Catch Me If You Can ?

Markov Time Parallel Sampling

{Marion.Dalle,Florence.Perronnin,Jean-Marc.Vincent}@imag.fr

Laboratoire d'Informatique de Grenoble,
Inria team MESCAL
University Grenoble-Alpes, France

Marmote Workshop, October 8-9

Outline

1. **Generation of Long Trajectories**

2. Time parallel simulation

3. Catch Me If You Can

4. psi3

5. Synthesis

Motivations

Applications

- Finite queuing networks (dynamic routing)
- Call centers
- Grid/cluster scheduling
- Rare event estimation
- Statistical verification of program

Models

- Discrete vector state-space $\mathcal{X}$
- Event based models

$$X_{n+1} = \Phi(X_n, e_{n+1}) \ , e_n \in \mathcal{E}$$

  Stochastic recurrence equation
- Independent events (iid)

Provide **long** trajectories of **stationary** states.

PSI2 : a Perfect Sampler

- Library of events (**monotone**, bounded,...)
- Simulation kernel
- Efficient simulator : polynomial in the model dimension

$\Longrightarrow$ **Extension time parallel sampling**

Motivations

Applications

- Finite queuing networks (dynamic routing)
- Call centers
- Grid/cluster scheduling
- Rare event estimation
- Statistical verification of program

Models

- Discrete vector state-space $\mathcal{X}$
- Event based models

$$X_{n+1} = \Phi(X_n, e_{n+1}) \ , e_n \in \mathcal{E}$$

Stochastic recurrence equation

- Independent events (iid)

Provide **long** trajectories of **stationary** states.

PSI2 : a Perfect Sampler

- Library of events (**monotone**, bounded,...)
- Simulation kernel
- Efficient simulator : polynomial in the model dimension

$\Longrightarrow$ **Extension time parallel sampling**

Motivations

Applications

- Finite queuing networks (dynamic routing)
- Call centers
- Grid/cluster scheduling
- Rare event estimation
- Statistical verification of program

Models

- Discrete vector state-space $\mathcal{X}$
- Event based models

$$X_{n+1} = \Phi(X_n, e_{n+1}) \ , e_n \in \mathcal{E}$$

Stochastic recurrence equation

- Independent events (iid)

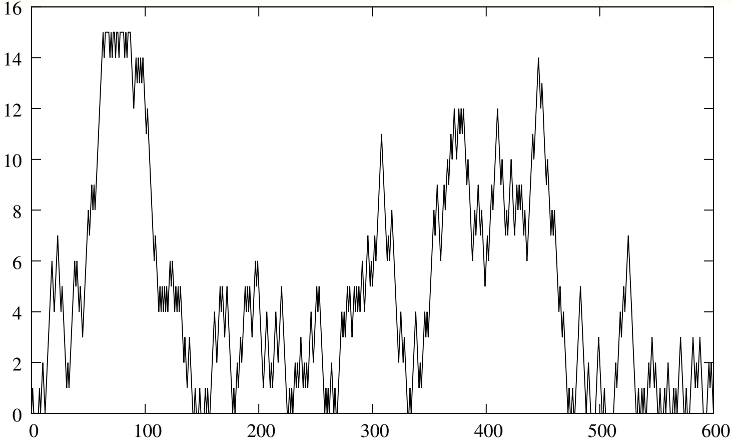
Provide **long** trajectories of **stationary** states.

PSI2 : a Perfect Sampler

- Library of events (**monotone**, bounded,...)
- Simulation kernel
- Efficient simulator : polynomial in the model dimension

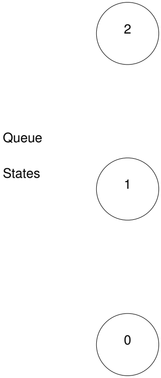
$\Longrightarrow$ **Extension time parallel sampling**

Generation of Long Trajectories

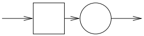


Simulation of a $M/M/1/15$ queue $\rho = 0.9$

Events and Poisson Systems

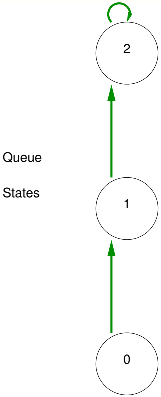


$M/M/1$ capacity $C = 2$

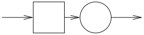


⇒ discrete time sampling

Events and Poisson Systems

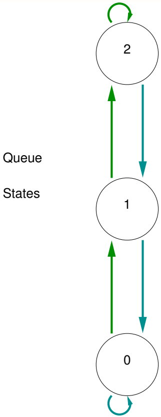


$M/M/1$ capacity $C = 2$

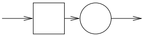


⇒ discrete time sampling

Events and Poisson Systems

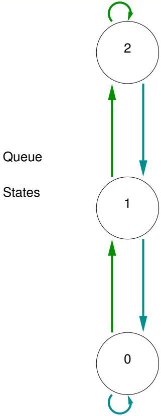


$M/M/1$ capacity $C = 2$

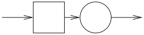


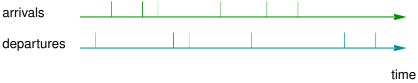
⇒ discrete time sampling

Events and Poisson Systems



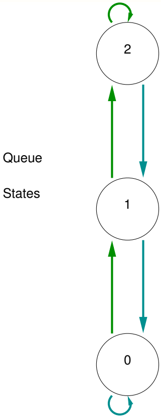
$M/M/1$ capacity $C = 2$



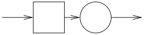


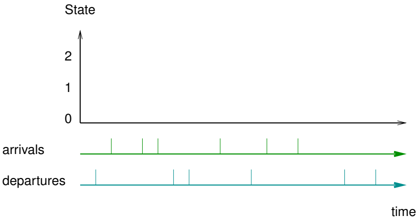
⇒ discrete time sampling

Events and Poisson Systems



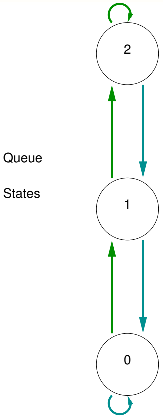
$M/M/1$ capacity $C = 2$



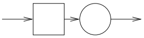


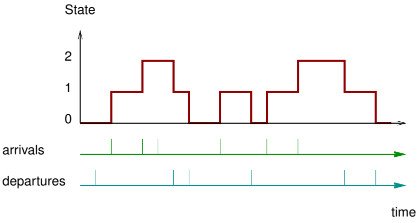
⇒ discrete time sampling

Events and Poisson Systems



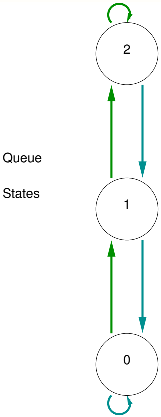
$M/M/1$ capacity $C = 2$



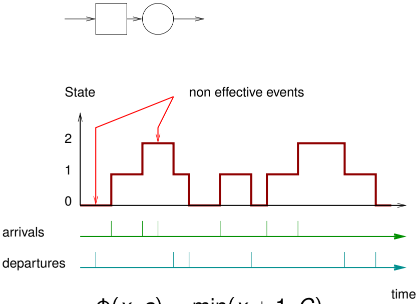


⇒ discrete time sampling

Events and Poisson Systems



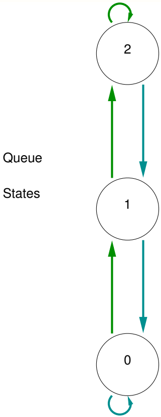
$M/M/1$ capacity $C = 2$



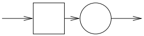
$$\Phi(x, a) = \min(x + 1, C)$$
$$\Phi(x, d) = \max(x - 1, 0)$$

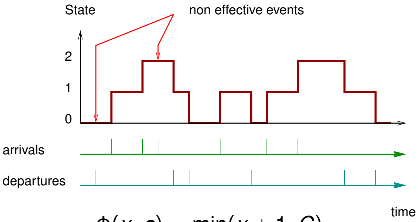
⇒ discrete time sampling

Events and Poisson Systems



$M/M/1$ capacity $C = 2$





$$\Phi(x, a) = \min(x + 1, C)$$
$$\Phi(x, d) = \max(x - 1, 0)$$

⇒ discrete time sampling

Ergodic Sampling

Transition function

$X_n = \phi(X_{n-1}, e_n)$

```
Generate_Trajectory()
  Data: A transition function $\Phi$ and an initial state $x_0$
  Output: A coherent trajectory of the Markov chain
  $x = x_0$ repeat
      $e = generate\_event()$
      $x = \Phi(x, e)$
      process($x$)
  until Stopping condition
  return trajectory
```

Remarks:

Completely sequential process

Strong dependence on the initial state

Outline

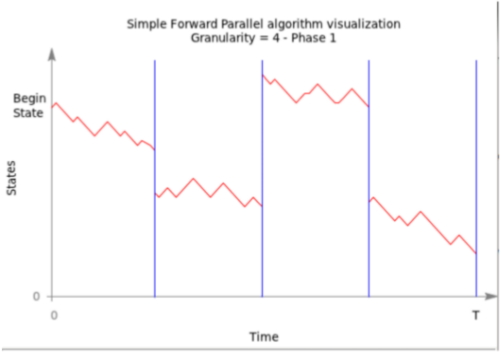
1. Generation of Long Trajectories
2. **Time parallel simulation**
3. Catch Me If You Can
4. psi3
5. Synthesis

Parallelization

- Parallelization of the transition function
  - Computational cost of Φ
  - Example : optimal routing policy, computation of indexes,...
- Parallelization of the trajectories
  - Provide "independent" samples of trajectories
  - Statistically efficient
  - Control of the simulation
- Time parallel simulations
  - Provide a single long trajectory
  - Use the all capability of the machine
  - Hard to code

Time parallel

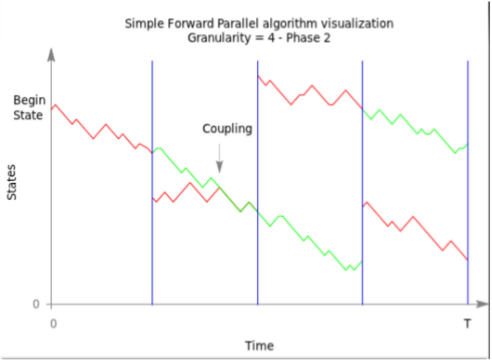
Nicol et al. algorithm



Time parallel trajectories

Time parallel

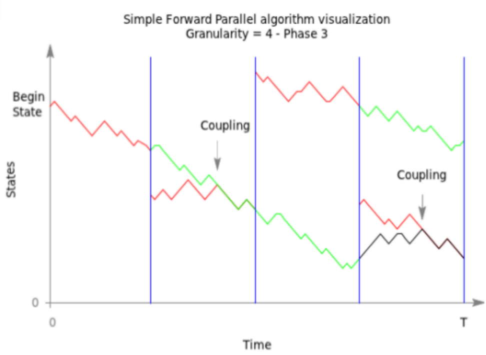
Nicol et al. algorithm



Time parallel trajectories

Time parallel

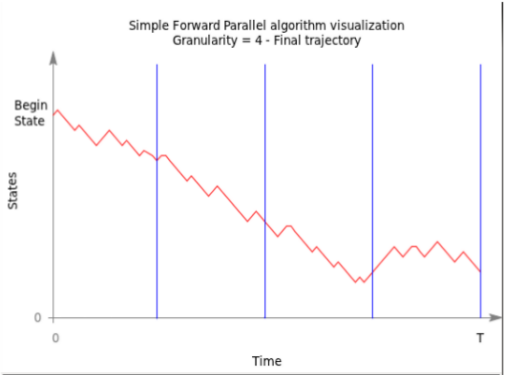
Nicol et al. algorithm



Time parallel trajectories

Time parallel

Nicol et al. algorithm



Time parallel trajectories

Time parallel

Key points:

- Task model of parallelism
- Fixed size tasks (slots)
- Master/slave scheme
- Parallel access to vectors of states

Tuning problems:

- Granularity of tasks : related to resource infrastructure
- Size of slots
  - constant
  - time-dependent
- Un-balanced load
- Synchronization problems
- Coupling tests
  - end of slots
  - during processing

Proposition:

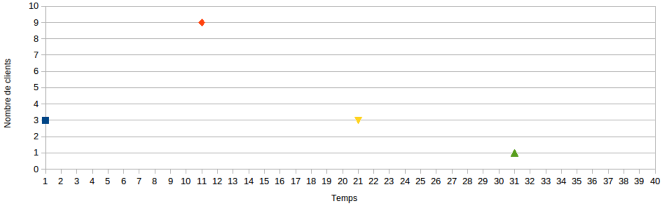
Stopping rule depending on the coupling condition

- Asynchronous scheme
- Redundant computation but local

Outline

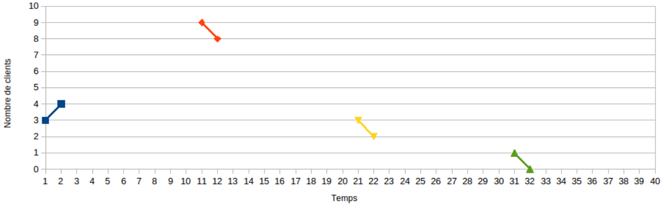
1. Generation of Long Trajectories
2. Time parallel simulation
3. **Catch Me If You Can**
4. psi3
5. Synthesis

Catch me if you can



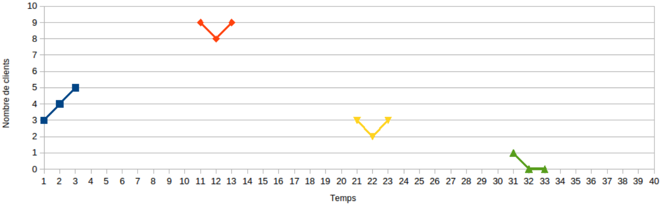
4 parallel trajectories

Catch me if you can



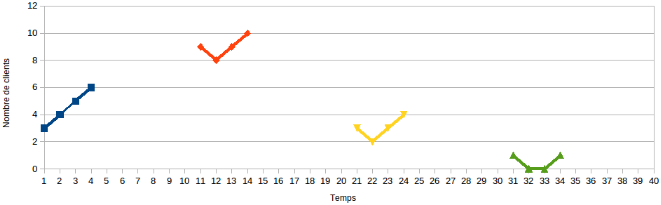
4 parallel trajectories

Catch me if you can



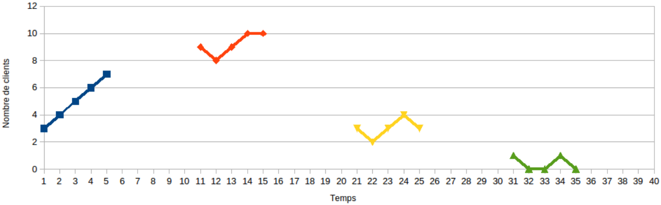
4 parallel trajectories

Catch me if you can



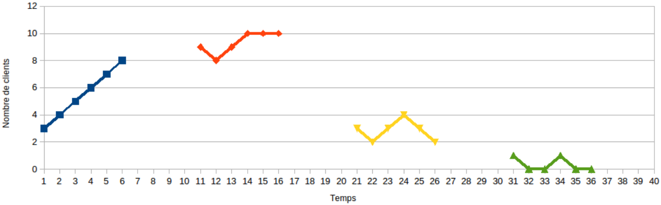
4 parallel trajectories

Catch me if you can



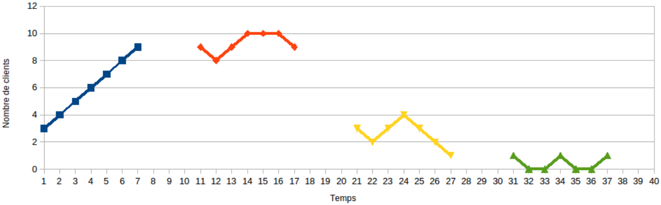
4 parallel trajectories

Catch me if you can



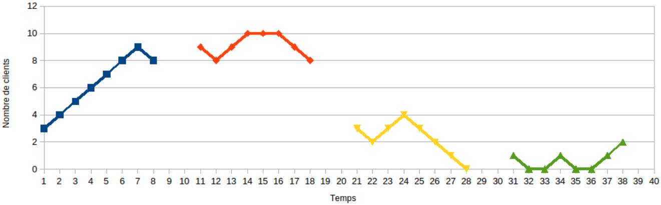
4 parallel trajectories

Catch me if you can



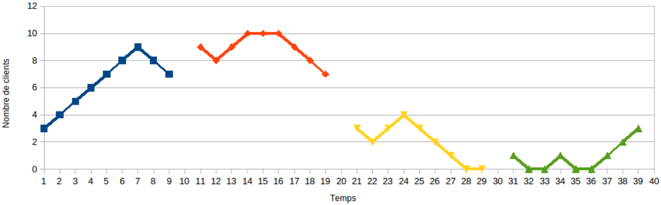
4 parallel trajectories

Catch me if you can



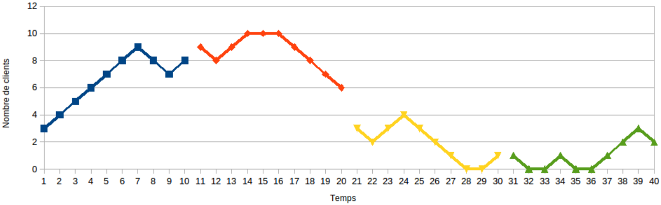
4 parallel trajectories

Catch me if you can



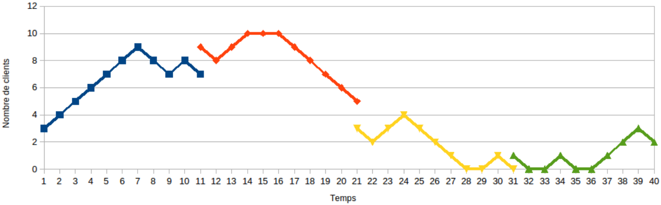
4 parallel trajectories

Catch me if you can



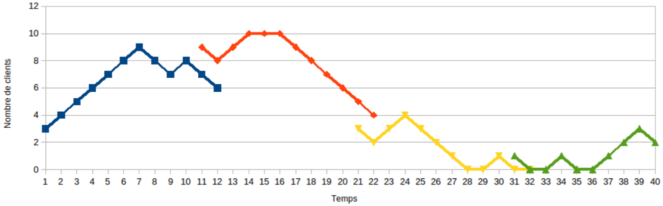
4 parallel trajectories

Catch me if you can



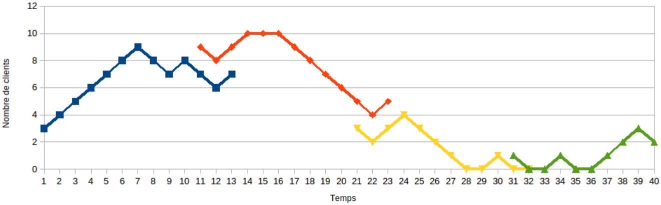
4 parallel trajectories

Catch me if you can



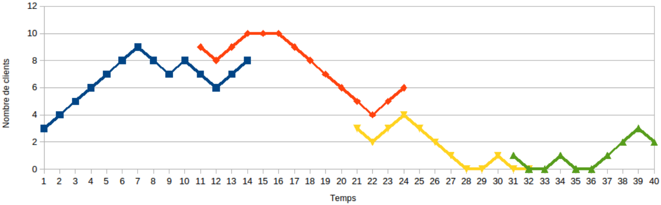
4 parallel trajectories

Catch me if you can



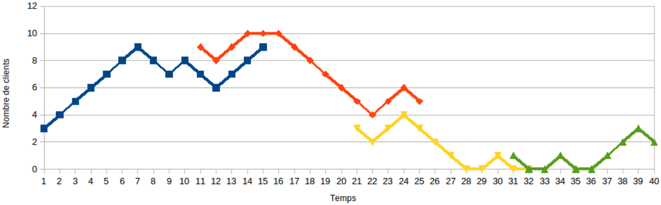
4 parallel trajectories

Catch me if you can



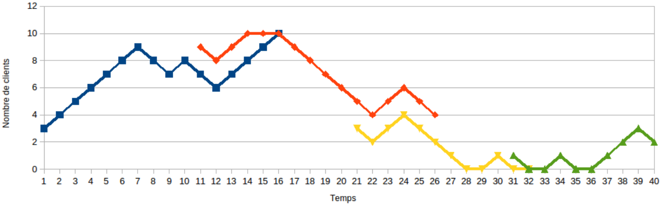
4 parallel trajectories

Catch me if you can



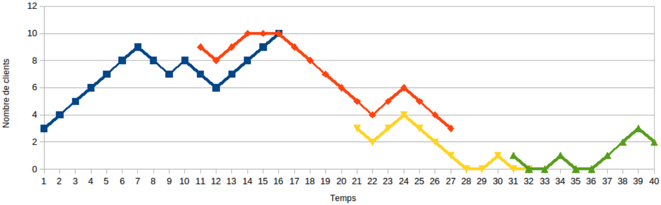
4 parallel trajectories

Catch me if you can



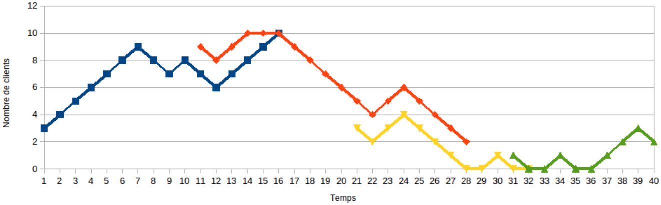
4 parallel trajectories

Catch me if you can



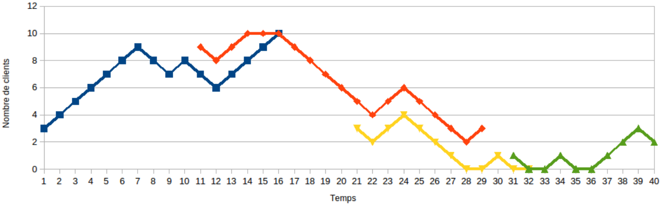
4 parallel trajectories

Catch me if you can



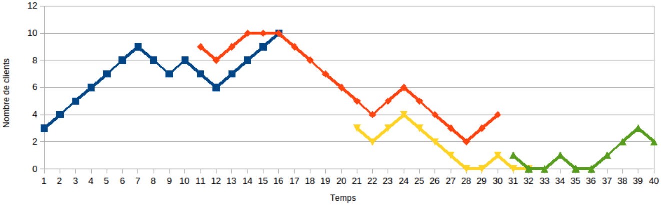
4 parallel trajectories

Catch me if you can



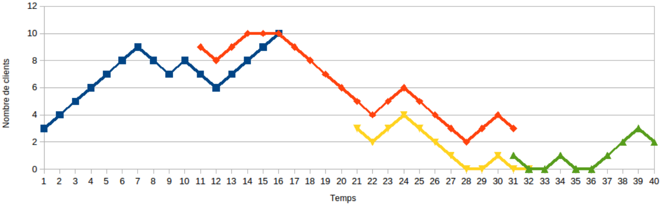
4 parallel trajectories

Catch me if you can



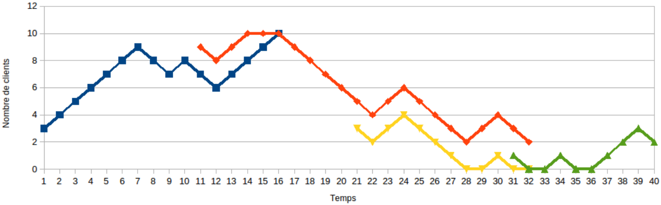
4 parallel trajectories

Catch me if you can



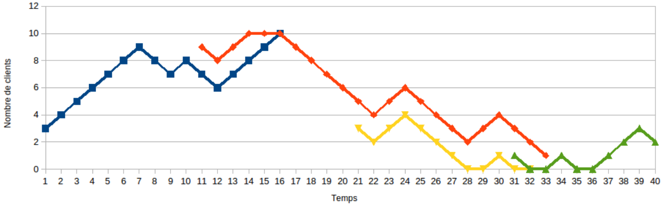
4 parallel trajectories

Catch me if you can



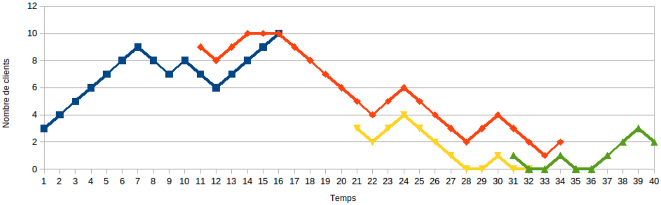
4 parallel trajectories

Catch me if you can



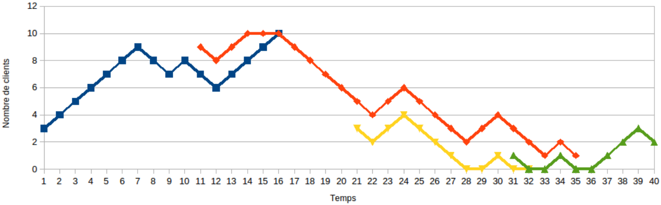
4 parallel trajectories

Catch me if you can



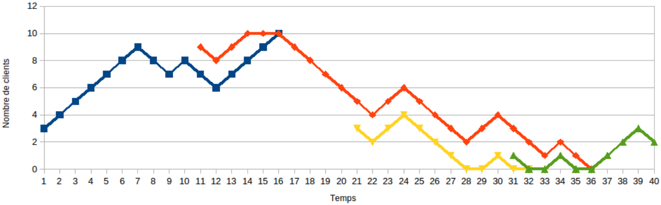
4 parallel trajectories

Catch me if you can



4 parallel trajectories

Catch me if you can



4 parallel trajectories

Parallelization framework

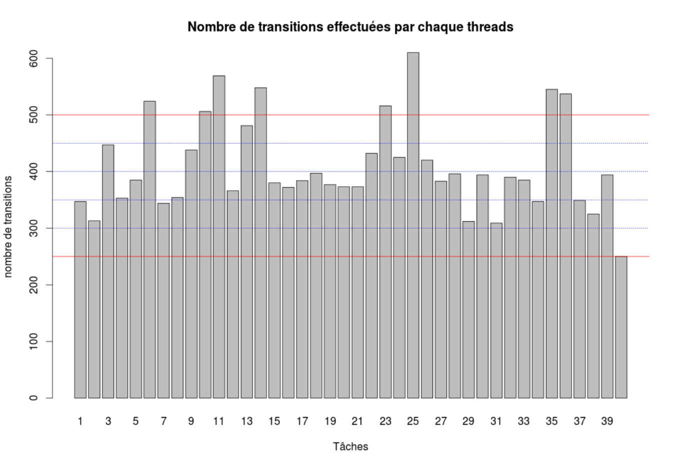
OpenMP with tasks

- Automatic thread kernel management
- Task model, (easy to implement)
- previous versions in psi3

Data structures

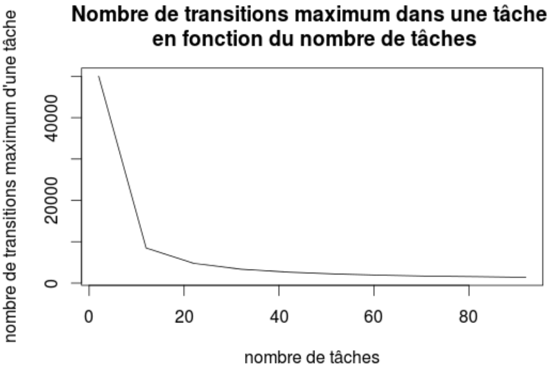
- Array of linked lists
  - State
  - Task id
- Linked lists of trajectories
  - Initial task id
  - Coupling task id
  - trajectory reference id
- List of events
- Coherent trajectory

Coupling times



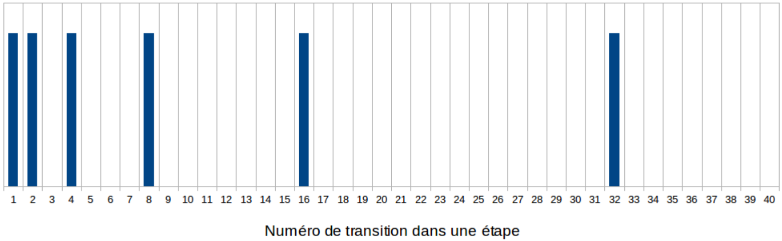
Simulation of a 40 servers Call center, trajectory length = 10000

Work



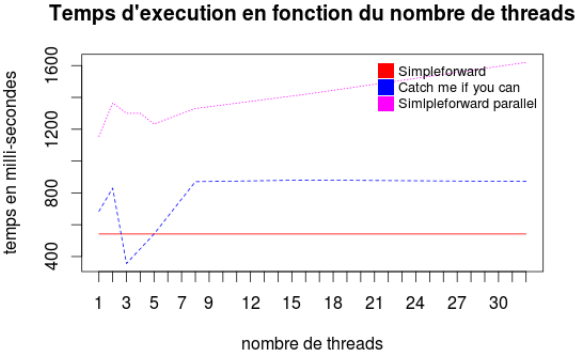
M / M / 1 *5 threads and 300 samples*

Shared memory calls



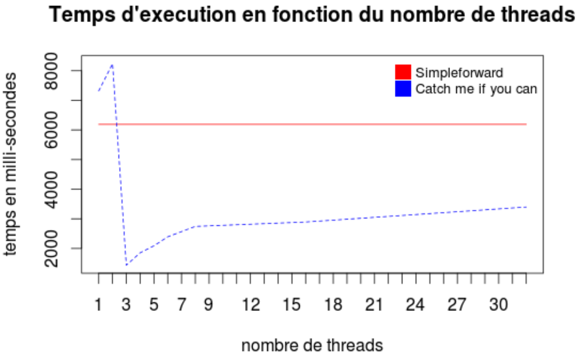
Shared memory access scheme

Simulation time of a $M/M/1$



IDFreeze (48 cores) same event list

Simulation time of a call center

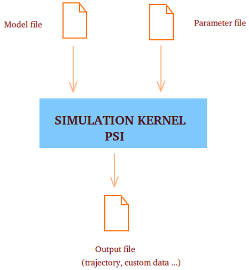


IDFreeze (48 cores) same event list

Outline

1. Generation of Long Trajectories
2. Time parallel simulation
3. Catch Me If You Can
4. **psi3**
5. Synthesis

Le logiciel Psi



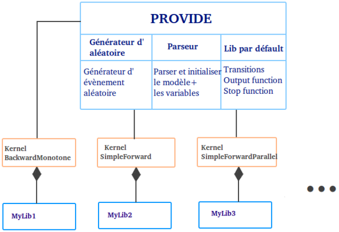
Model file

Parameter file

SIMULATION KERNEL
PSI

Output file
(trajectory, custom data ...)

Psi3 organization



PROVIDE

Générateur d' aléatoire	Parseur	Lib par défault
Générateur d' évènement aléatoire	Parser et initialiser le modèle+ les variables	Transitions Output function Stop function

Kernel BackwardMonotone

Kernel SimpleForward

Kernel SimpleForwardParallel

MyLib1

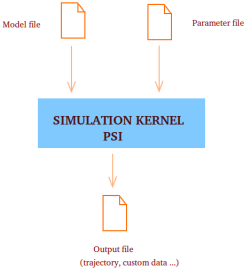
MyLib2

MyLib3

• • •

psi3 architecture

Le logiciel Psi



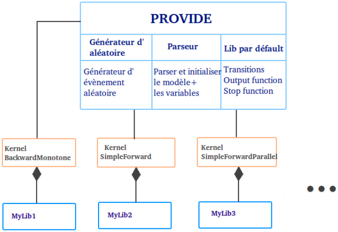
Model file

Parameter file

SIMULATION KERNEL PSI

Output file (trajectory, custom data ...)

Psi3 organization



PROVIDE

Générateur d' aléatoire	Parseur	Lib par défault
Générateur d' évènement aléatoire	Parser et initialiser le modèle+ les variables	Transitions Output function Stop function

Kernel BackwardMonotone

Kernel SimpleForward

Kernel SimpleForwardParallel

MyLib1

MyLib2

MyLib3

• • •

psi3 architecture

Outline

1. Generation of Long Trajectories

2. Time parallel simulation

3. Catch Me If You Can

4. psi3

5. **Synthesis**

Synthesis

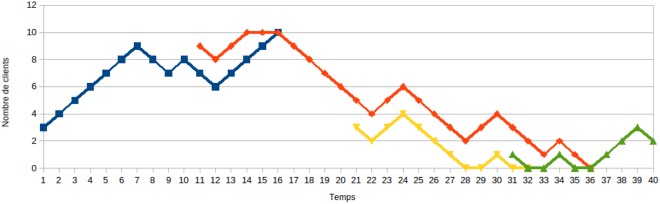
Software prototype

- Implementation of Nicol's algorithm
  - still not efficient
  - tuning procedures ???
- Implementation of Catch me if you can
  - still not efficient
  - better cores utilization
  - better execution time

Further work

- Optimization mechanisms : threads scheduling, memory decomposition,...
- Scheduling strategies
- Synchronization schemes
- I/O parallelization

Open questions



On p resources

$$T_p(n) \leq T_1(\frac{n}{p}) + \max_{1 \leq k \leq p-1} \tau_i$$

Hitting times : $\tau_{x,y}$