Parallel Sampling of Markov Chains

Benjamin.Briot@inria.fr, Jean-Marc.Vincent@imag.fr
Laboratoire d'Informatique de Grenoble
Équipe projet Inria POLARIS
ANR MARMOTE

Roadef, février 2016

Parallel Sampling of Markov Chains

Markov Chains

Markov Chains

- Widely used models for performance evaluation of systems and networks
- Large state-space N

Formal Solving

Based on structured models

- Closed formula : birth and death processes
- Product form networks

Numerical Solving $N \leqslant 10^7$

Based on Matrix representation

$$\begin{aligned} \pi_{n+1} &= \pi_n P && \text{transient distribution} \\ \pi &= \pi P && \text{fixed point : steady state} \end{aligned}$$

Numerical algorithms : power method, CGS, ...

Markov Chains

Markov Chains

- Widely used models for performance evaluation of systems and networks
- Large state-space N

Formal Solving

Based on structured models

- Closed formula : birth and death processes
- Product form networks

Numerical Solving $N \leqslant 10^7$

Based on Matrix representation

$$\begin{aligned} \pi_{n+1} &= \pi_n P && \text{transient distribution} \\ \pi &= \pi P && \text{fixed point : steady state} \end{aligned}$$

Numerical algorithms : power method, CGS, ...

Simulation Methods

Based on Algorithmic model (program)
Execution ⇒ Trajectory sampling
Statistical analysis ⇒ Performance indexes

- Forward simulation
  - Iterated from an initial state
  - Burn-in time period
- Steady state sampling (perfect)
  - Avoid burn-in time
  - Gives directly a steady-state of the system

Simulation Cost

- Forward simulation
  - Linear in the length of the trajectories
  - Linear in the size of the sampling (large samples needed)
- Steady state sampling (perfect)
  - Global coupling cost (usually linear in the dimension of the model)
  - Linear in the size of the sampling

A TYPICAL EXAMPLE

Evaluate the packet loss of a network

A typical example

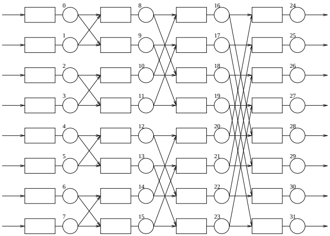
Evaluate the packet loss of a network

- Modeling of the network and the workload
  - State-space size : 100^{32}

Model

Delta network

- 32 queues, 8 input/output
- Queue capacity : 100
- Input rate : 1.8
- Service rate : 2.0



A typical example

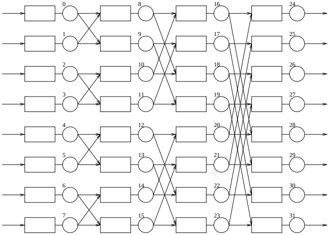
Evaluate the packet loss of a network

- Modeling of the network and the workload
  - State-space size : 100^{32}
- Sampling a huge number of steady states
  - 10^6 samples : to get a good confident interval

Model

Delta network

- 32 queues, 8 input/output
- Queue capacity : 100
- Input rate : 1.8
- Service rate : 2.0



A typical example

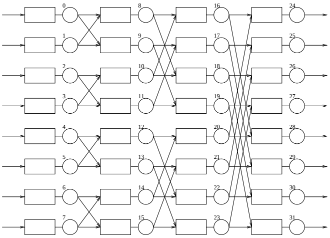
Evaluate the packet loss of a network

- Modeling of the network and the workload
  - State-space size : 100^{32}
- Sampling a huge number of steady states
  - 10^6 samples : to get a good confident interval
- Applying a statistical analysis on this samples
  - Proportion of samples showing at least one full queue

Model

Delta network

- 32 queues, 8 input/output
- Queue capacity : 100
- Input rate : 1.8
- Service rate : 2.0



A typical example

Evaluate the packet loss of a network

- Modeling of the network and the workload
  - State-space size : 100^{32}
- Sampling a huge number of steady states
  - 10^6 samples : to get a good confident interval
- Applying a statistical analysis on this samples
  - Proportion of samples showing at least one full queue

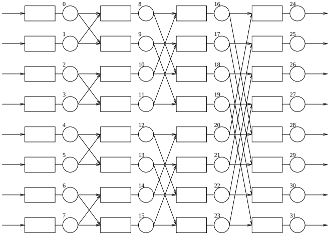
Results

Simulation time : X
Loss rate : Y% $\pm$ Z

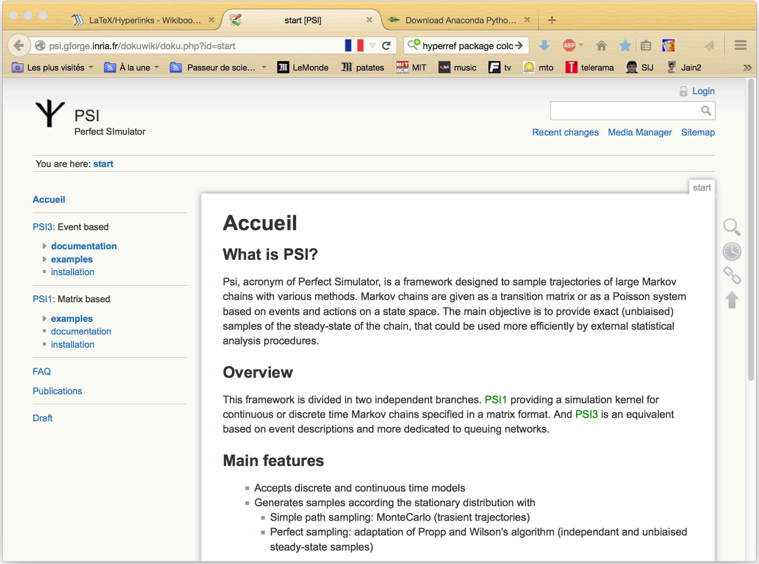
Model

Delta network

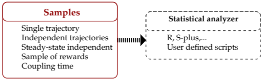
- 32 queues, 8 input/output
- Queue capacity : 100
- Input rate : 1.8
- Service rate : 2.0



PSI.GFORGE.INRIA.FR



Simulation Workflow



Simulation Workflow

Model libraries

Queues description
servercs, capacities
Event description
Rate
Activation condition
Action of the event

Simulation kernels

Forward sampling
trajectories
Backward sampling
Monotone
Envelopes
Envelopes and split

Samples

Single trajectory
Independent trajectories
Steady-state independent
Sample of rewards
Coupling time

Statistical analyzer

R, S-plus,...
User defined scripts

SIMULATION WORKFLOW

Events library

Predifined routing
overflow, blocking
JSQ, JSWT

Index routing
Table of index functions
$Dest = argmin I_i(x_i)$

Model libraries

Queues description
servercs, capacities

Event description
Rate
Activation condition
Action of the event

Simulation control

Max backward simulation run
Sample size
Stopping criteria
(reward function)
Number of variates

Simulation kernels

Forward sampling
trajectories

Backward sampling
Monotone
Envelopes
Envelopes and split

Samples

Single trajectory
Independent trajectories
Steady-state independent
Sample of rewards
Coupling time

Statistical analyzer

R, S-plus,...
User defined scripts

SIMULATION WORKFLOW

Events library

Predifined routing
- overflow, blocking
- JSQ, JSWT

Index routing
- Table of index functions
- $Dest = argmin I_i(x_i)$

Model libraries

Queues description
- servercs, capacities

Event description
- Rate
- Activation condition
- Action of the event

Model

Description
parameters

Simulation control

Max backward simulation run
Sample size
Stopping criteria
(reward function)
Number of variates

Simulation kernels

Forward sampling
- trajectories

Backward sampling
- Monotone
- Envelopes
- Envelopes and split

Method

Description
parameters

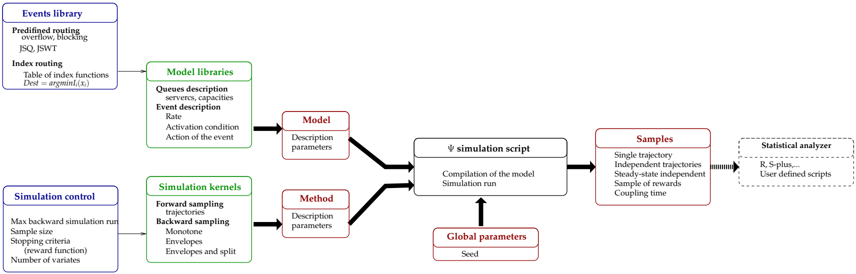
Samples

Single trajectory
Independent trajectories
Steady-state independent
Sample of rewards
Coupling time

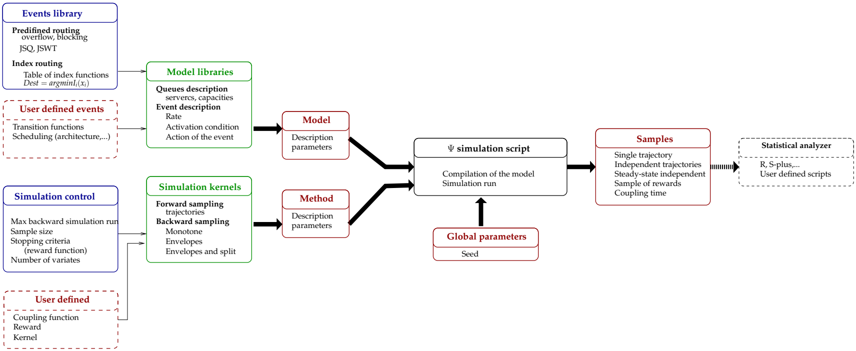
Statistical analyzer

R, S-plus,...
User defined scripts

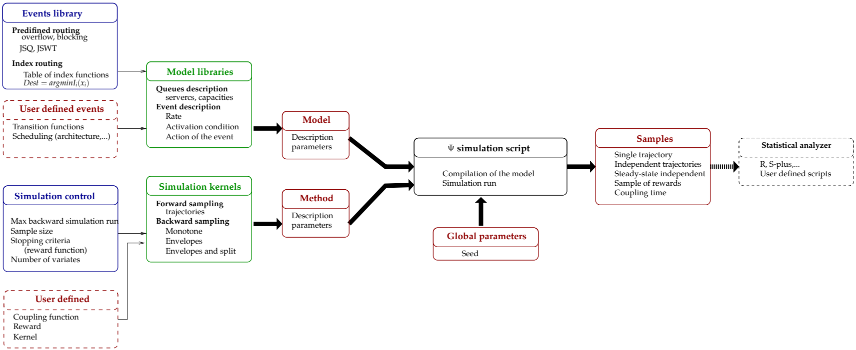
Simulation Workflow



Simulation Workflow



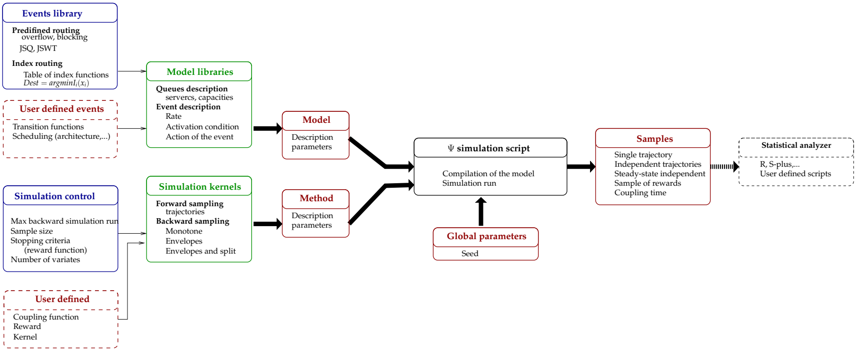
Simulation Workflow



Aim of the software

- Finite capacity queueing network simulator
- Rare events estimation (rejection, blocking, ...)
- Statistical guarantees (independence of samples)

SIMULATION WORKFLOW



Aim of the software

- Finite capacity queueing network simulator
- Rare events estimation (rejection, blocking, ...)
- Statistical guarantees (independence of samples)

⇒ Simulation Kernels

- Open source (C, GPL licence)
- Extensible library of events
- Multiplatforms (Linux, Mac OSX, ...)
- Sequential codes

Parallel Simulation

Computing ressources

- Many cores machine (desktop (4), server (16) or parallel (48))
- Parallel programing frameworks (OpenMP)

Objective

- Evaluate efficiency of parallel implementations on multicore platforms

Forward Simulation

- Space parallel approach
  - Generate a separate Markov chain on each core and combine results **[Glynn92]**
- Time parallel approach
  - Divide the iteration space and try to precompute parts of the Markov chain **[Nicol94]**
- Space-time parallel approach
  - Divide the model in several Markov chains **[HsiehG09]**

Backward Simulation

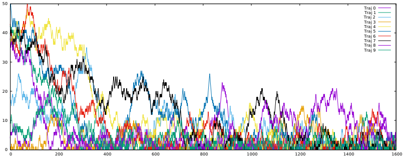
- Generate samples on each core (natural approach)

Multiple Forward : transient analysis

Forward Simulation

- Return a trajectory (states sequence) of a markov chain.
- Sequential process (Monte Carlo simulation).

Combining multiple Markov chains



Space parallel approach

- Task view : all trajectories are independant tasks.
- Kernel duplication : each core runs its own simulation kernel to compute tasks.

Difficulties

- Ensure an independance between the different random number streams on the separate tasks :
  - Use the random generator of L'Ecuyer **[LEcuyer98]**.
- Manage tasks along the simulation
  - OpenMP runtime

Multiple Forward : Performance

Model

- 32 input/output delta-switching network (total 192 queues)
- Queue capacity is 100
- Packets are rejected if the queue is full

Machine (from Parasilo cluster on GRID5000@Rennes)

- CPU : 16 cores, Intel Xeon E5-2630, 2.4GHz
- Storage : SSD disk

Multiple Forward : Performance

Model

- 32 input/output delta-switching network (total 192 queues)
- Queue capacity is 100
- Packets are rejected if the queue is full

Machine (from Parasilo cluster on GRID5000@Rennes)

- CPU : 16 cores, Intel Xeon E5-2630, 2.4GHz
- Storage : SSD disk

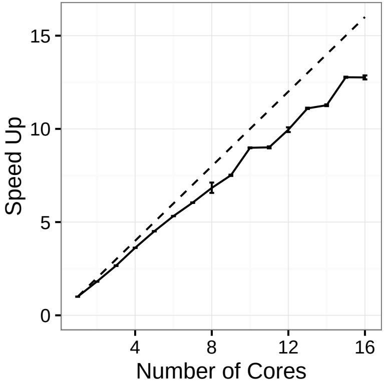
Timing

Sequential :	155 seconds
16 cores :	12 seconds

Observations

- Speed-up is close to be linear

Results



Multiple Perfect Sampler

Percect Sampling

- Return a steady-state of the system.
- Sequential process (Prop & Willson algorithm).

Space parallel approach

- Task view : all samples are independant tasks
- Kernel duplication : each core runs its own simulation kernel to compute tasks

Difficulties

- Ensure an independance between the different random number streams on the separate tasks :
  - Use the random generator of L'Ecuyer **[LEcuyer98]**.
- Manage tasks along the simulation
  - OpenMP runtime

Multiple Perfect Sampler : Performance

Model

- 32 input/output delta-switching network (total 192 queues)
- Queue capacity is 100
- Packets are rejected if the queue is full

Machine (from Parasilo cluster on GRID5000@Rennes)

- CPU : 16 cores, Intel Xeon E5-2630, 2.4GHz
- Storage : SSD disk

Multiple Perfect Sampler : Performance

Model

- 32 input/output delta-switching network (total 192 queues)
- Queue capacity is 100
- Packets are rejected if the queue is full

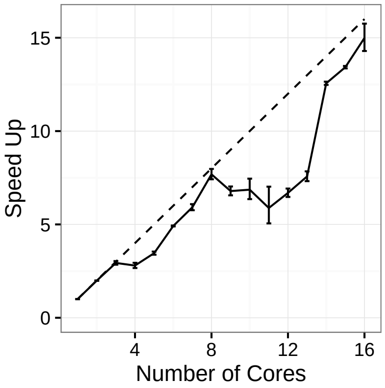
Machine (from Parasilo cluster on GRID5000@Rennes)

- CPU : 16 cores, Intel Xeon E5-2630, 2.4GHz
- Storage : SSD disk

Timing

Sequential :	550 seconds
16 cores :	37 seconds

Results



Observations

- Kernel replication allows good performances on backward simulation
- Some irregularities : we guess some load sharing problems

Overlapping computation and data movements

Generating ONE long trajectory.

- Hard to parallelize due to its sequential intrinsic properties.
- I/O bounded : nearly 98% of the simulation time is spent in extracting data.

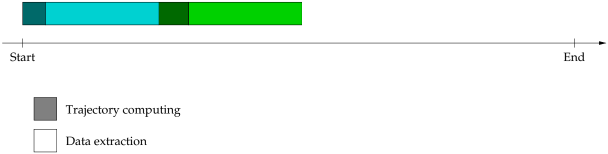
Generating a trajectory : timeline

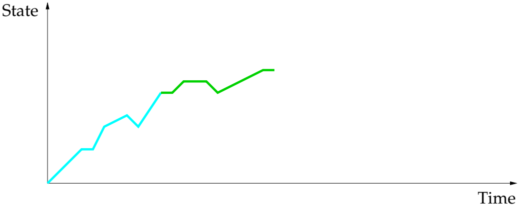
Overlapping computation and data movements

Generating ONE long trajectory.

- Hard to parallelize due to its sequential intrinsic properties.
- I/O bounded : nearly 98% of the simulation time is spent in extracting data.

Generating a trajectory : timeline



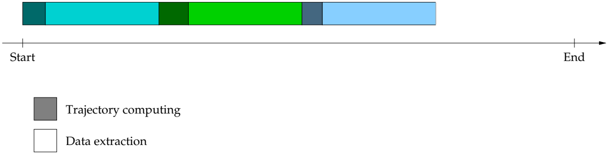


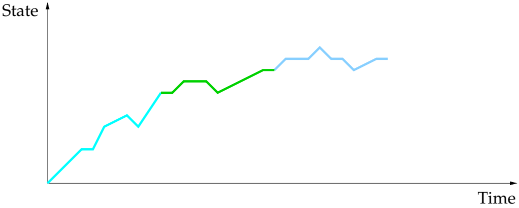
OVERLAPPING COMPUTATION AND DATA MOVEMENTS

Generating ONE long trajectory.

- Hard to parallelize due to its sequential intrinsic properties.
- I/O bounded : nearly 98% of the simulation time is spent in extracting data.

Generating a trajectory : timeline



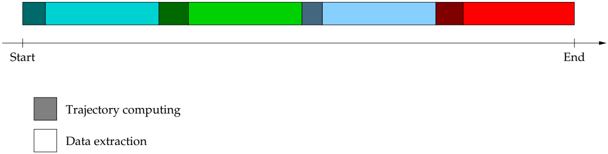


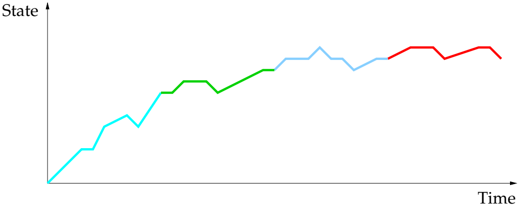
OVERLAPPING COMPUTATION AND DATA MOVEMENTS

Generating ONE long trajectory.

- Hard to parallelize due to its sequential intrinsic properties.
- I/O bounded : nearly 98% of the simulation time is spent in extracting data.

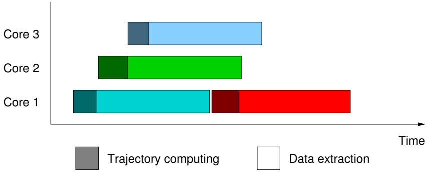
Generating a trajectory : timeline





Overlapping computation and data movements : Performance

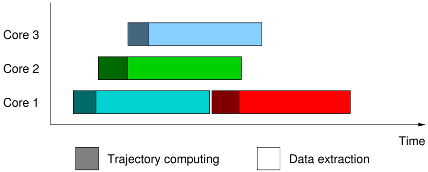
I/O pipelining : Gantt chart



- Trajectory is still computed sequentially.
- I/O are made in parallel.

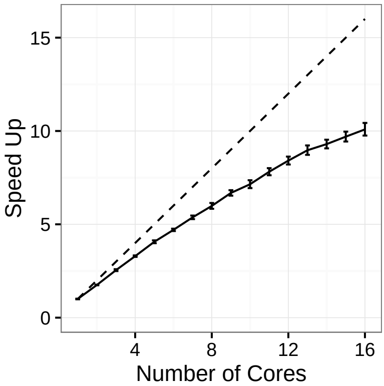
Overlapping computation and data movements : Performance

I/O pipelining : Gantt chart



- Trajectory is still computed sequentially.
- I/O are made in parallel.

Results



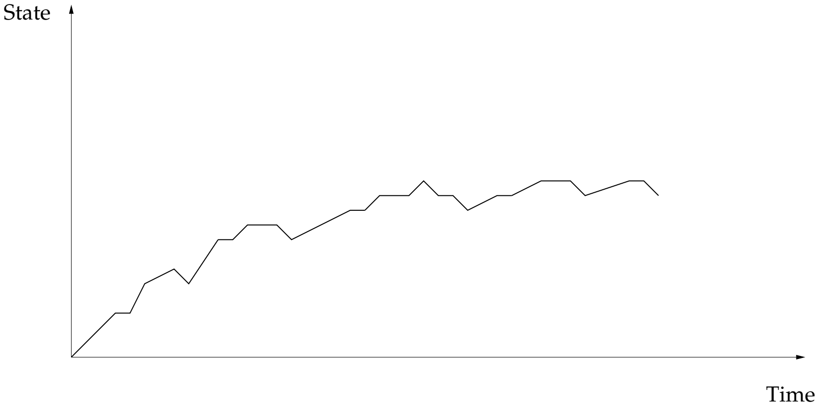
Timing

Sequential : 1560 milliseconds
16 cores : 154 milliseconds

Observations

- Performances are not linear and seems to reach a limit.

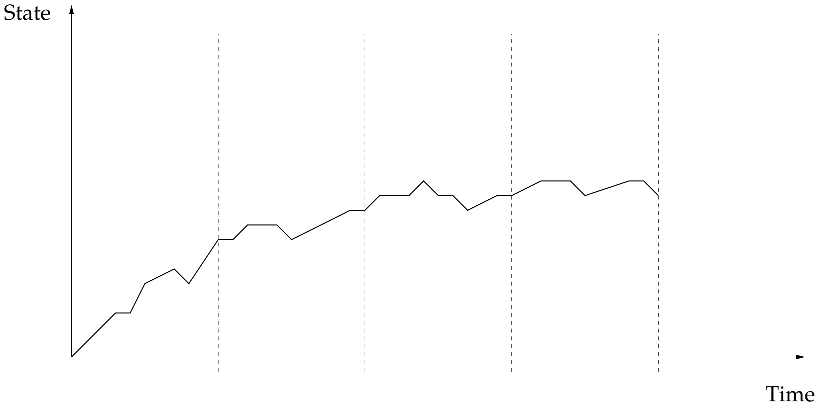
Nicol's Algorithm



Time parallel approach

- Events sequence is generated.
- Dividing the trajectory in several intervals.
- Each thread will compute one interval.
  - Initialize all intervals to a particular value.
- At each iteration : ckeck if some chuncks have coupled.

Nicol's Algorithm

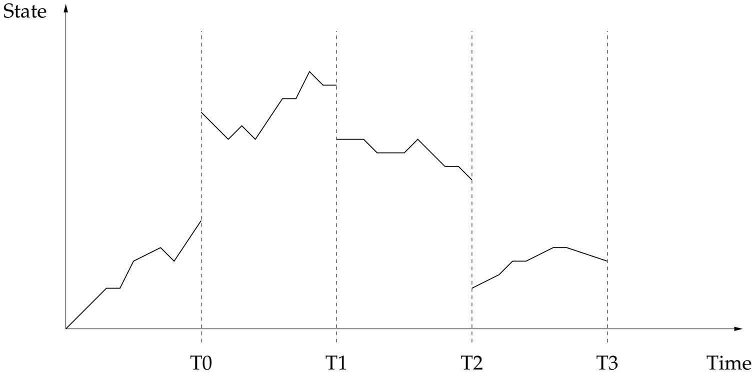


Time parallel approach

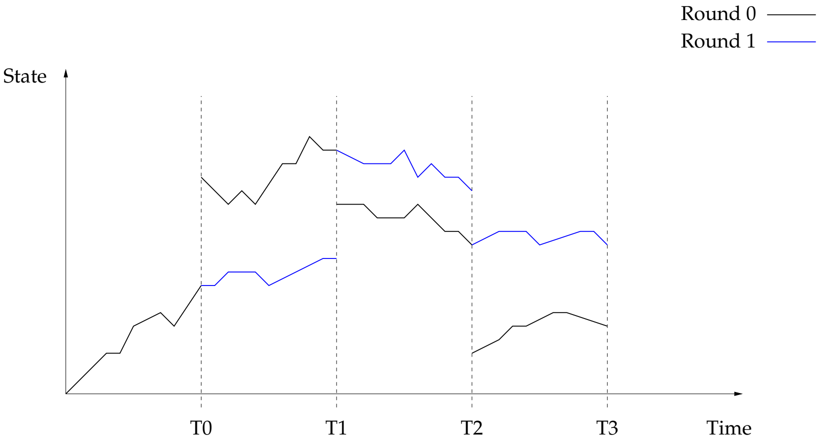
- Events sequence is generated.
- Dividing the trajectory in several intervals.
- Each thread will compute one interval.
  - Initialize all intervals to a particular value.
- At each iteration : ckeck if some chuncks have coupled.

Nicol's Algorithm : Example

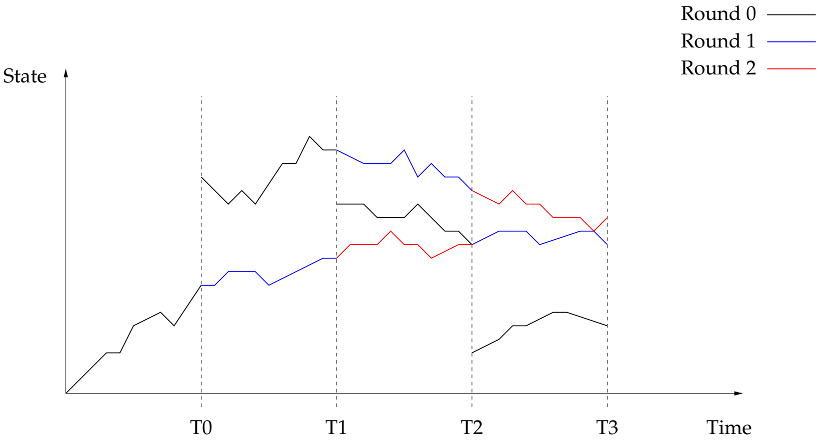
Round 0



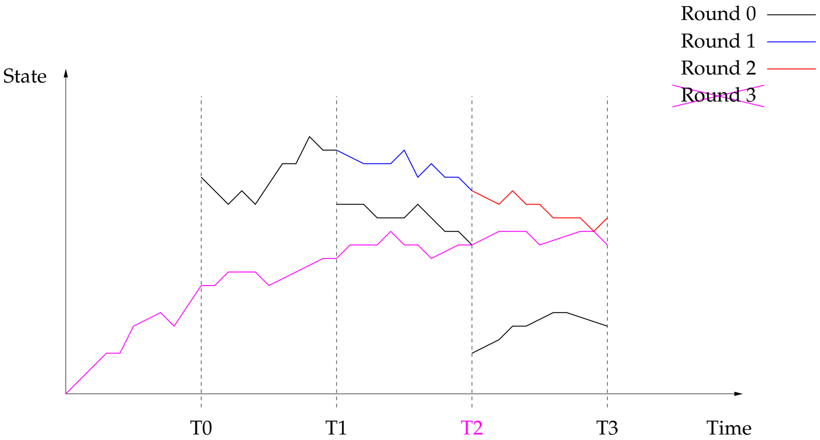
Nicol's Algorithm : Example



NICOL'S ALGORITHM : EXAMPLE



Nicol's Algorithm : Example



SYNTHESIS

Parallelization of Markov chains

- Is possible through several approaches.
- Gives good results.

PSI 3

- Parallel Backward
- Parallel Forward
- Forward with I/O pipelining

Future work

- Forward methods should come with *in-situ* analysis.
- Evaluate good parameters for Nicol's algorithm.

Reproducibility

- Everything is on : psi.gforge.inria.fr