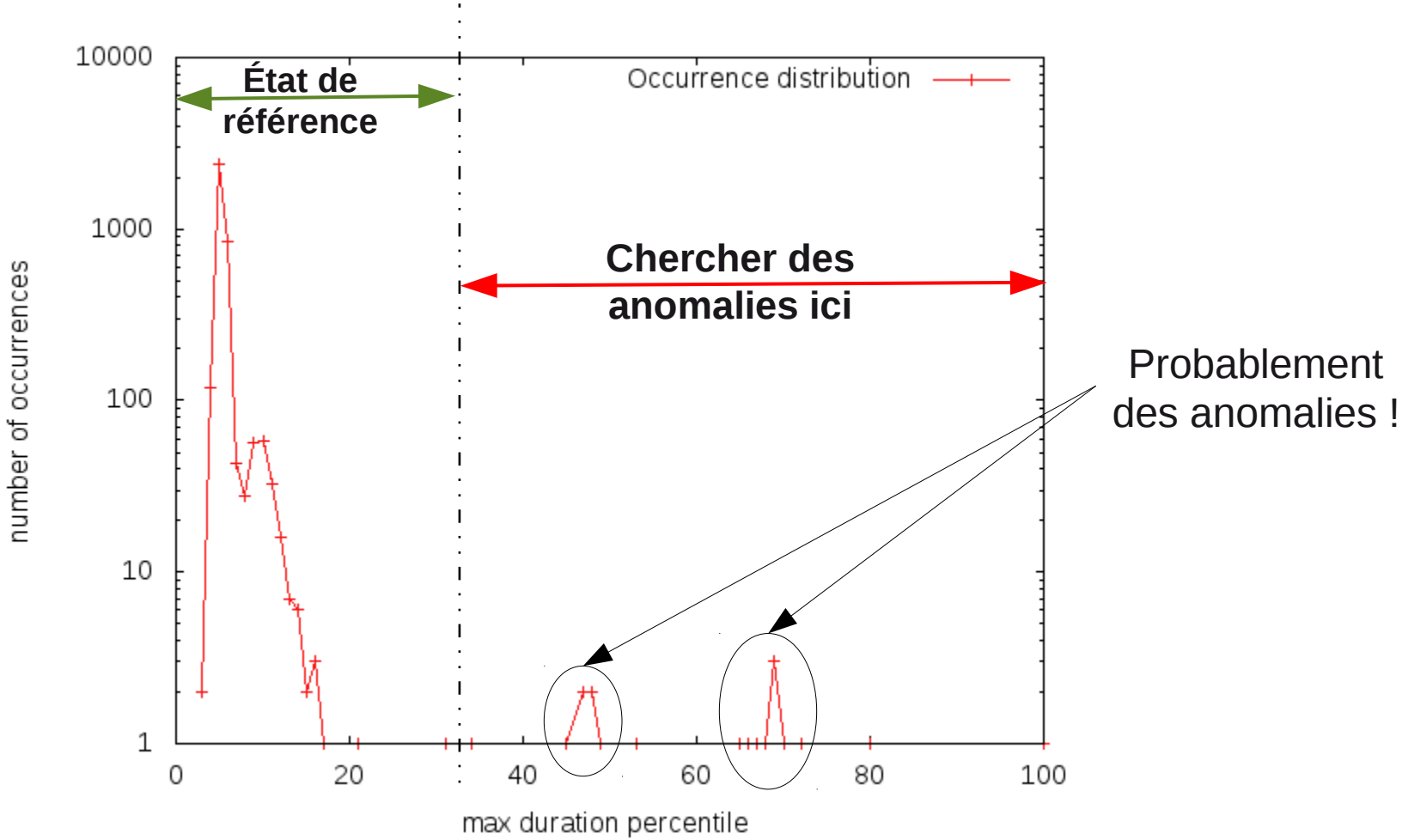
Courbe de distribution des durées



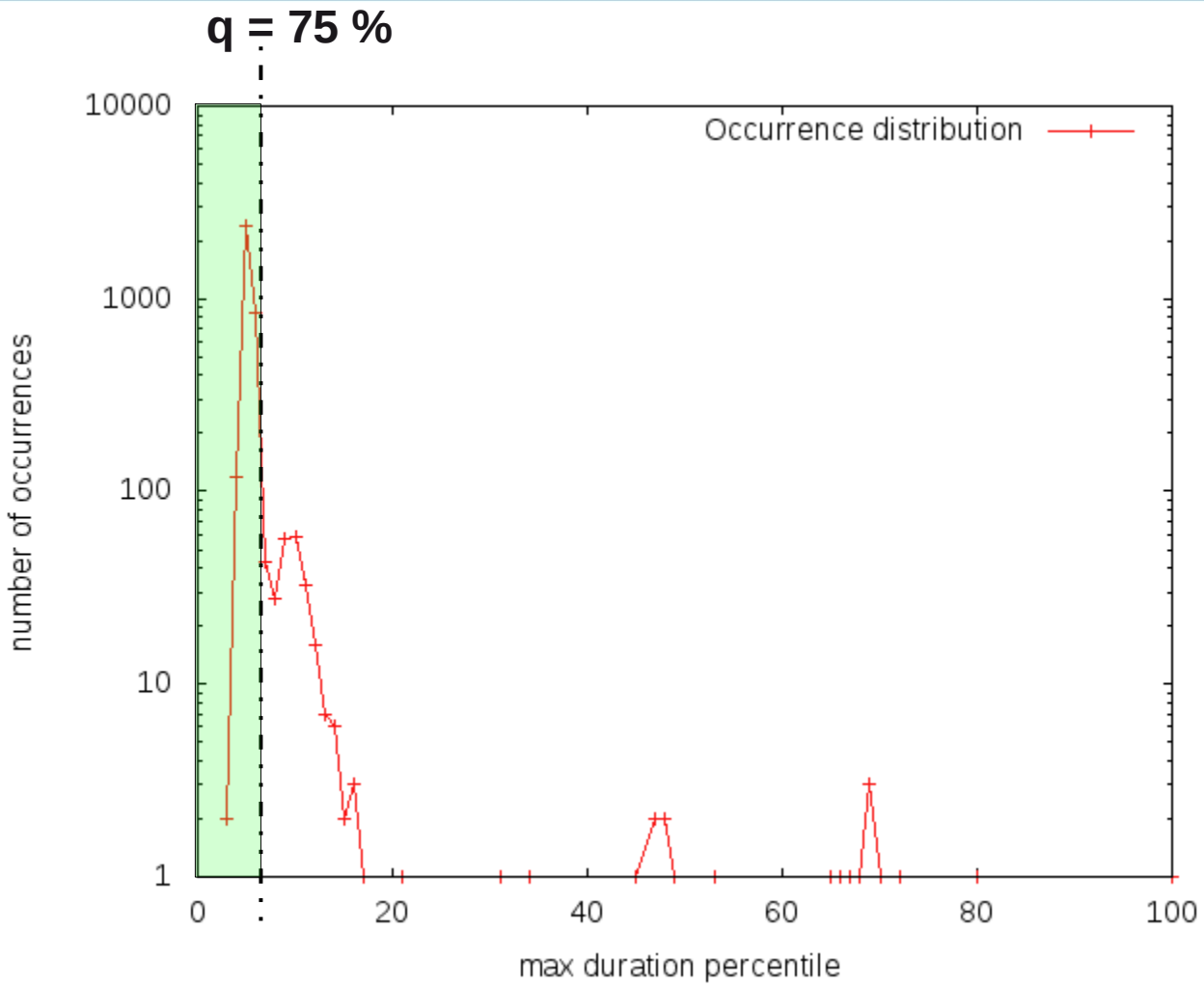
Courbe de distribution des durées



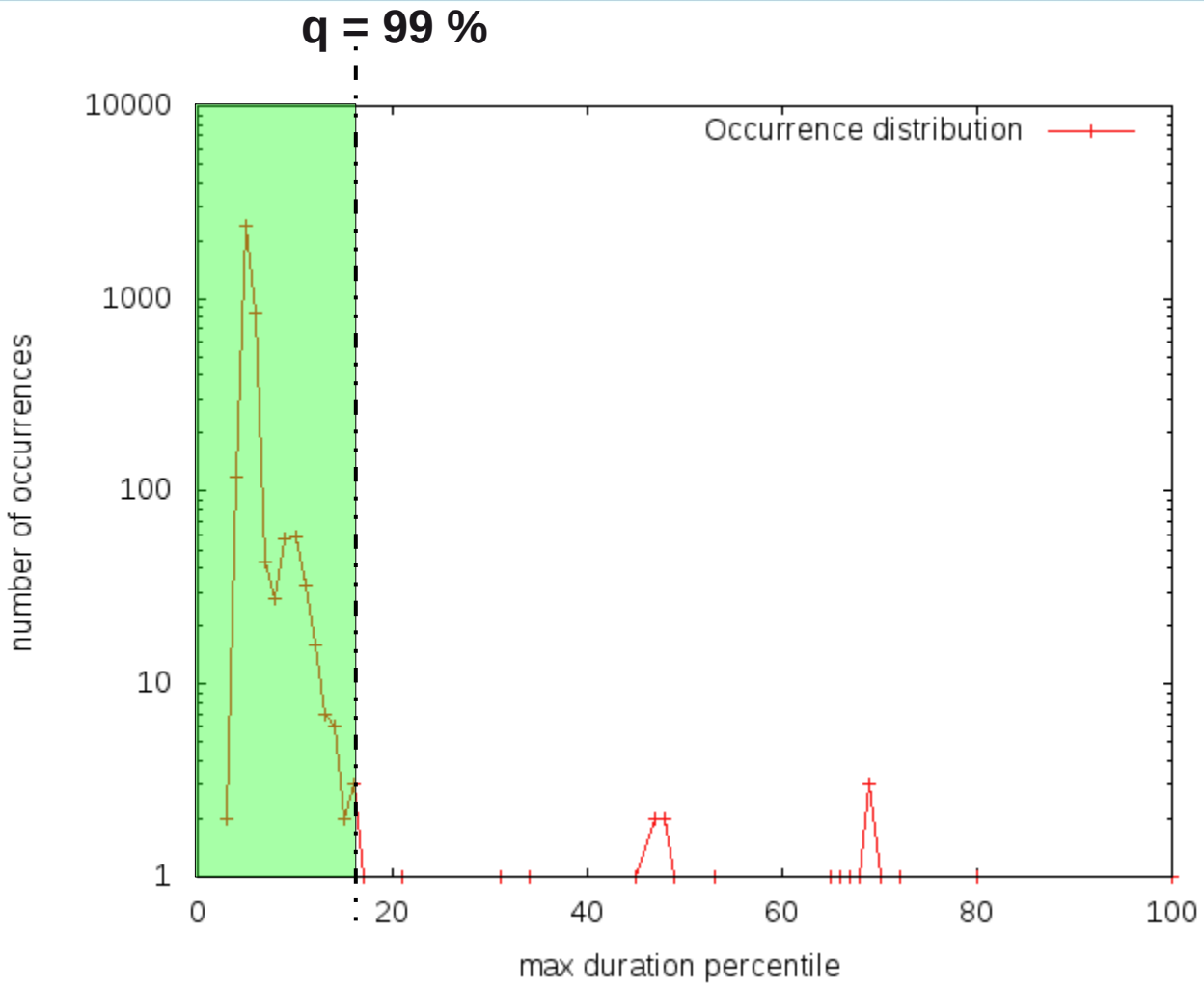
Courbe de distribution des durées



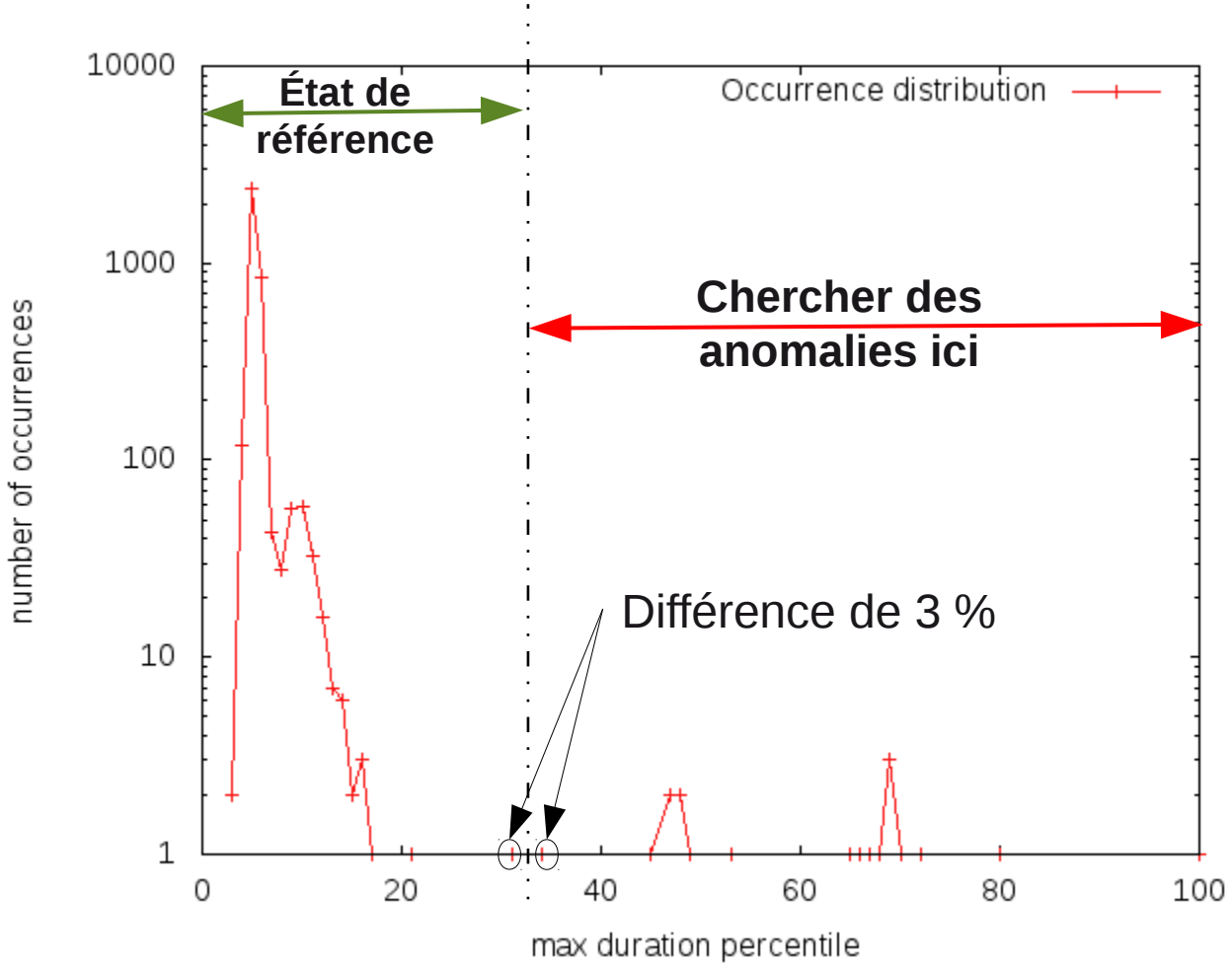
q : taille de l'ensemble de référence



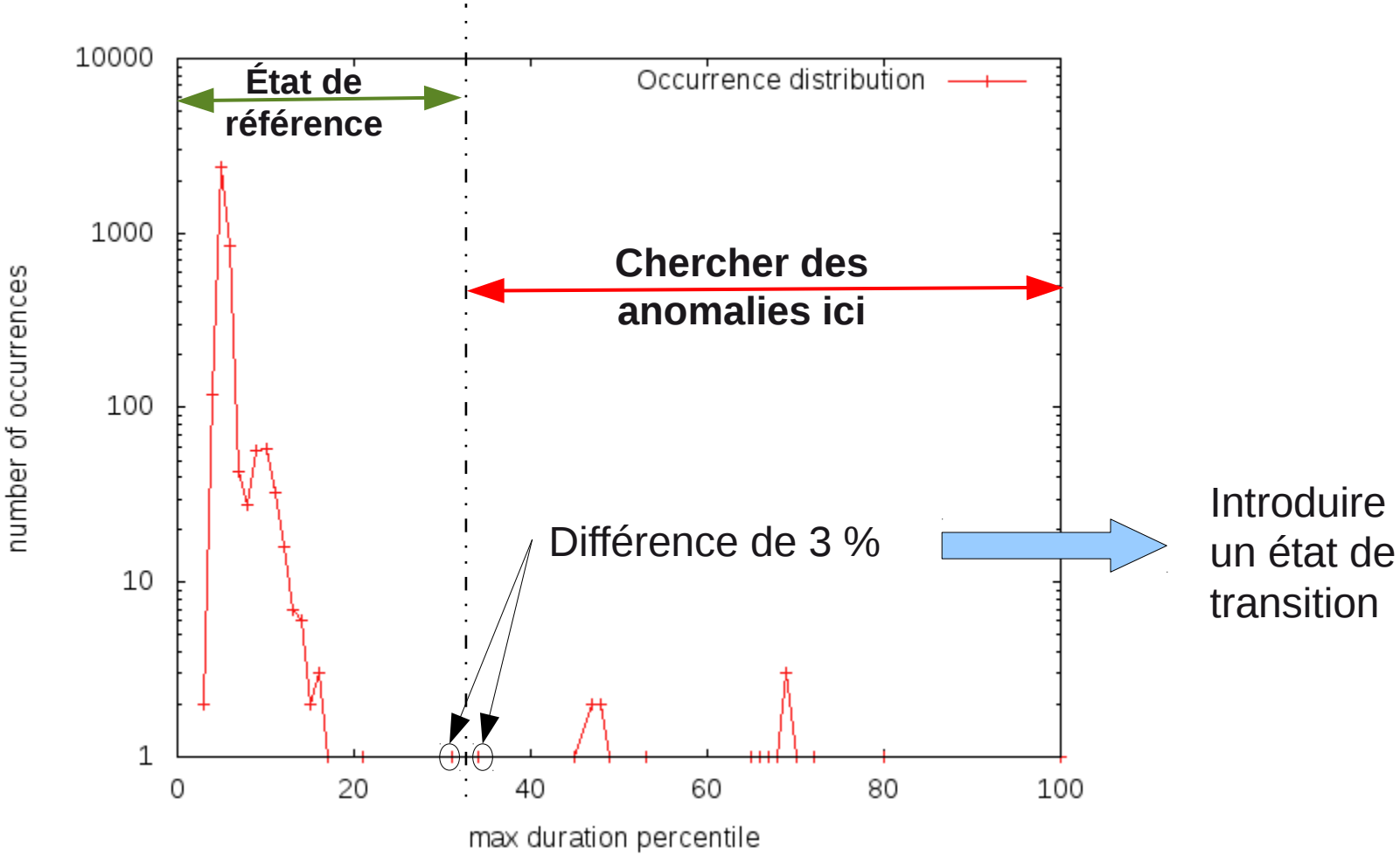
q : taille de l'ensemble de référence



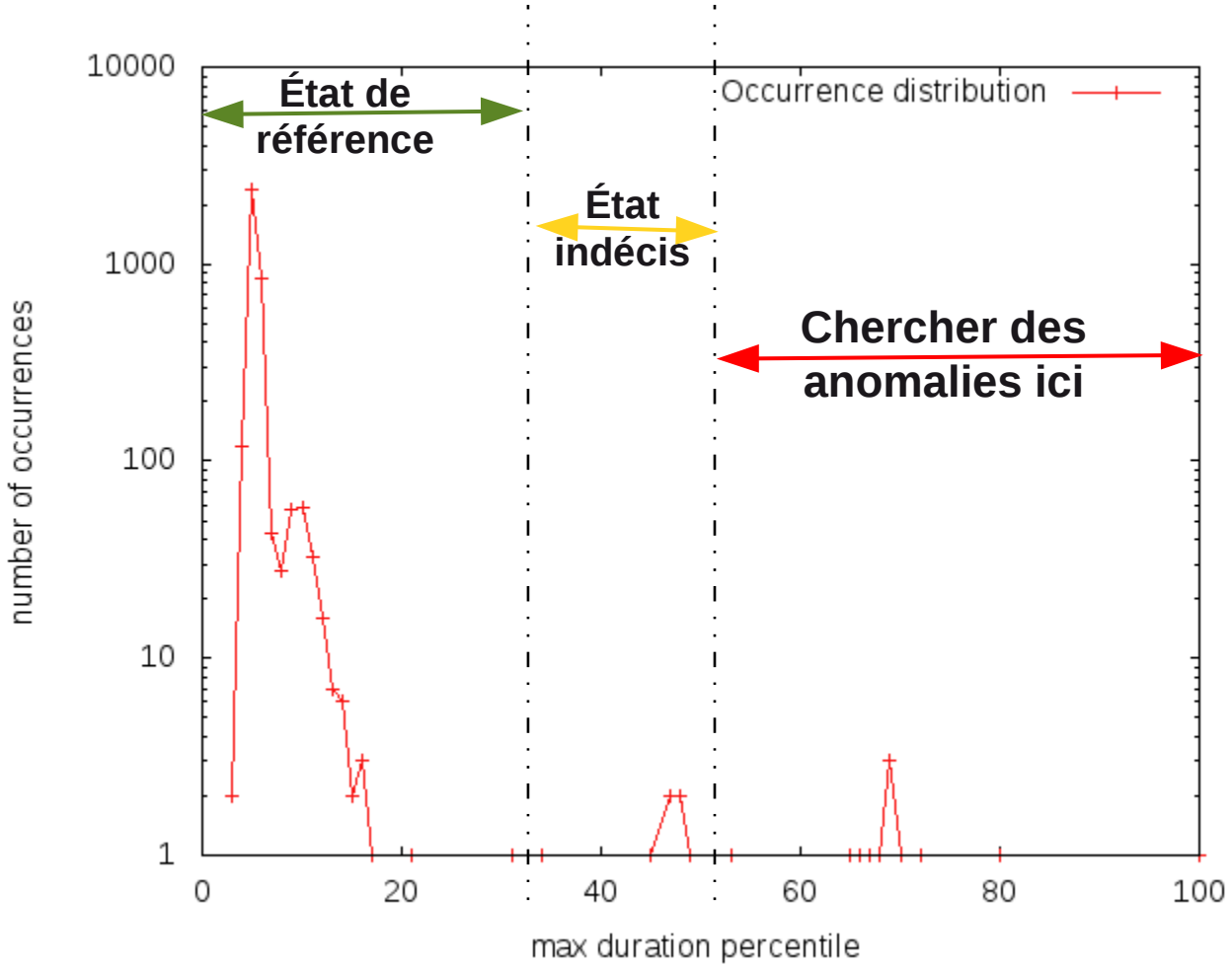
Courbe de distribution des durées



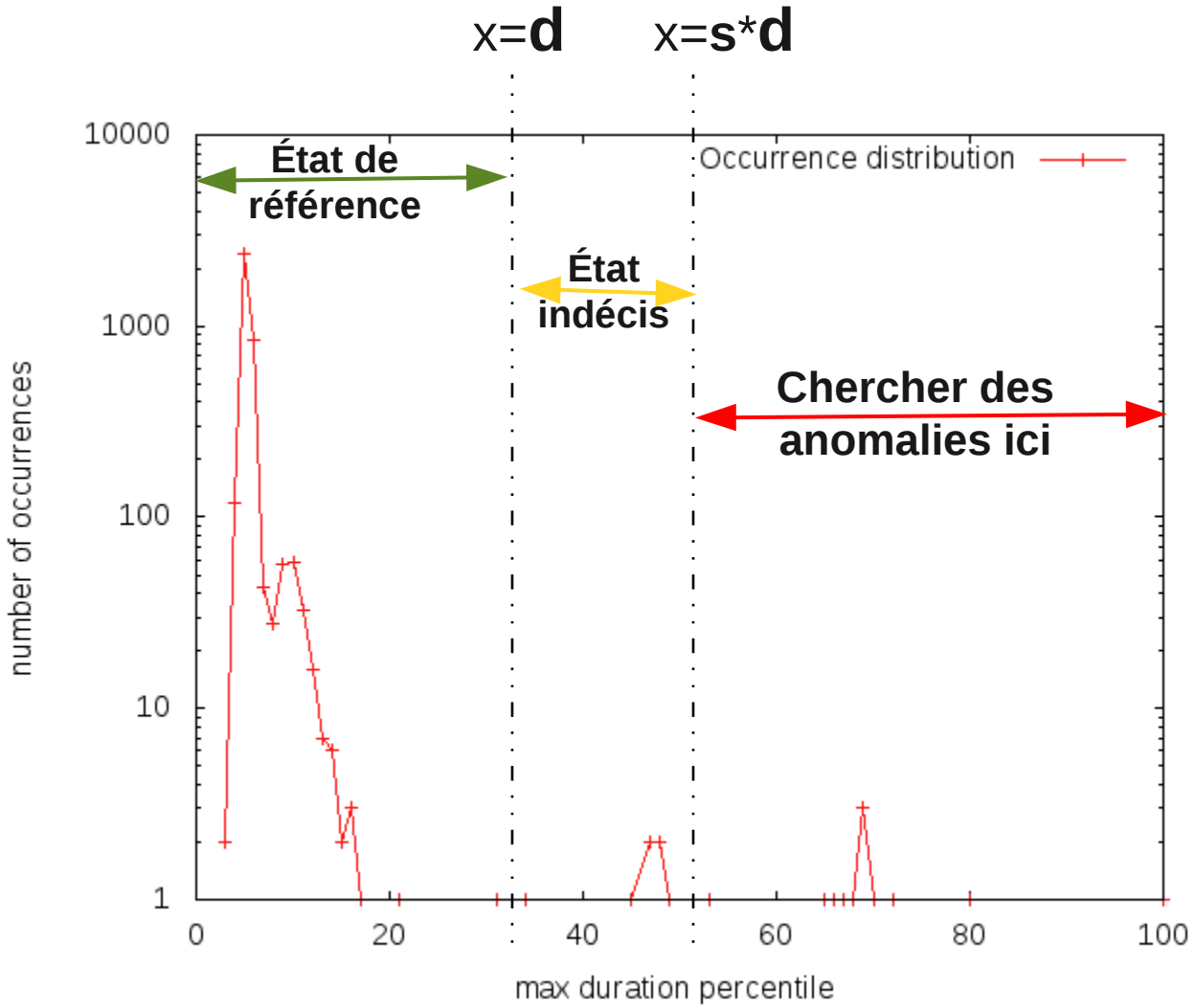
Courbe de distribution des durées



Courbe de distribution des durées



s : seuil d'anomalie



(Motifs + Anomalies) détectés

- **Définition :**

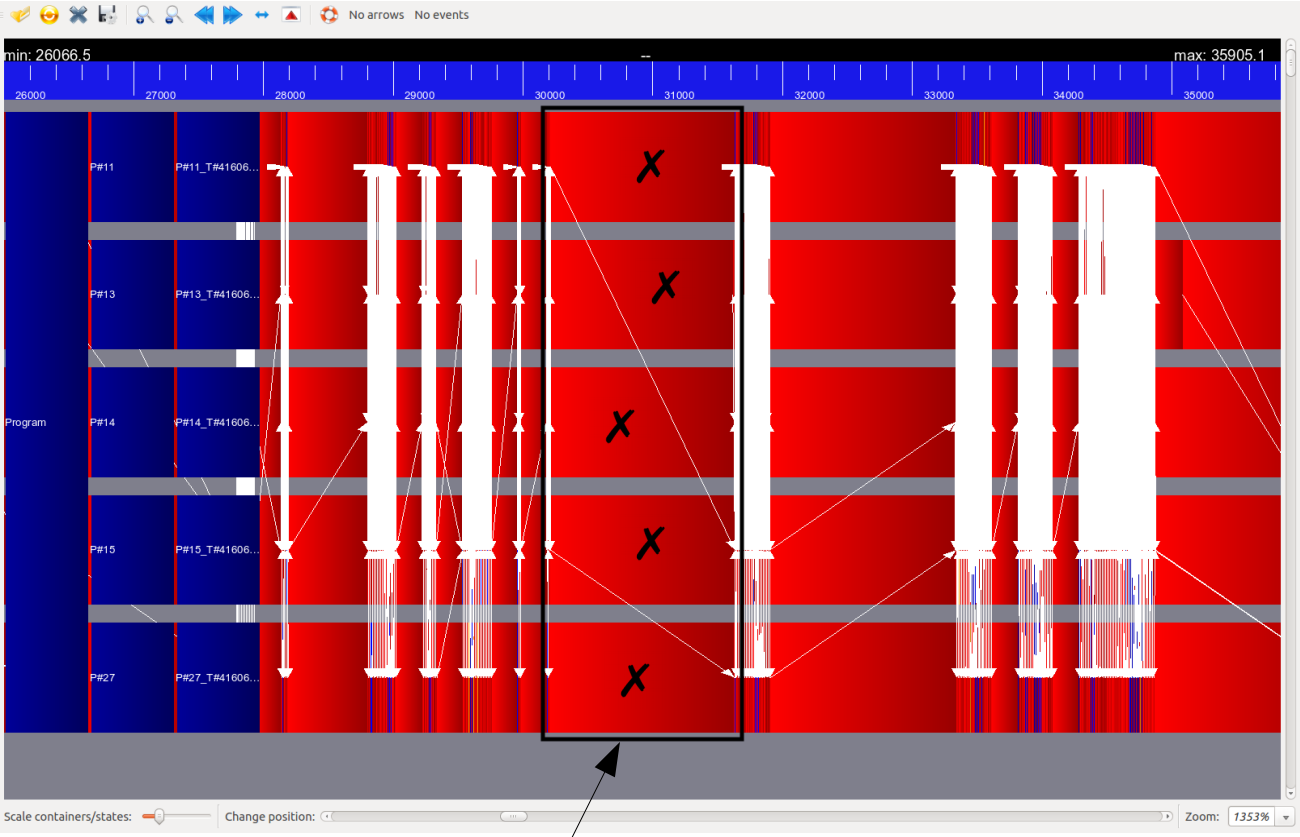
```
motif #1234 {
   séquence 1
   3 x séquence 2
   séquence 3
}
```

- **Durée de référence : 20**

- **Liste des occurrences anormales :**
  - Occ **11**, t = 123.45, durée = 2438
  - Occ **54**, t = 9999.87, durée = 570
  - etc

Détection des sources d’anomalies

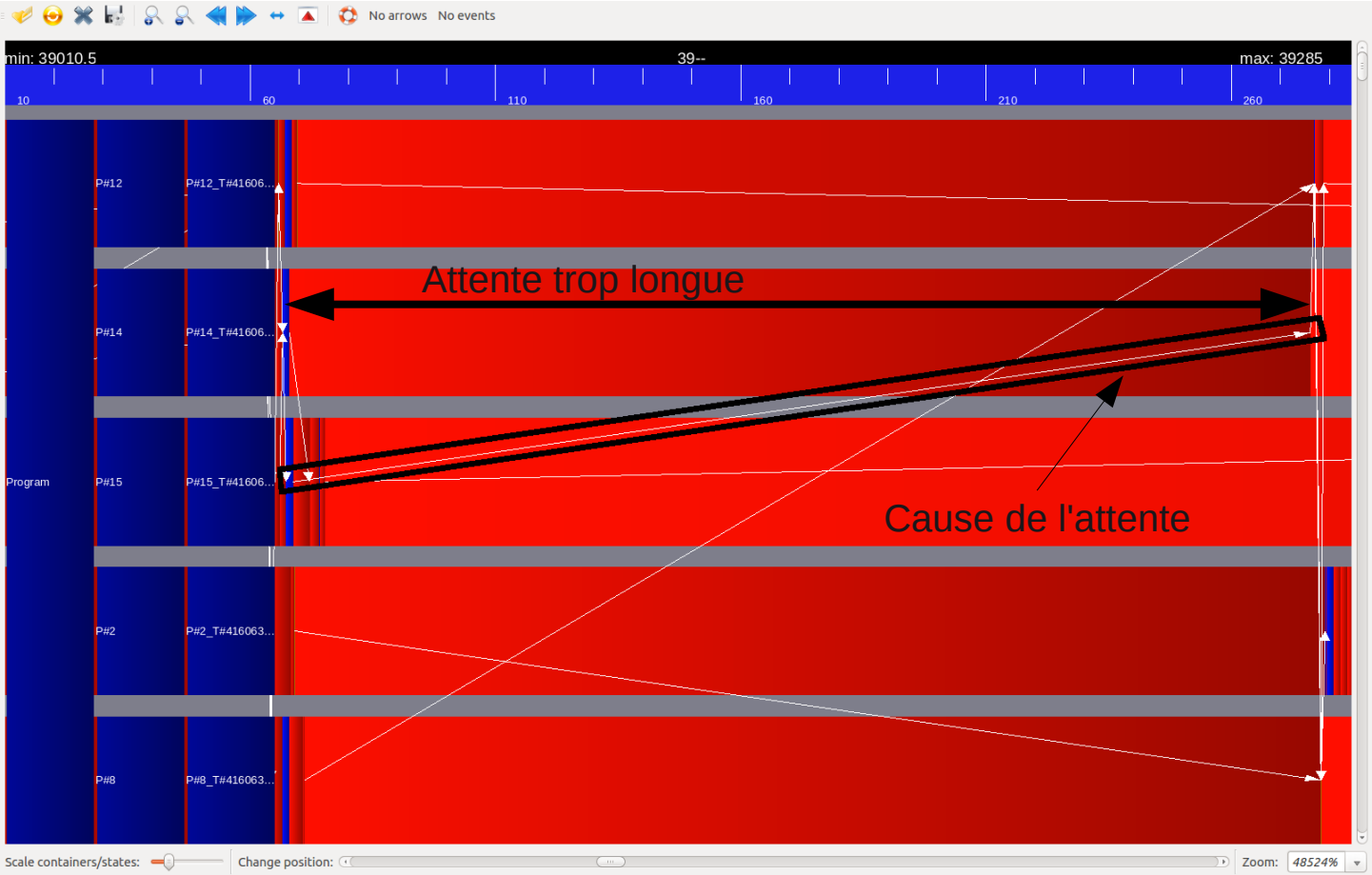
Des anomalies… à leurs sources



Concentration d'anomalies → Des relations de causalité ?

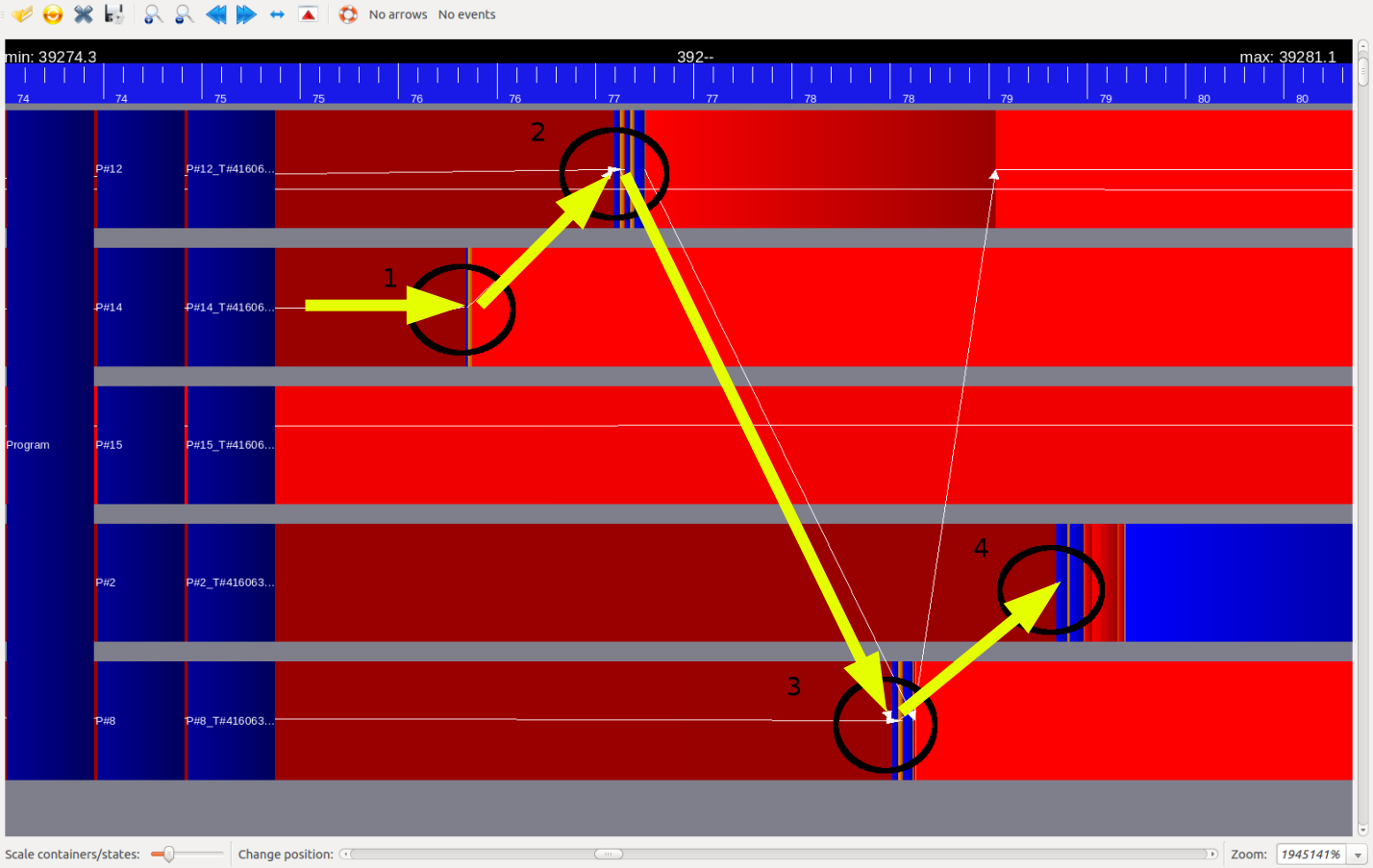
Recherche des causes racines : motivation

Exemple d'anomalie racine (i.e. source de problème)



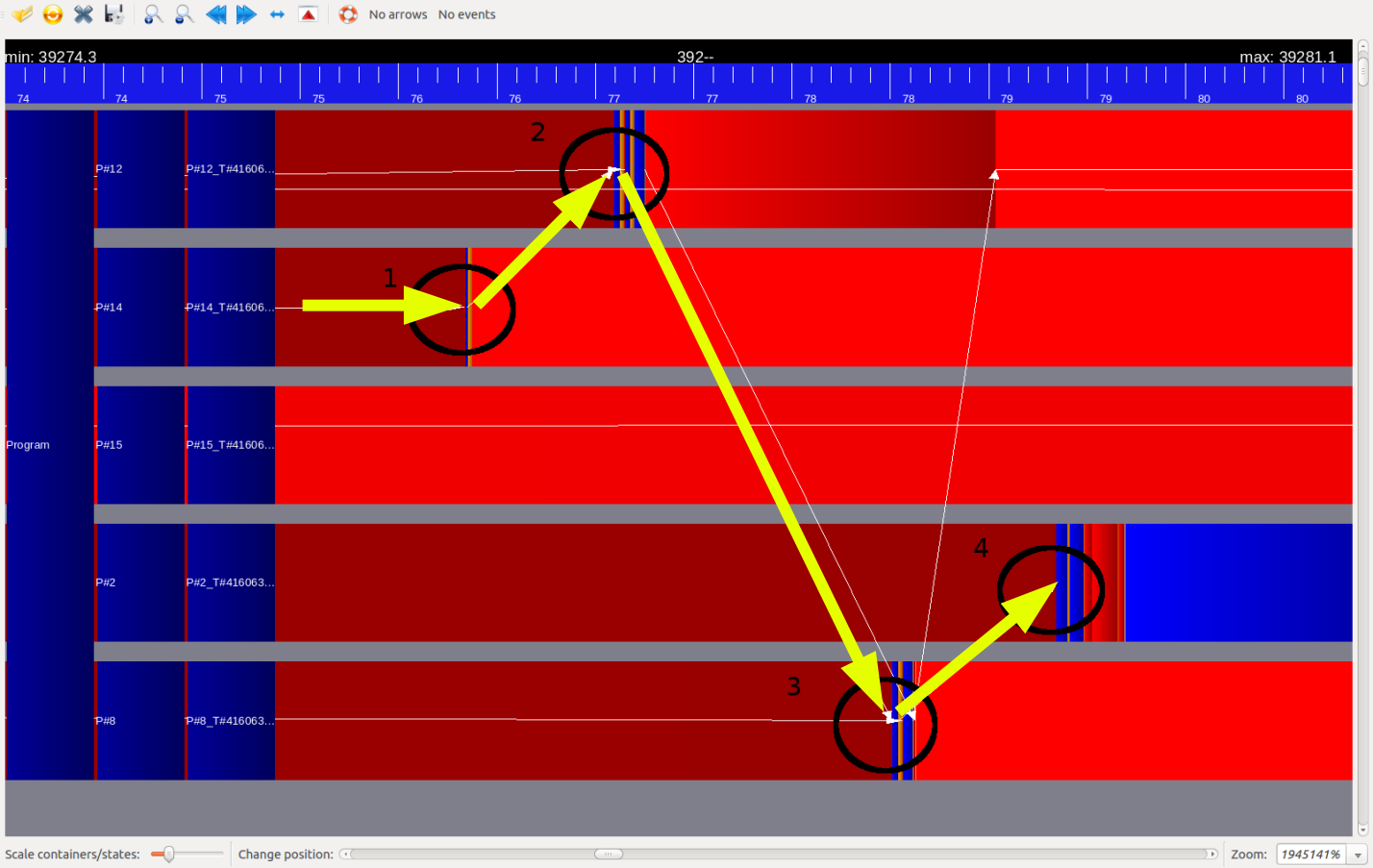
Recherche des causes racines : motivation

D'autres processus sont affectés...



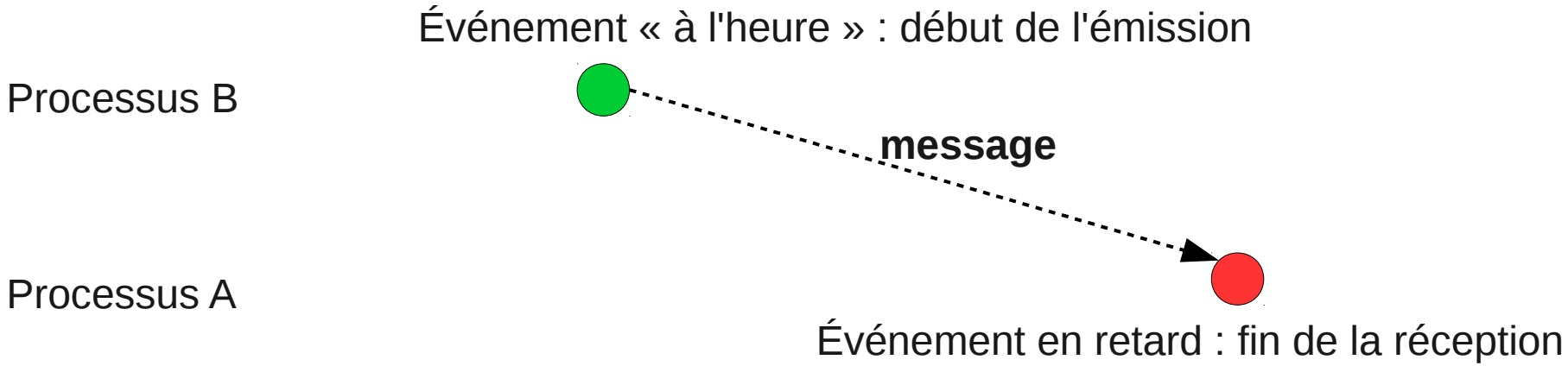
Recherche des causes racines : motivation

D'autres processus sont affectés...

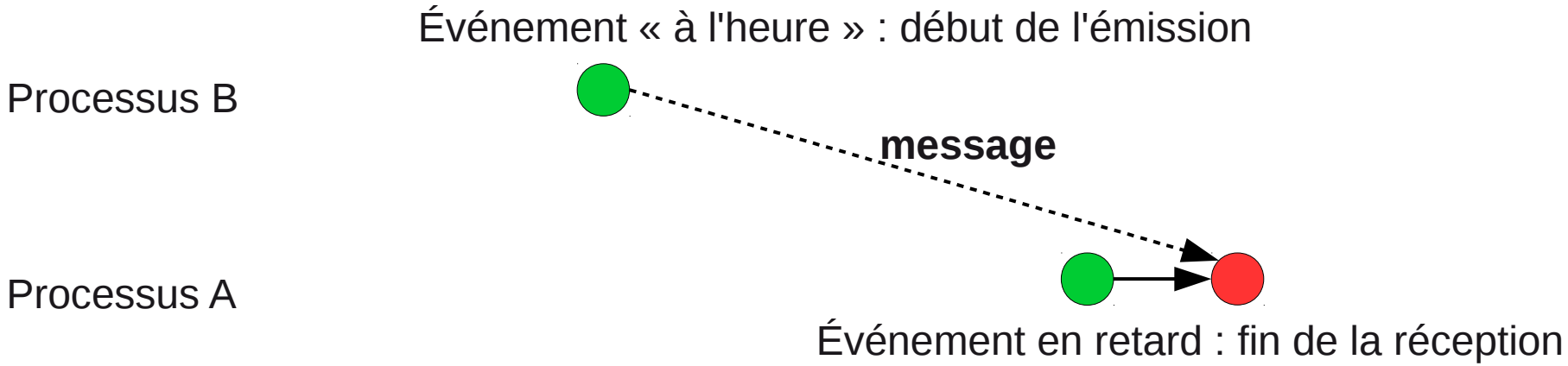


Il faut détecter ce genre de « cascades »

Détection des sources d'anomalies



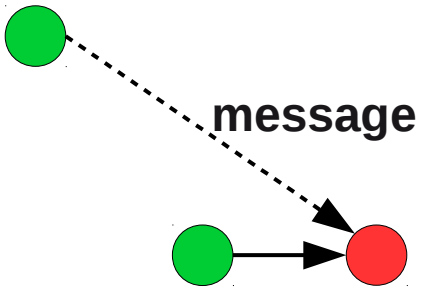
Détection des sources d'anomalies



Détection des sources d'anomalies

Événement « à l'heure » : début de l'émission

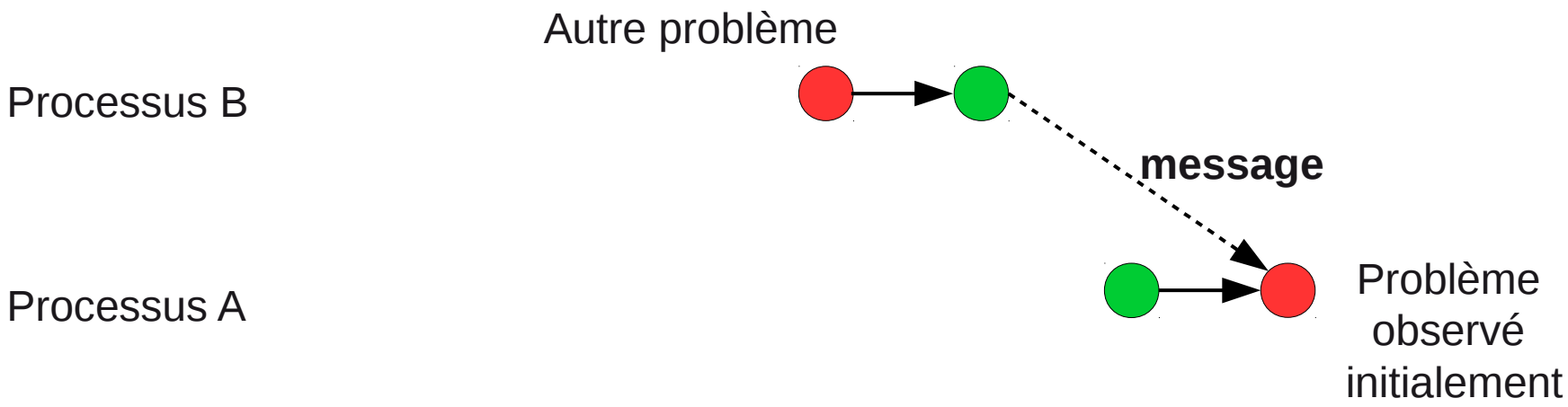
Processus B



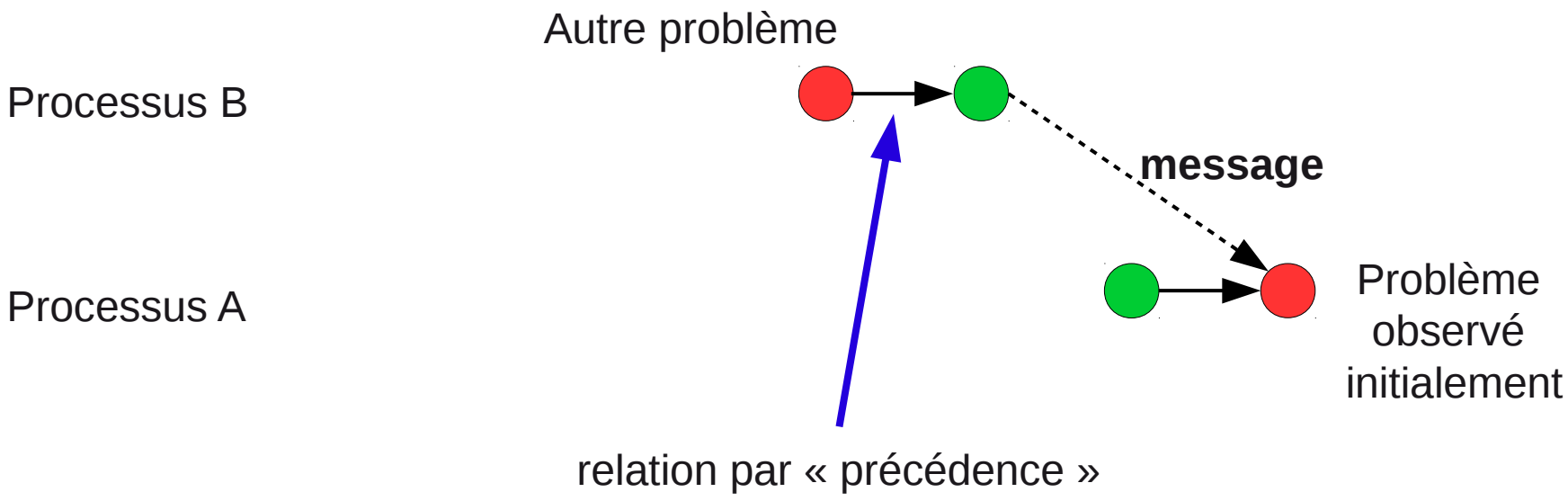
Processus A

Événement en retard : fin de la réception

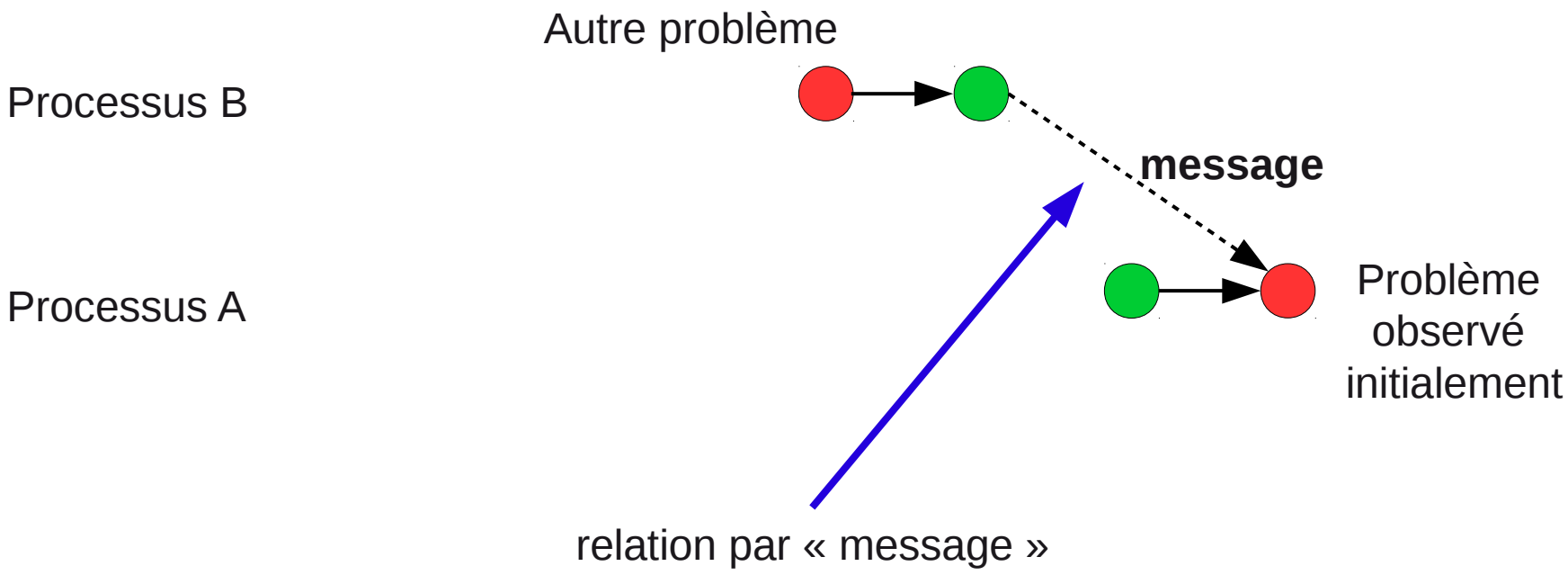
Détection des sources d'anomalies



Détection des sources d'anomalies

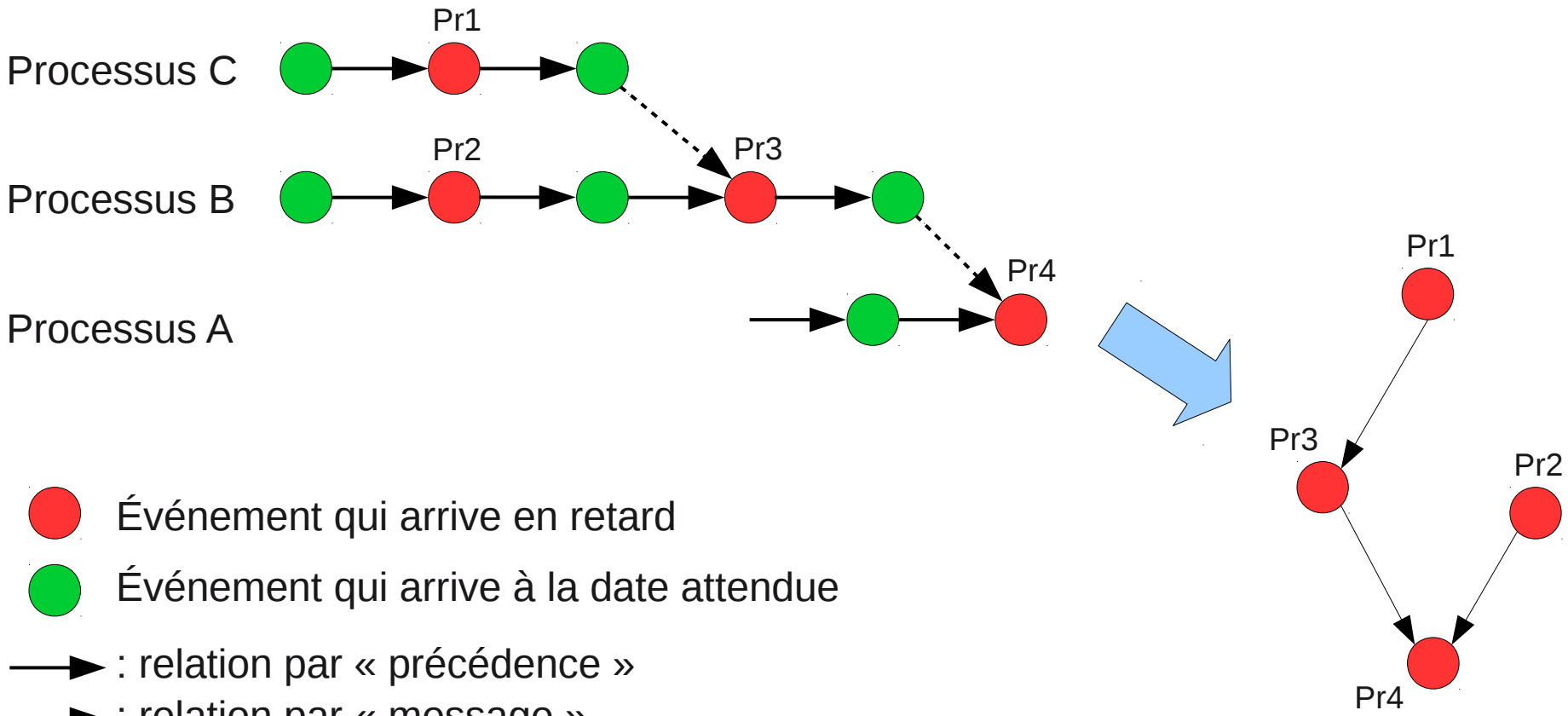


Détection des sources d'anomalies



Détection des sources d'anomalies

Relations de causalité



Événement qui arrive en retard

Événement qui arrive à la date attendue

: relation par « précédence »

: relation par « message »

Détection d’anomalies : Évaluation préliminaire

Évaluation préliminaire

- **Objectifs :**
  - Tester la détection des motifs
  - Tester la détection d'anomalies
  - Tester la détection des causes racines

- **Implémentation dans eztrace**

- **Application sur des traces NAS Parallel Benchmark (NPB)**
  - Version MPI
  - Classe A, 16 processus

Évaluation préliminaire

kernel	*#événements*	*détection (ms)*	*#motifs*	*#motifs sélectionnés*	*#problèmes*	*#sources*	l_{max}
EP	3 090	3.51	32	31 (97%)	32	0	1
FT	10 256	9.44	80	19 (24%)	19	0	1
IS	18 552	35.43	48	0	–	–	–
CG	284 754	178.23	160	65 (40%)	611	34	3
MG	118 688	186.17	2 728	494 (18%)	5 357	350	2
SP	557 318	596.84	174	48 (28%)	2 132	0	1
BT	399 944	951.51	112	66 (59%)	1 367	159	15
LU	4 568 002	4 564.80	210	101 (48%)	13 721	2372	6

TABLE 1 – Statistiques sur la recherche de problèmes de performance dans les traces NPB (Classe=A, NProcs=16) - seuil d'anomalie s=2

Détection des motifs

kernel	*#événements*	*détection (ms)*	*#motifs*	*#motifs sélectionnés*	*#problèmes*	*#sources*	l_{max}
EP	3 090	3.51	32	31 (97%)	32	0	1
FT	10 256	9.44	80	19 (24%)	19	0	1
IS	18 552	35.43	48	0	–	–	–
CG	284 754	178.23	160	65 (40%)	611	34	3
MG	118 688	186.17	2 728	494 (18%)	5 357	350	2
SP	557 318	596.84	174	48 (28%)	2 132	0	1
BT	399 944	951.51	112	66 (59%)	1 367	159	15
LU	4 568 002	4 564.80	210	101 (48%)	13 721	2372	6

TABLE 1 – Statistiques sur la recherche de problèmes de performance dans les traces NPB (Classe=A, NProcs=16) - seuil d'anomalie s=2

⇒ **temps de détection des motifs < 5 s**

⇒ **# motifs détectés varie de quelques dizaines à quelques centaines**

Détection des anomalies

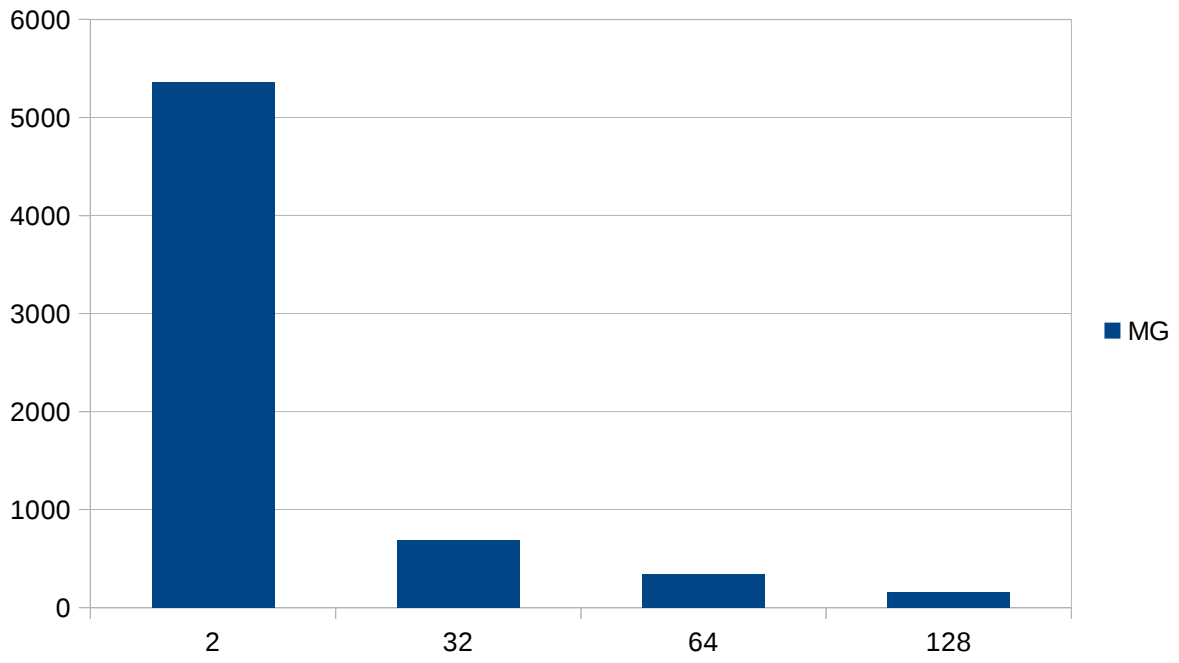
kernel	*#événements*	*détection (ms)*	*#motifs*	*#motifs sélectionnés*	*#problèmes*	*#sources*	l_{max}
EP	3 090	3.51	32	31 (97%)	32	0	1
FT	10 256	9.44	80	19 (24%)	19	0	1
IS	18 552	35.43	48	0	–	–	–
CG	284 754	178.23	160	65 (40%)	611	34	3
MG	118 688	186.17	2 728	494 (18%)	5 357	350	2
SP	557 318	596.84	174	48 (28%)	2 132	0	1
BT	399 944	951.51	112	66 (59%)	1 367	159	15
LU	4 568 002	4 564.80	210	101 (48%)	13 721	2372	6

TABLE 1 – Statistiques sur la recherche de problèmes de performance dans les traces NPB (Classe=A, NProcs=16) - seuil d'anomalie s=2

⇒ # anomalies détectées varie de quelques dizaines à quelques milliers

Détection des anomalies

Nombre d'anomalies en fonction de s (NPB-MG, classe A, 16 processus)



Détection des sources d'anomalies

kernel	*#événements*	*détection (ms)*	*#motifs*	*#motifs sélectionnés*	*#problèmes*	*#sources*	l_{max}
EP	3 090	3.51	32	31 (97%)	32	0	1
FT	10 256	9.44	80	19 (24%)	19	0	1
IS	18 552	35.43	48	0	–	–	–
CG	284 754	178.23	160	65 (40%)	611	34	3
MG	118 688	186.17	2 728	494 (18%)	5 357	350	2
SP	557 318	596.84	174	48 (28%)	2 132	0	1
BT	399 944	951.51	112	66 (59%)	1 367	159	15
LU	4 568 002	4 564.80	210	101 (48%)	13 721	2372	6

TABLE 1 – Statistiques sur la recherche de problèmes de performance dans les traces NPB (Classe=A, NProcs=16) - seuil d'anomalie s=2

⇒ **nombre non négligeable d'anomalies qui se propagent**

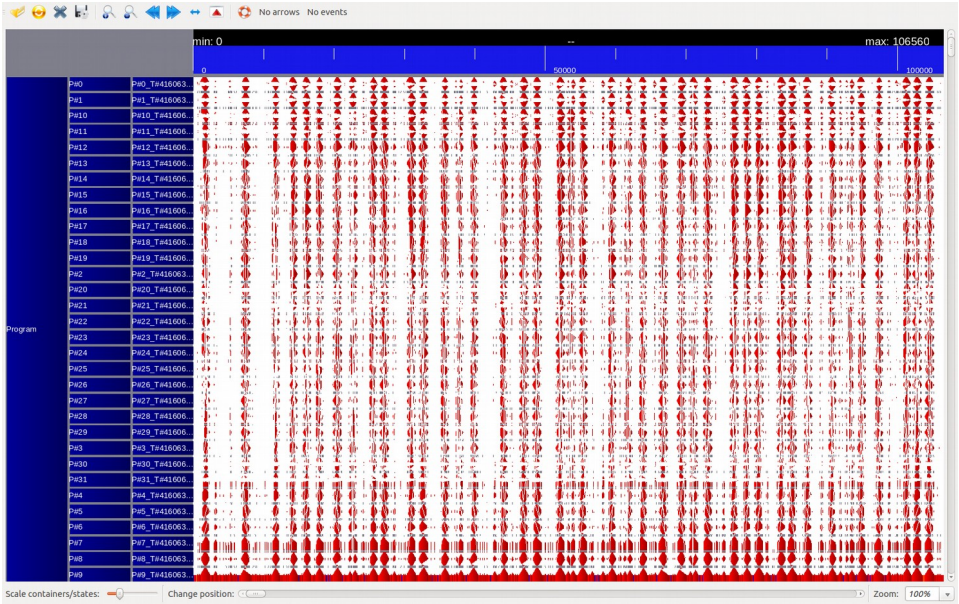
⇒ **chemins problématiques assez longs**

Cas d'utilisation : trace sous forme brute

- **Application : NPB-CG, 32 processus, classe B**

- **Affichage brut : tous les processus, tous les événements**

Les 32 processus



Cas d'utilisation : aperçu des résultats de l'analyse

- **On repère un chemin problèmatique long**
  - Chaque ligne représente une séquence problématique

```
83370
83371 Printing a problematic path... length: 4
83372 seq[@(t=39069.253042,C=P#15_T#4160637728,green@39069.253042,red@39069.286244)] ->
83373 seq[@(t=39068.486996,C=P#14_T#4160637728,green@39068.512087,red@39276.603688)] ->
83374 seq[@(t=39068.948573,C=P#12_T#4160637728,green@39070.081805,red@39277.362556)] ->
83375 seq[@(t=39064.009049,C=P#18_T#4160637728,green@39071.038224,red@39278.922616)] ->
83376
83377 Printing a problematic path... length: 5
83378 seq[@(t=39069.253042,C=P#15_T#4160637728,green@39069.253042,red@39069.286244)] ->
83379 seq[@(t=39068.486996,C=P#14_T#4160637728,green@39068.512087,red@39276.603688)] ->
83380 seq[@(t=39068.948573,C=P#12_T#4160637728,green@39070.081805,red@39277.362556)] ->
83381 seq[@(t=39068.432521,C=P#8_T#4160637728,green@39071.633487,red@39278.773367)] ->
83382 seq[@(t=39063.610377,C=P#2_T#4160637728,green@39069.560624,red@39279.614156)] ->
83383
83384 Printing a problematic path... length: 3
83385 seq[@(t=39069.253042,C=P#15_T#4160637728,green@39069.253042,red@39069.286244)] ->
83386 seq[@(t=39068.486996,C=P#14_T#4160637728,green@39068.512087,red@39276.603688)] ->
83387 seq[@(t=39062.959138,C=P#26_T#4160637728,green@39068.240217,red@40496.835832)] ->
83388
```

Longueur du chemin

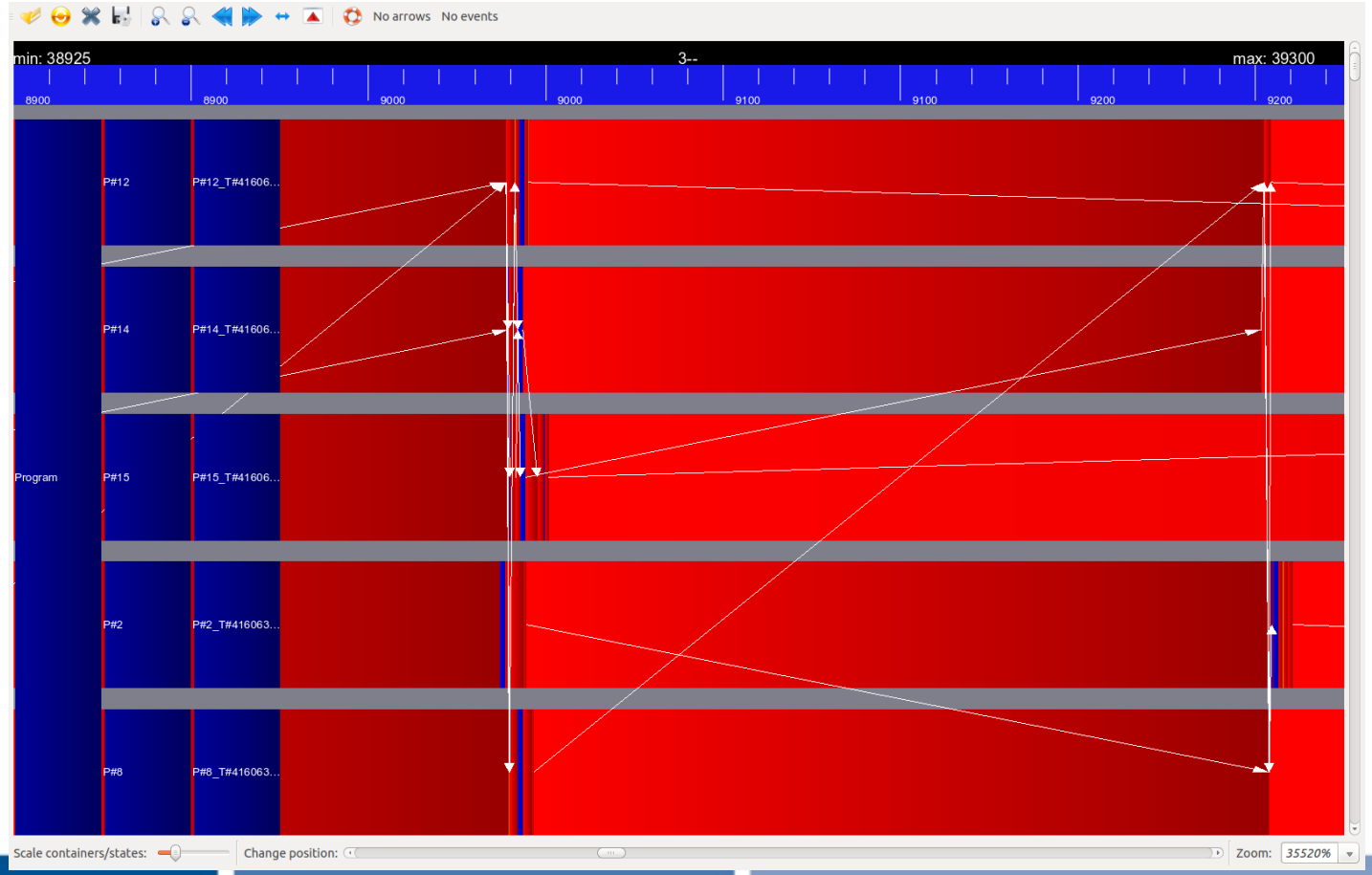
id du processus

Date de l'événement anormal

Date de l'événement suivant

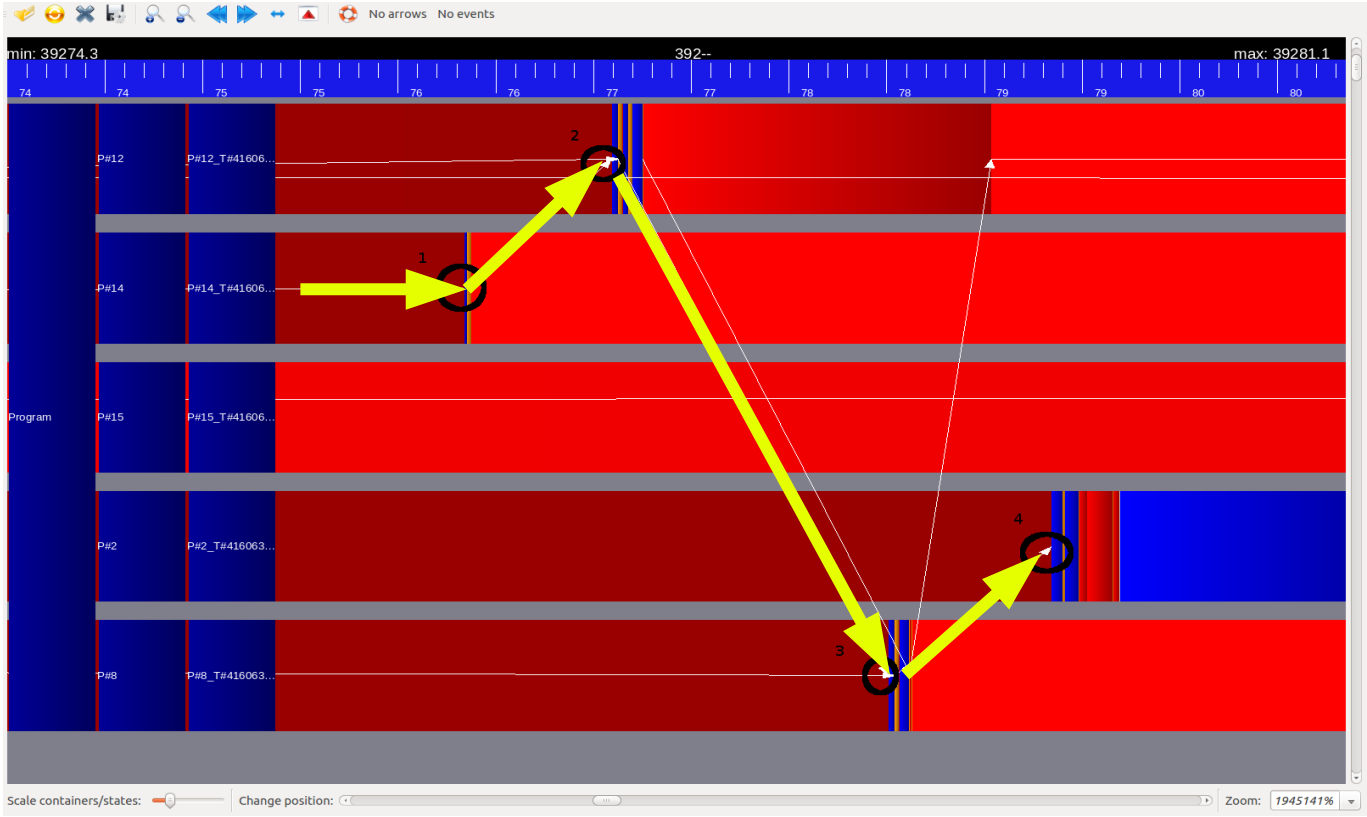
Cas d'utilisation : vue restreinte

- **Sélection des processus impliqués et de la fenêtre temporelle adéquate**



Cas d'utilisation : visualisation du chemin problématique

- les points de synchronisation exacts qui ont permis à l'anomalie de se propager



Conclusion

- **Héritage : outils de traces**
- **Idée : détection de motifs ⇒ détection d'anomalies ⇒ détection d'anomalies sources**
- **Résultats expérimentaux encourageants**
- **Travaux futurs :**
  - Faux positifs, faux négatifs
  - Extension à d'autres applications (e.g. architectures client/serveur)