



**HAL**  
open science

# Stochastic control optimization & simulation applied to energy management: From 1-D to N-D problem distributions, on clusters, supercomputers and Grids

Stéphane Vialle, Xavier Warin, Constantinos Makassikis, Patrick Mercier

## ► To cite this version:

Stéphane Vialle, Xavier Warin, Constantinos Makassikis, Patrick Mercier. Stochastic control optimization & simulation applied to energy management: From 1-D to N-D problem distributions, on clusters, supercomputers and Grids. Grid@Mons conference, May 2008, Mons, Belgium. hal-00291821

**HAL Id: hal-00291821**

**<https://centralesupelec.hal.science/hal-00291821>**

Submitted on 25 Feb 2009

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

**Stochastic control optimization & simulation  
applied to energy management:  
From 1-D to N-D problem distributions, on  
clusters, supercomputers and Grids.**

Stéphane Vialle *Stephane.Vialle@supelec.fr*

Xavier Warin *Xavier.Warin@edf.fr*

Constantinos Makassikis *Constantinos.Makassikis@supelec.fr*

Patrick Mercier *Patrick.mercier@supelec.fr*

Grid@Mons – May 9, 2008 – Mons

# Project 1: the first steps

**“One-dimensional stochastic optimization:  
application to gas storage valuation”**

# 1 – First Motivations and Objectives

For technical, financial and legal reasons, energy trading companies need to have energy stocks.

→ For example: gas storage.

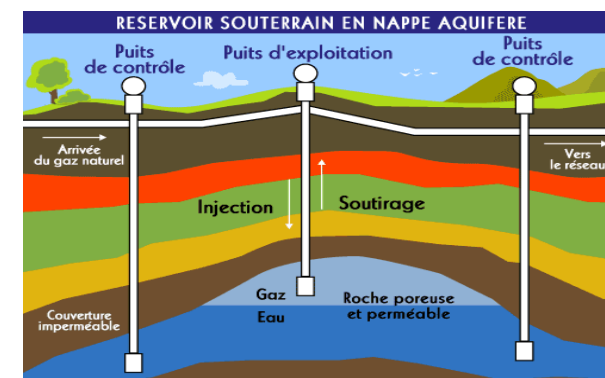
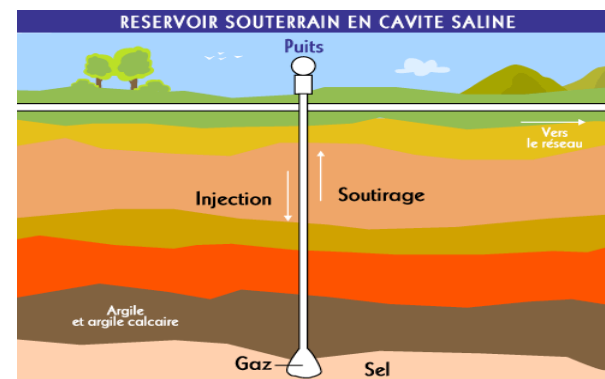
These are expensive investments!

→ How to optimize the management of gas storages?

→ How to compute the renting price ?

## Computation of a « gas storage valuation »:

- compute the « value » of an investment project about a gas storage cavity,
- compute profits and risks associated with the ownership of a gas storage cavity, considering an open energy market.



# 2.1 – Resolution method

- Instantaneous profit:  $\begin{cases} < 0 \text{ when injecting gas (we pay this gas)} \\ > 0 \text{ when withdrawing gas (we sell this gas)} \end{cases}$

$$\phi(t, S_t, I_t), \text{ with } \begin{cases} S_t : \text{gas price at time } t \\ I_t : \text{regime (injection, withdrawal, nothing)} \end{cases}$$

- Expected profit from  $t_0$  maturity (T):

$$J(t_0, s, c, i, u) = \mathbb{E} \int_{t_0}^T \phi_{u_r}(r, S_r, I_r) . dr$$

includes random variables

$u$  : strategy of gas storage management

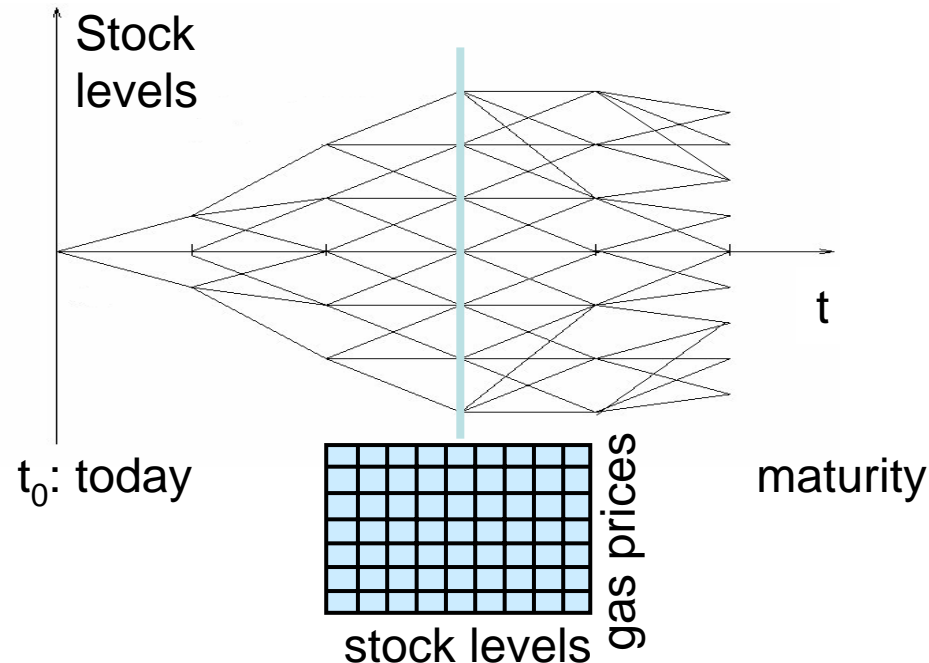
- Looking for a strategy  $u$  leading to the maximal profit  $J^*$ :

$$J^*(t_0, s, c, i) = \sup_{u \in U_{t_0}} J(t_0, s, c, i, u)$$

# 2.2 – Dynamic Programming Algorithm

## Computation of the maximal profit and associated strategy:

- using a « backward dynamic programming algorithm »
- fixing a strategic scenario at maturity
- considering 2 main variables:
  - ✓ admissible stock levels
  - ✓ possible gas prices: “alea”
- computing a 2D table of possible maximal profits at  $t_n$  function of:
  - ✓ possible profit 2D table at  $t_{n+1}$
  - ✓ action from  $t_n$  to  $t_{n+1}$
- at  $t_0$  (today) the algorithm joins the current stock level and gas price.



- Results:**
- 1) Expected (maximal) profit from  $t_0$  to maturity,
  - 2) List of actions to do at each time step in any case

# 2.2 – Dynamic Programming Algorithm

Backward dynamic programming algorithm:

for  $t \leftarrow (M - 1)\Delta t$  to 0

for  $c \in \{\text{admissible stock levels}\}$

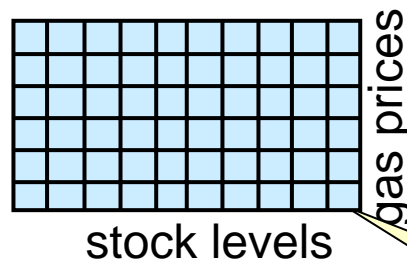
for  $s \in \{\text{possible price levels}\}$

$$\tilde{J}^*(s, c) \leftarrow \max \left( \begin{array}{l} -(a_{in}s + K_{in})\Delta t + \mathbb{E}(J^*(S_{t+\Delta t}, c + a_{in}\Delta t) | S_t = s), \\ +(a_{out}s - K_{out})\Delta t + \mathbb{E}(J^*(S_{t+\Delta t}, c - a_{out}\Delta t) | S_t = s), \\ -K_s\Delta t + \mathbb{E}(J^*(S_{t+\Delta t}, c) | S_t = s) \end{array} \right);$$

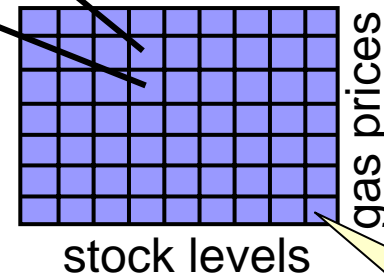
Looking for the maximal profit among 3 cases:

- ← injection
- ← withdrawal
- ← doing nothing

$J^* \leftarrow \tilde{J}^*;$



$J^*$  at  $t_n$



$J^*$  at  $t_{n+1}$

## 2.3 – Price models (“alea”)

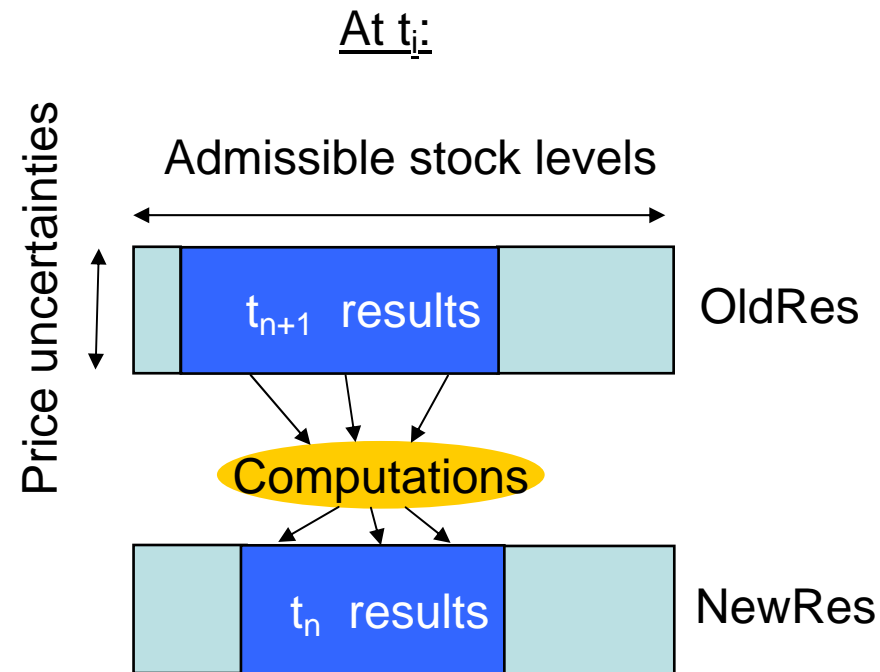
### 3 models of gas prices have been considered:

- « **1-factor Gaussian** »:
    - considering one short-term random variable to model spot prices
    - evolution of prices without any jump
  - « **2-factor Gaussian** »:
    - considering one short-term and one long-term random variables
    - evolution of prices without any jump
- Computation of the expected profit ( $\mathbb{E}$ ) using trinomial trees
- « **Normal Inverse Gaussian** »:
    - “jumping” model
    - to model the impact of phenomena leading to brutal change of gas price
- Computation of the expected profit ( $\mathbb{E}$ ) by solving a complex PDE (computed with an iterative algorithm)



# 3.1 – Parallelization strategy

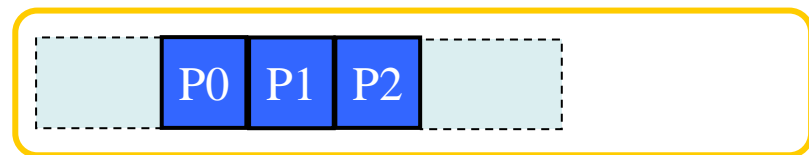
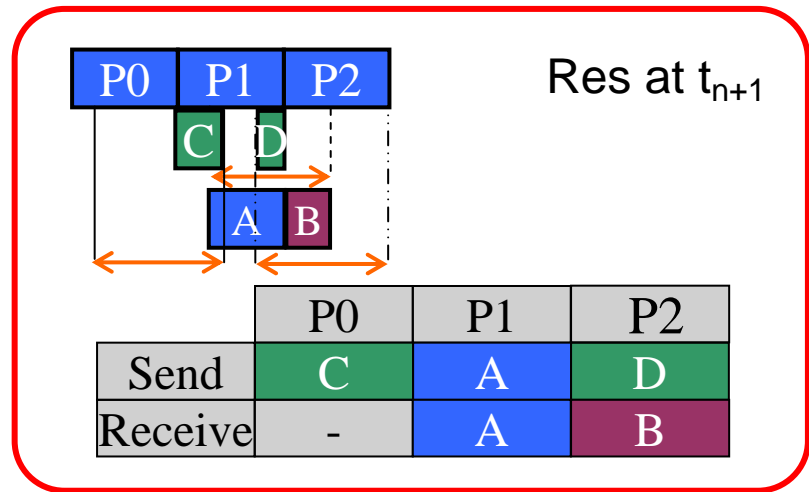
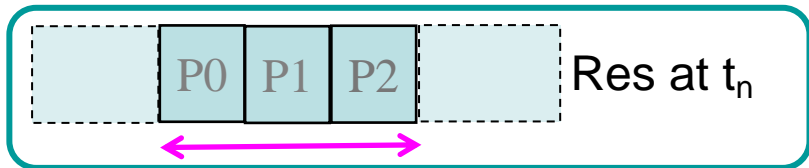
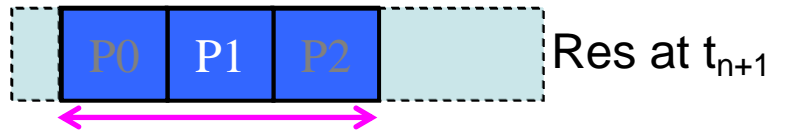
- **At each time step** two tables are used:
  - **NewRes**: for storing results of ongoing time step  $t_n$ .
  - **OldRes**: results of time step  $t_{n+1}$ .
- **Problem 1**: stock levels of interest are contiguous but with variable bounds at each time step!
- **Problem 2**: complete set of data is too large to be stored on one node!



## Strategy:

- Distribute *OldRes* and *NewRes* tables of data and computations
- Compute and route only required parts of *OldRes* table

# 3.2 – Communication scheme



Example on P1:

1) Determine the **new bounds and distribution of computations at  $t_n$**

2) Determine **data required at  $t_n$**  by P1: data to receive  
 3) Determine **data to send** by P1

} **Routing plan**

4) Allocate optimal in size **data structures**

5) Make **communications** according to routing plan with MPI library

6) **Compute Res at  $t_n$**

# 3.3 – Distributed Algorithm

Load balancing of computations at  $t_M$   
Each processor computes prices at  $t_M$

## Data exchange planning and execution

Each proc. computes the entire new load balancing map  
Each proc. computes the entire new distribution map of required input data  
Each proc. computes its routing planning, to redistribute input data for the next computation step  
Each proc. resizes its local input tables (minimizing the used memory space)

Each proc. achieves its routing planning, accordingly to a fixed routing scheme: **Send** →  
**Recv** ←

## New computation processing

Each proc. processes its range of stock values, and computes prices at  $t_n$   
function of some prices at  $t_{n+1}$

Short post processing and final result collect

# 3.4 – MPI – C++ implementation

## Implementation strategy:

To speed up: parallelization and overlapping of all communications during the execution of one routing plan,

but computations and communications can not be overlapped

To size up: avoid extra communication buffers

→ Use of `MPI_Issend` / `MPI_Irecv` / `MPI_Wait` routines

→ Memory management to **allocate strictly required memory** at each step

Others communication routines have been experimented...

... “Issend / Irecv / Wait” appeared to be the fastest.

And: uses Blitz++, Boost and Clapack libraries

# 4 - Testbeds

## 3 distributed architectures:

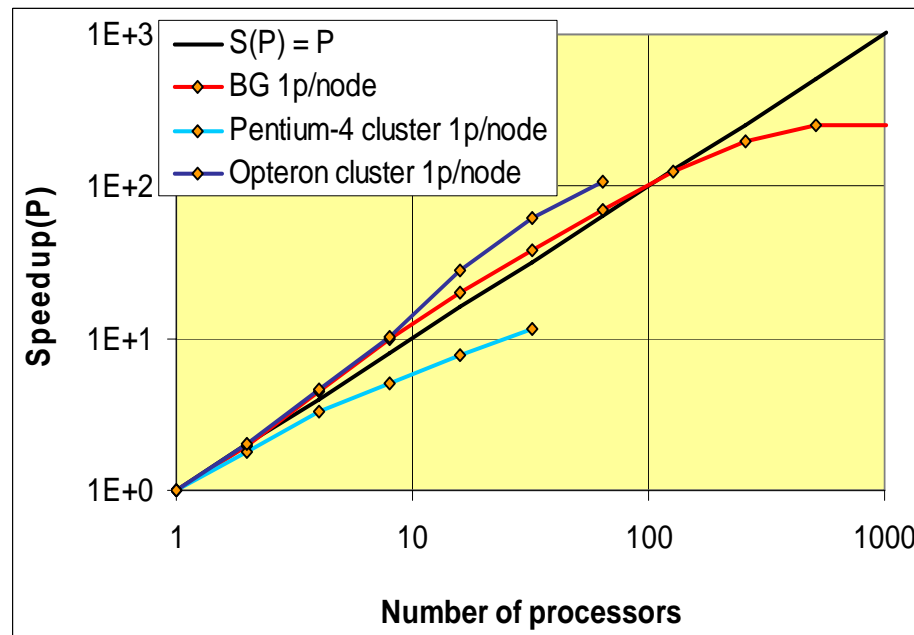
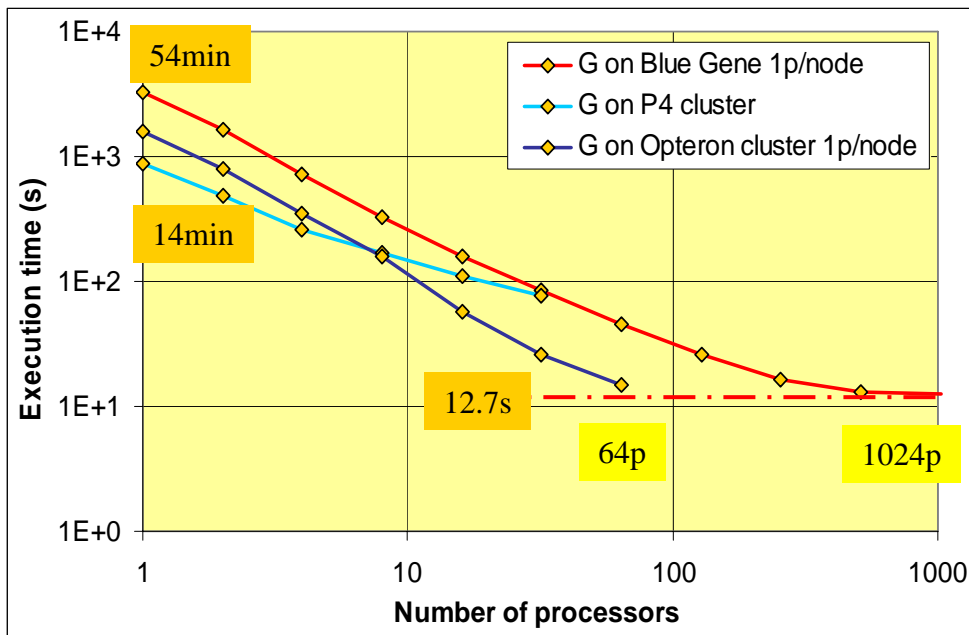
- **Small and cheap desktop-PC cluster:**
  - 32 nodes: 32x (1 mono-core Pentium-IV 3GHz, 2GB of RAM)
  - cheap Gigabit Ethernet network : 2x 24-port switches, double linked
- **Medium sized blade-PC cluster (a cluster of Grid'5000):**
  - 72 nodes: 72x (2 mono-core Opteron 2GHz, 2GB of RAM)
  - fast/normal Gigabit Ethernet network (1 large switch)
- **IBM Blue Gene/L (EDF supercomputer):**
  - 4096 nodes: 4096x (2 mono-core processors, 700MHz, 1GB of RAM)
  - Several fast interconnection networks

## 3 benchmark programs:

- with “1-factor Gaussian” price model → needs speed-up
- with “Normal Inverse Gaussian” price model → needs speed-up and size-up
- with “2-factor Gaussian” price model → needs size-up and speed-up

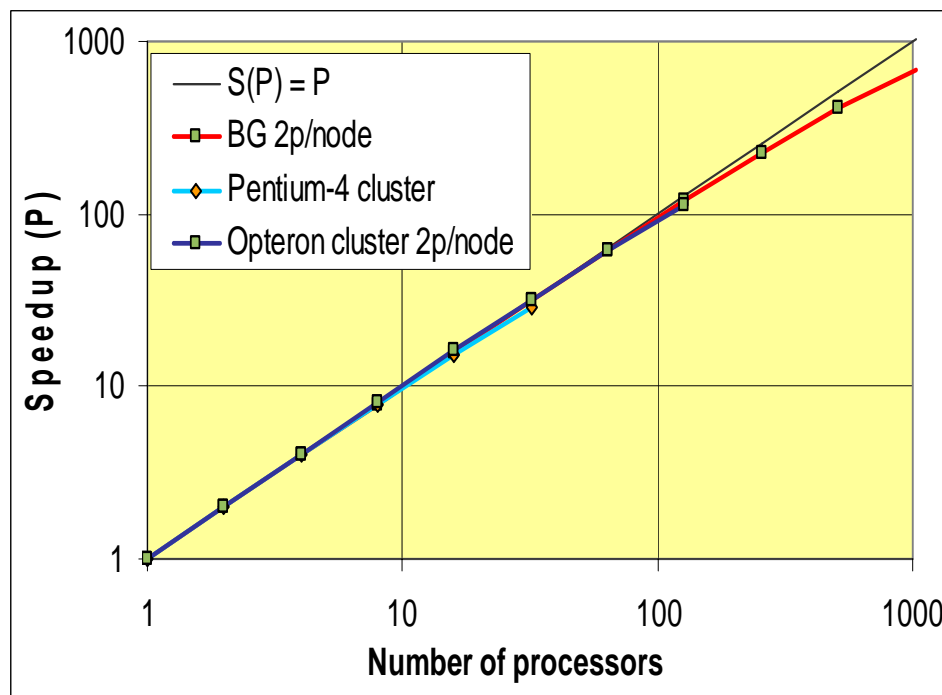
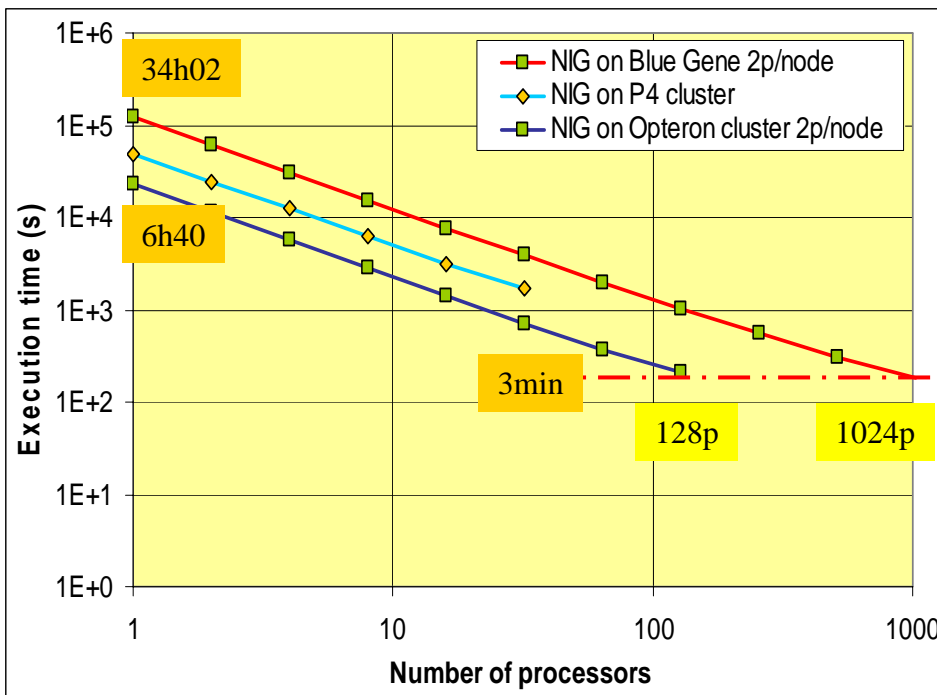
# 5.1 – 1-factor Gaussian model

- Small problem!
- Good scaling on **BG/L** and on **blade PC cluster**, and final speedup close to **254** and **107**
- Bad scaling of the small cluster (with cheap Gigabit switches)
- Some superlinear speedup due to cache memory increase (?)
- Using 2 proc. per node leads to poor performances...
- Finally, BG/L appears a little bit faster than the blade PC cluster.
- But **both seems to reach their scalability limit** on this small problem.



# 5.2 – Normal Inverse Gaussian model

- Long execution time problem
- Higher computation/communication ratio (than for 1-factor Gaussian model)
- Good scaling and final speedup (680, 29 and 110) on all parallel machines
- Using 2 proc. per node does not disturb the performances
- Finally, Blue Gene appears as fast as the blade PC cluster
- Speedups are very close of the ideal speedup

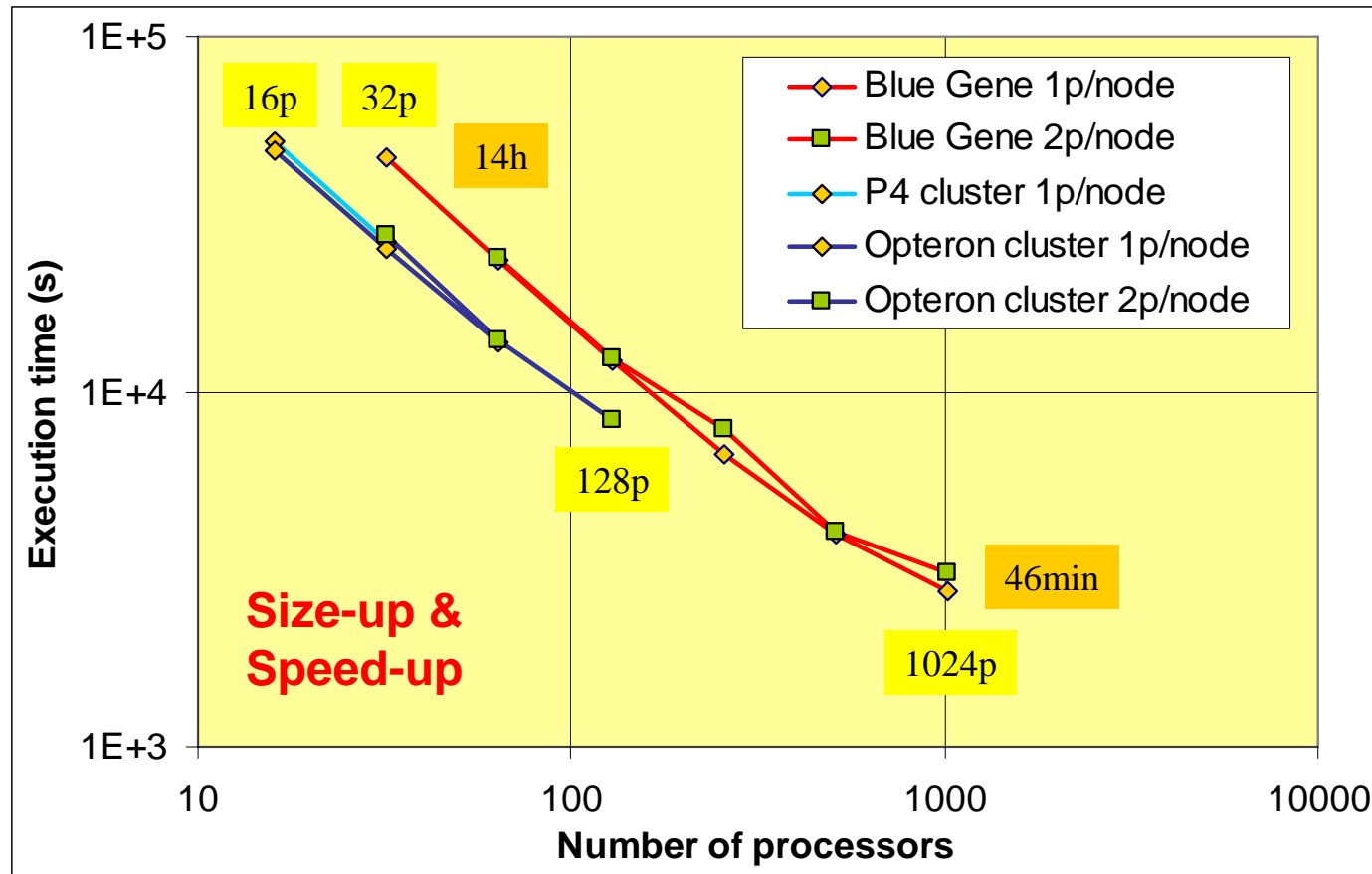


# 5.3 – 2-factor Gaussian model

- Large size and long exec. time problem: at least 11GB of memory (for sequential runs)  
→ impossible to run on 1 proc. and to compute rigorous speedups
- Highest computation/communication ratio (of the 3 benchmarks)
- Good scaling on all parallel machines

• Using 2 proc. per node does not disturb (too much) the performances.

• Blue Gene/L achieved better performances, taking advantage of its very large number of processors.





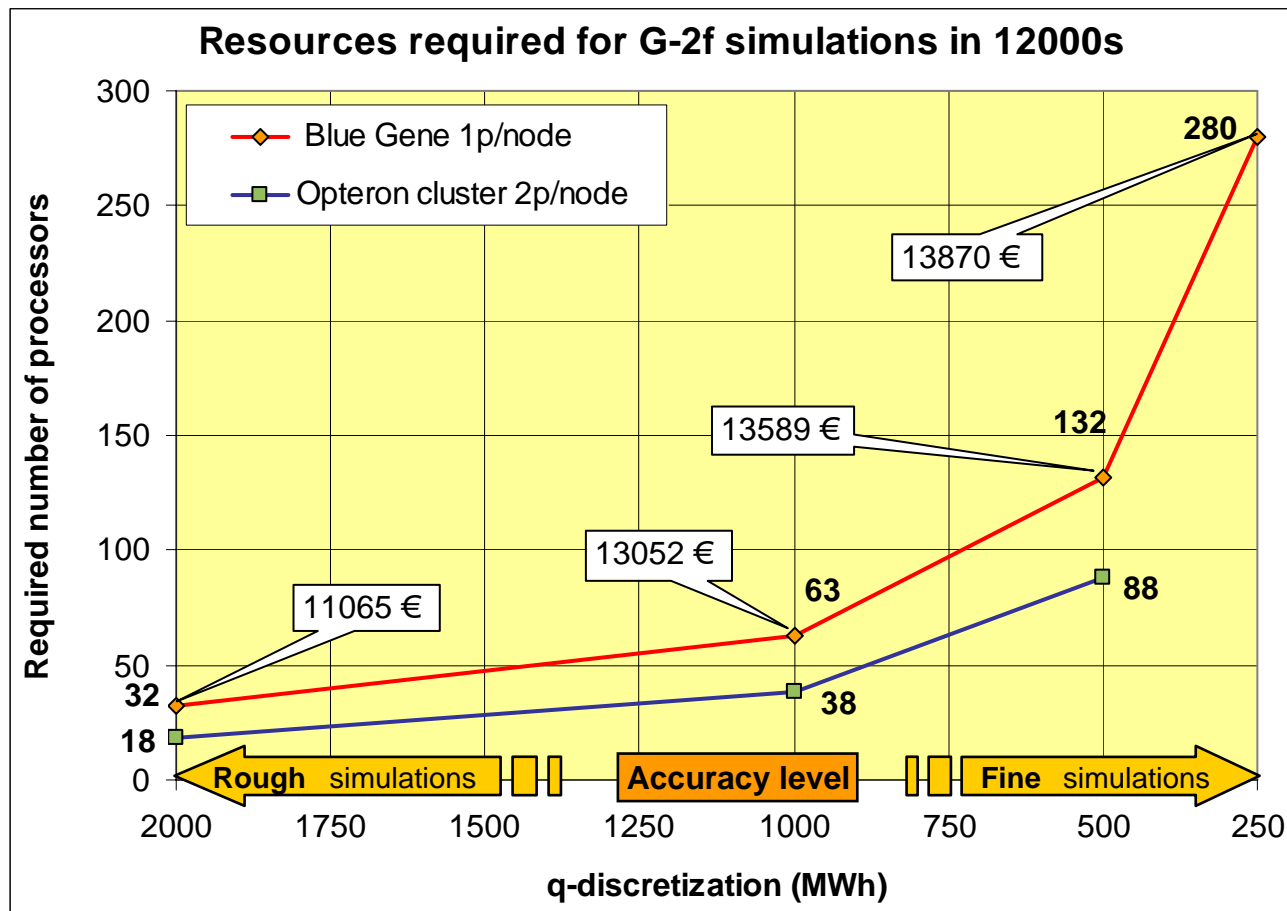
# 5.4 – Scalability experiment

- Previous simulations used a *q-discretization* factor of stock level equal to 500MWh.
- In order to run a simulation in limited time (before making a deal) and not to monopolize too many computing resources, it is necessary to adapt to the required accuracy.

→ **New scalability test** of our parallelization.

## Example:

Successful identification of the right nb. of proc. to run the parallel simulation in 12000s (3h20) function of the accuracy factor *q-discretization*.

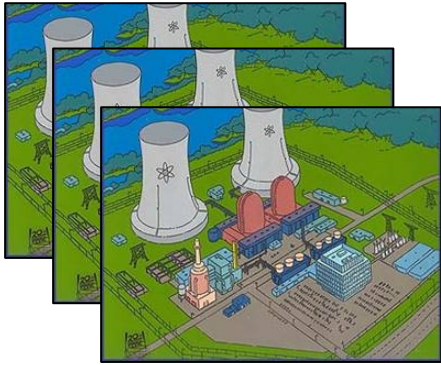


# Project 2: the real problem

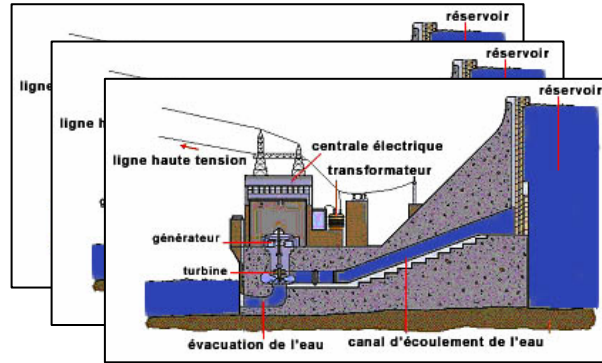
**“N-dimensional stochastic optimization and simulation: application to electricity asset management”**

# 6 – Second Motivations and Objectives

« Management of French electricity production to control cost while satisfying demand »



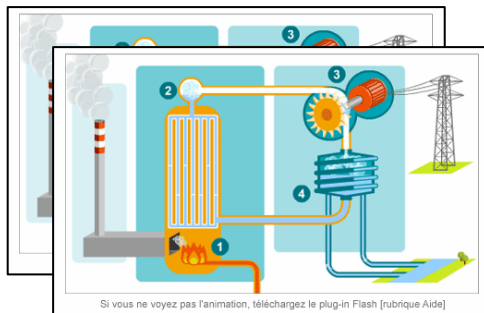
Nuclear energy



Hydraulic energy



Energy trading



Thermal energy

Etc...

**Goal:** save energetic resources and money

**Main levers:** the commands to manage the different energetic stocks

→ A N-dimensional stochastic optimization problem!

→ We extend our 1-D distributed stochastic control algorithm to N-D

# 7.1 – Dynamic Programming Algorithm

## Backward dynamic programming algorithm:

for  $t \leftarrow (M - 1)\Delta t$  to 0

for  $c \in \{\text{admissible stock levels}\}$

for  $s \in \{\text{possible price levels}\}$

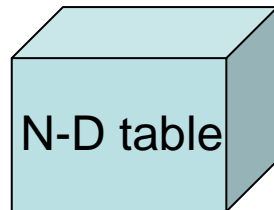
$$\tilde{J}^*(s, c) = \infty$$

for  $nc \in \{\text{possible commands for stocks}\}$

$$\tilde{J}^*(s, c) = \min \left( \tilde{J}^*(s, c), \phi(nc) + \mathbb{E}(J^*(s^*, c + nc) | s) \right);$$

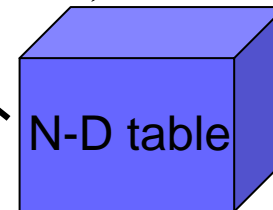
$J^* \leftarrow \tilde{J}^*;$

Same kind of algorithm (compared to 1-D gas storage valuation)



Cost [stock levels, prices, commands]

$J^*$  at  $t_n$



Cost [stock levels, prices, commands]

$J^*$  at  $t_{n+1}$

# 7.1 – Dynamic Programming Algorithm

Backward dynamic programming algorithm:

for  $t \leftarrow (M - 1)\Delta t$  to 0

for  $c \in \{\text{admissible stock levels}\}$

for  $s \in \{\text{possible price levels}\}$

$$\tilde{J}^*(s, c) = \infty$$

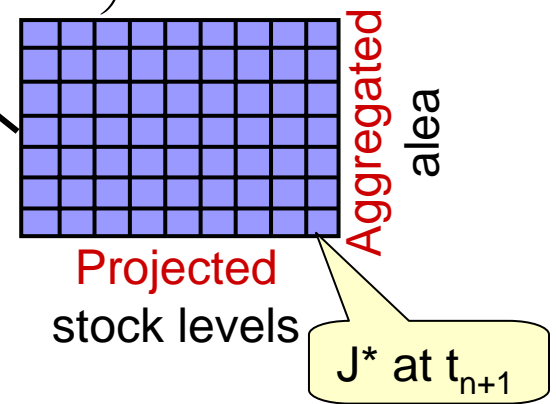
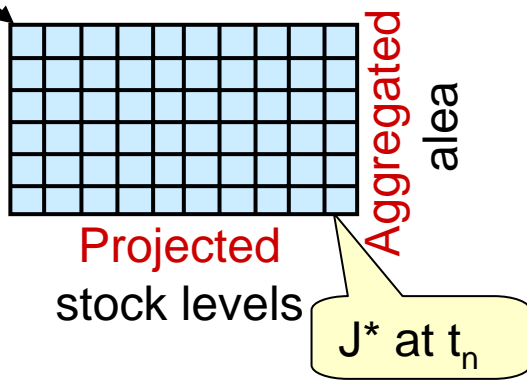
for  $nc \in \{\text{possible commands for stocks}\}$

$$\tilde{J}^*(s, c) = \min \left( \tilde{J}^*(s, c), \phi(nc) + \mathbb{E}(J^*(s^*, c + nc) | s) \right);$$

$J^* \leftarrow \tilde{J}^*$ ;

Same kind of algorithm (compared to 1-D gas storage valuation)

2D tables!! (implementation & mathematic optim)



# 7.2 – Application overview

## Part-1: Input file reading and initializations

- Read files and load data in each process memory space
- BG/L PFS → concurrent read (no pb!)
- PC cluster NFS → crash!  
→ P0 reads data files and broadcast data.

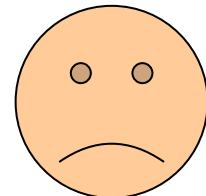
## Part-2: stochastic optimization

- N-D version of the previous algorithm
- + intermediate results storage  
→ Many IO & many communications & **huge computations**



## Part-3: stochastic simulation (compute risk associated to the optimized strategy)

- Independent Monte-Carlo forward simulations
- BUT: requires to read intermediate result of part 2  
→ Many IO & many communications & **few computations !**



# 8.1 – N-D data distribution strategy

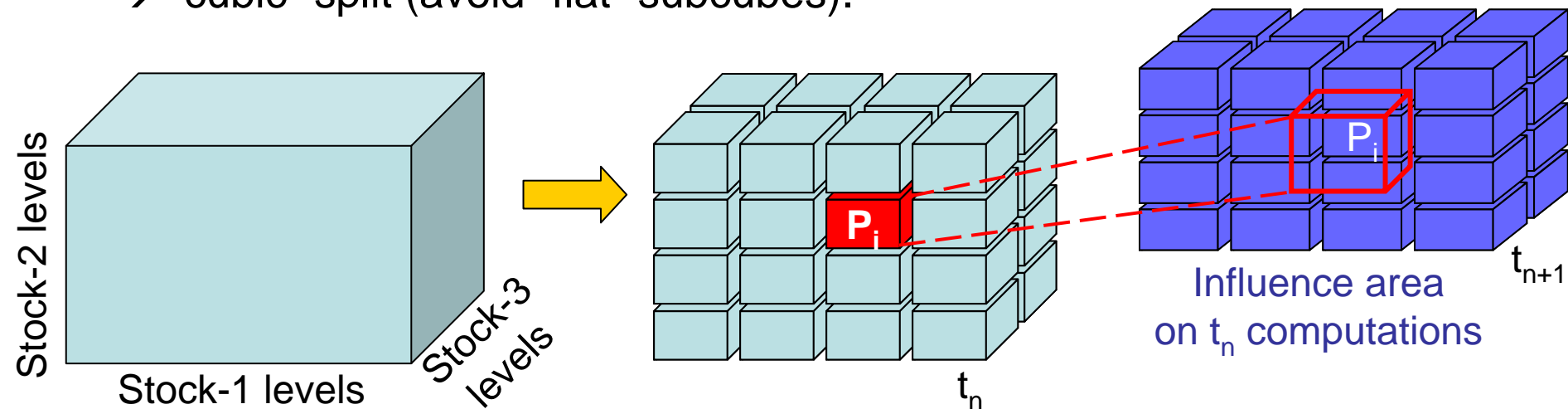
## Part-2 data distribution requirements:

- We have to split a N-stock hypercube of profits
- We have  $NbNodes = 2^k$  on Blue Gene/L (no constraint on PC-cluster)

Split and map  
a N-cube of data  
on a  $2^k$ -cube of  
computing nodes

## Objectives of the split:

- to load balance the data and the computations
- to minimize the communications of the « N-D influence area » !  
→ “cubic” split (avoid “flat” subcubes).



# 8.1 – N-D data distribution strategy

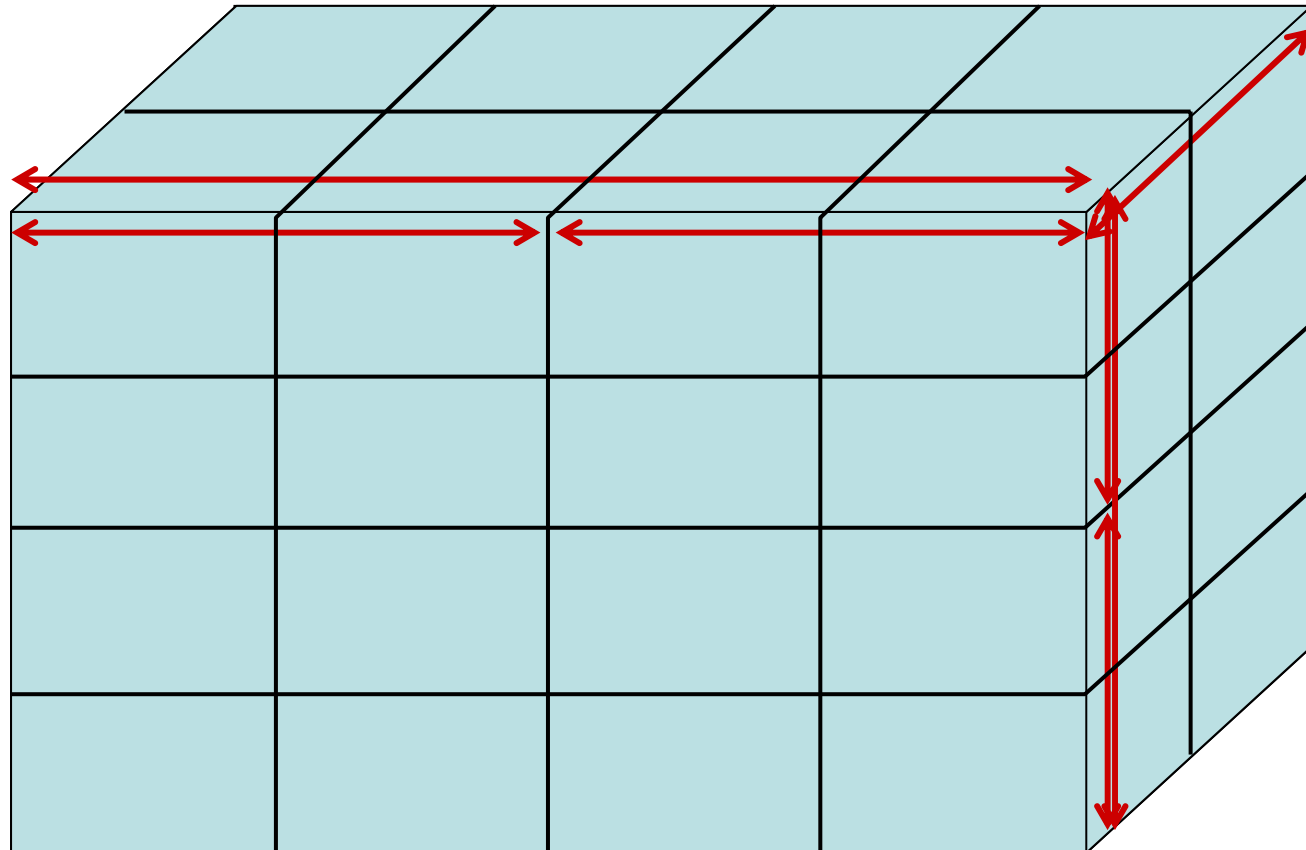
Considering a 3-stock problem:

$P_i$ : Compute the N-dimensional cube split in  $P = 2^k$  subcubes, avoiding « flat cubes »

- identify the dimension with the largest « subcube edges »
- split each of these subcube edges

repeat...

Each (MPI) process computes this data distribution map



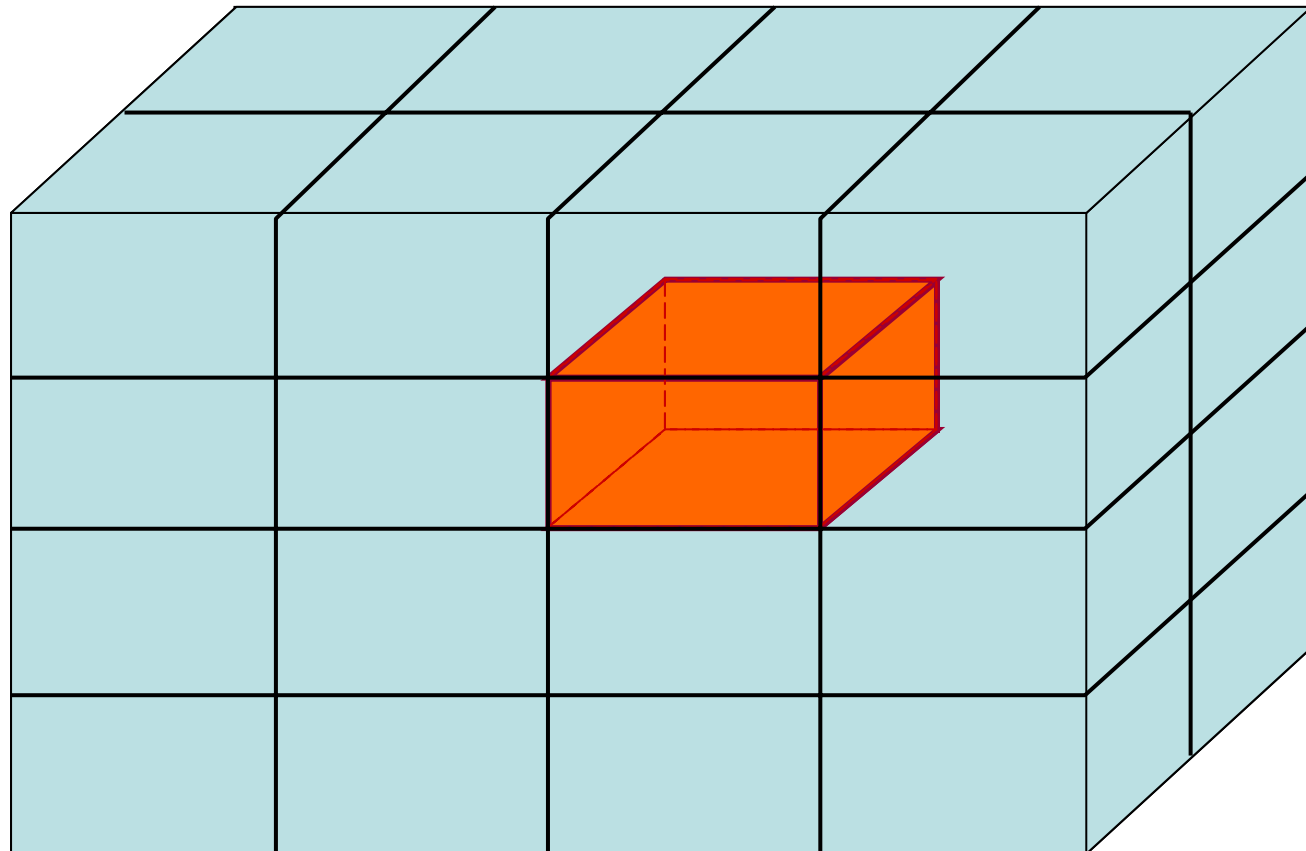


# 8.1 – N-D data distribution strategy

Considering a 3-stock problem:

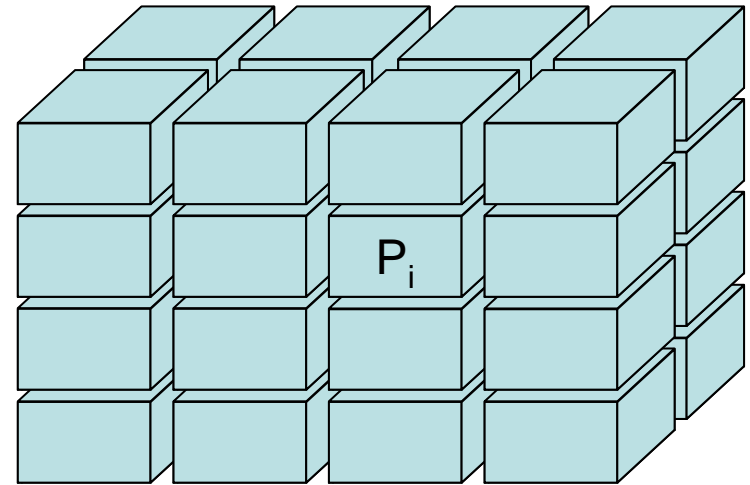
$P_i$ : Compute the N-dimensional cube split in  $P = 2^k$  subcubes, avoiding « flat cubes »

- identify its ND-subcube in the global distribution map
- allocate optimal in size datastructures



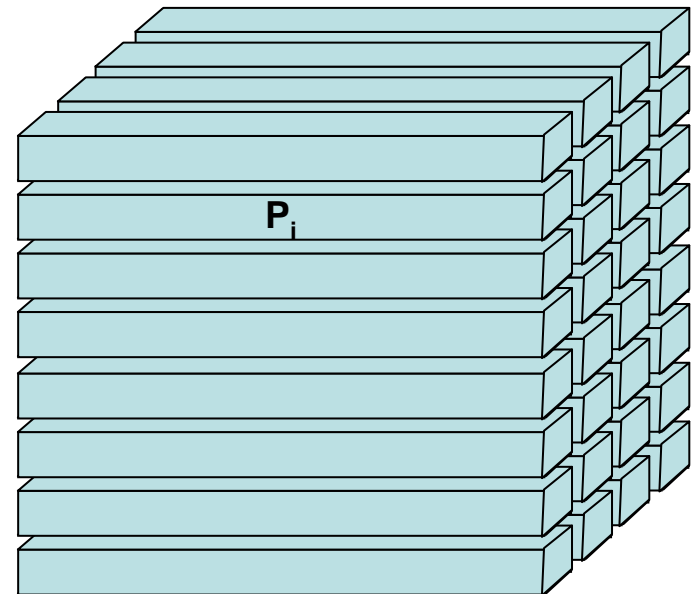
# 8.1 – N-D data distribution strategy

- When all dimensions can be split each processor manages very “cubic” ND-subcubes:



- When at least one dimension cannot be split processors manage more “flat” ND-subcubes:

*Some stocks are aggregated into one large stock, leading to optimized but complex computations. Then this dimension cannot be split, and parallelization is not optimal.*



# 8.2 – N-D communication scheme

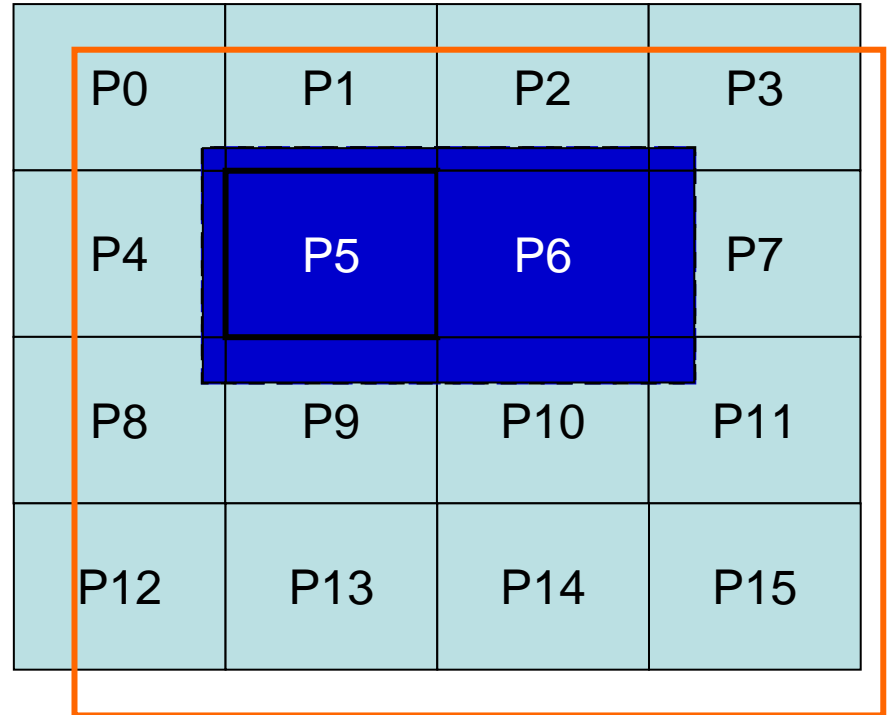
Considering a 2-stock problem:

What happens on **P5** (for example) ?

At  $t_n$ , it manages a new 2D-subcube, in the new 2D-cube

... influenced by a 2D-subcube of  $t_{n+1}$  data

So it has to establish its routing plan:



**P5** Routing plan:

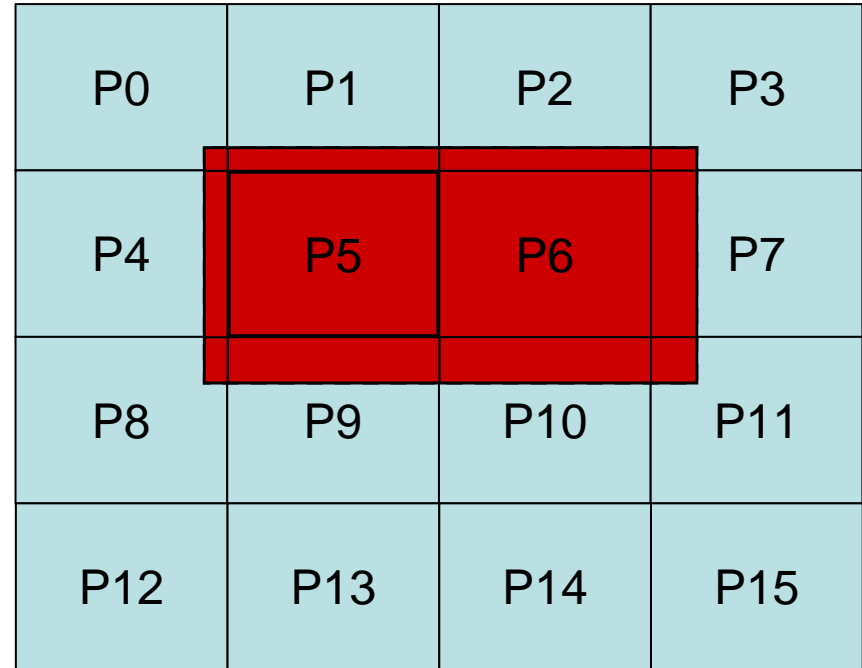
Proc	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Recv																
Send																

# 8.2 – N-D communication scheme

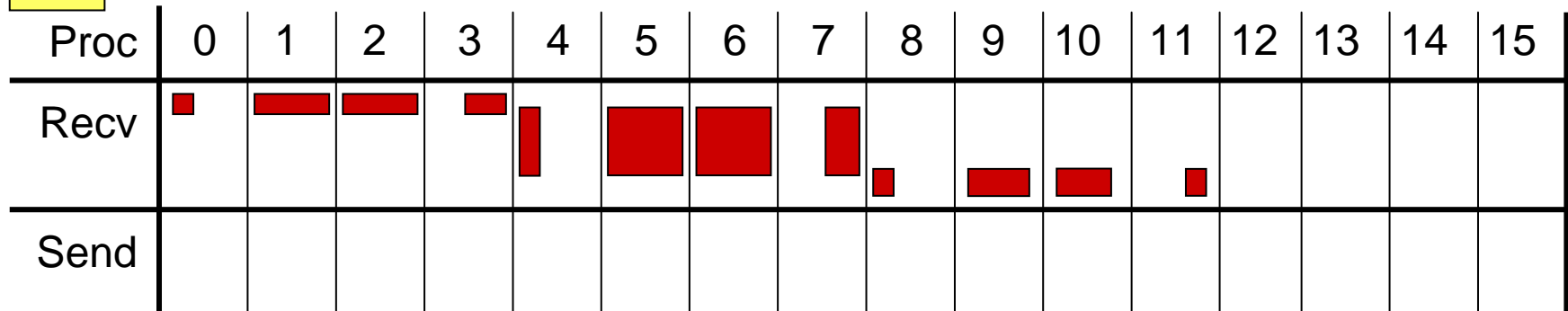
Considering a 2-stock problem:

What happens on **P5** (for example) ?

It determines all 2D-subcubes it has to receive from other processors



**P5** Routing plan:



# 8.2 – N-D communication scheme

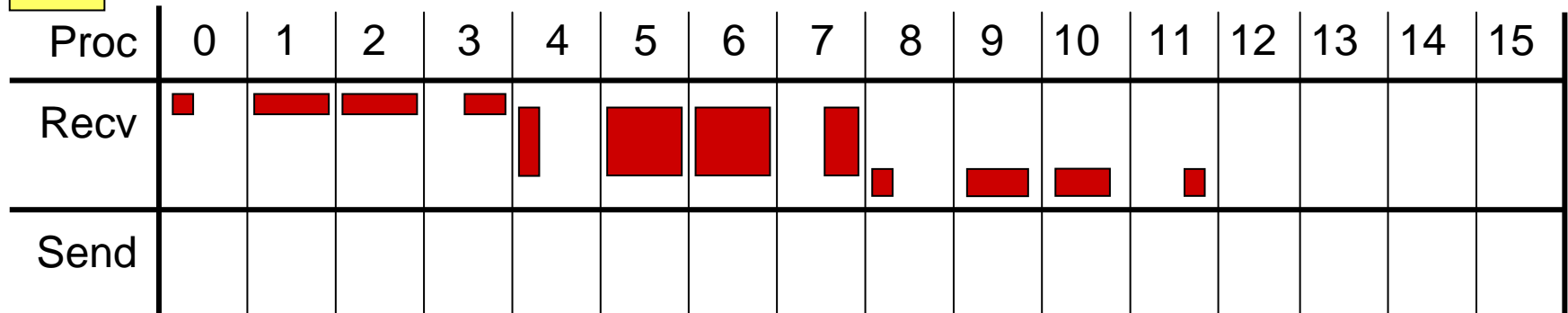
Considering a 2-stock problem:

What happens on **P5** (for example) ?

It determines all 2D-subcubes it has to receive from other processors

P0	P1	P2	P3
P4	P5	P6	P7
P8	P9	P10	P11
P12	P13	P14	P15

**P5** Routing plan:



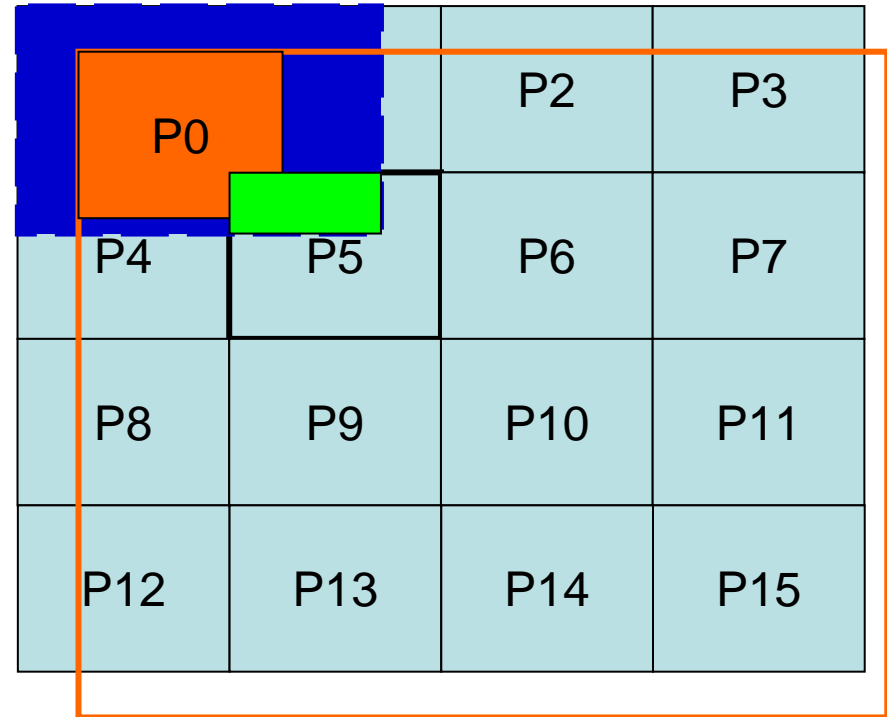
# 8.2 – N-D communication scheme

Considering a 2-stock problem:

What happens on **P5** (for example) ?

It determines all 2D-subcubes it has to send to other processors:

- compute « influence area » of P0
- compute the intersection with its  $t_{n+1}$  2D-subcube of data



**P5** Routing plan:

Proc	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Recv	■	■ ■ ■	■ ■ ■	■	■	■ ■ ■	■ ■ ■	■	■	■ ■ ■	■ ■ ■	■				
Send	■															

# 8.2 – N-D communication scheme

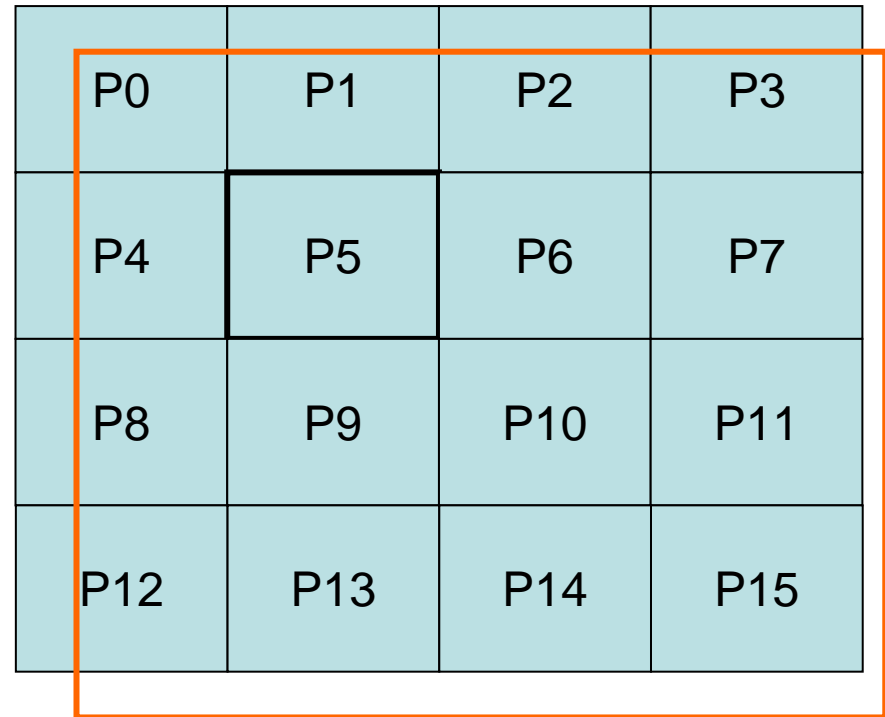
Considering a 2-stock problem:

What happens on **P5** (for example) ?

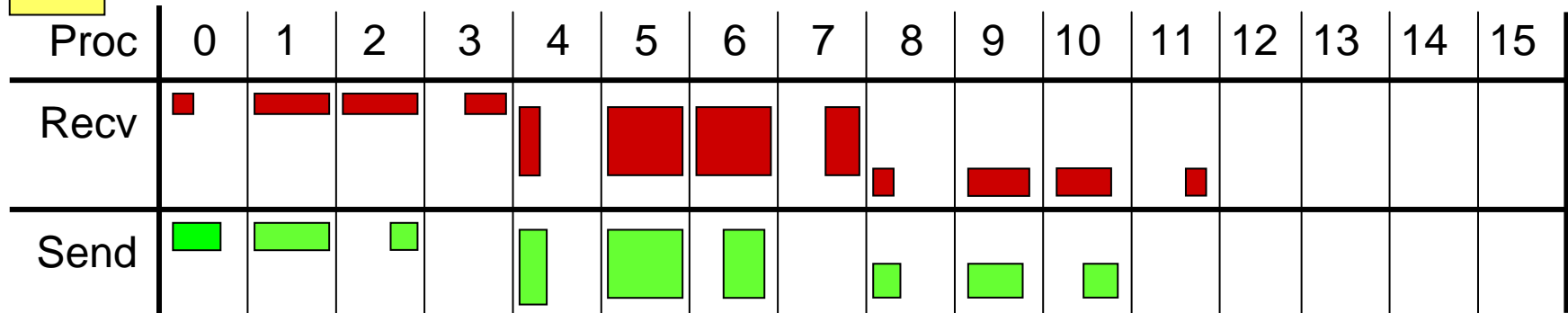
It determines all 2D-subcubes it has to send to other processors:

- repeat with other processors...

The routing plan of P5 is complete!  
 → Execute it quickly!



**P5** Routing plan:



# 8.3 – C++ implementation

## Parallelization:

- MPI: Mpich-1, OpenMPI, IBM MPI  
communication routines: MPI\_Issend, MPI\_Irecv, MPI\_Wait
  - overlap all communications when executing a routing plan (to speedup)
  - do not use “extra communication buffers” (to size up)
- + multithreading: Intel TBB or OpenMP
  - to speedup and to size up more (than using only message passing)

## Scientific computing libraries:

Blitz++, Boost, Clapack, Sprng.

## Total:

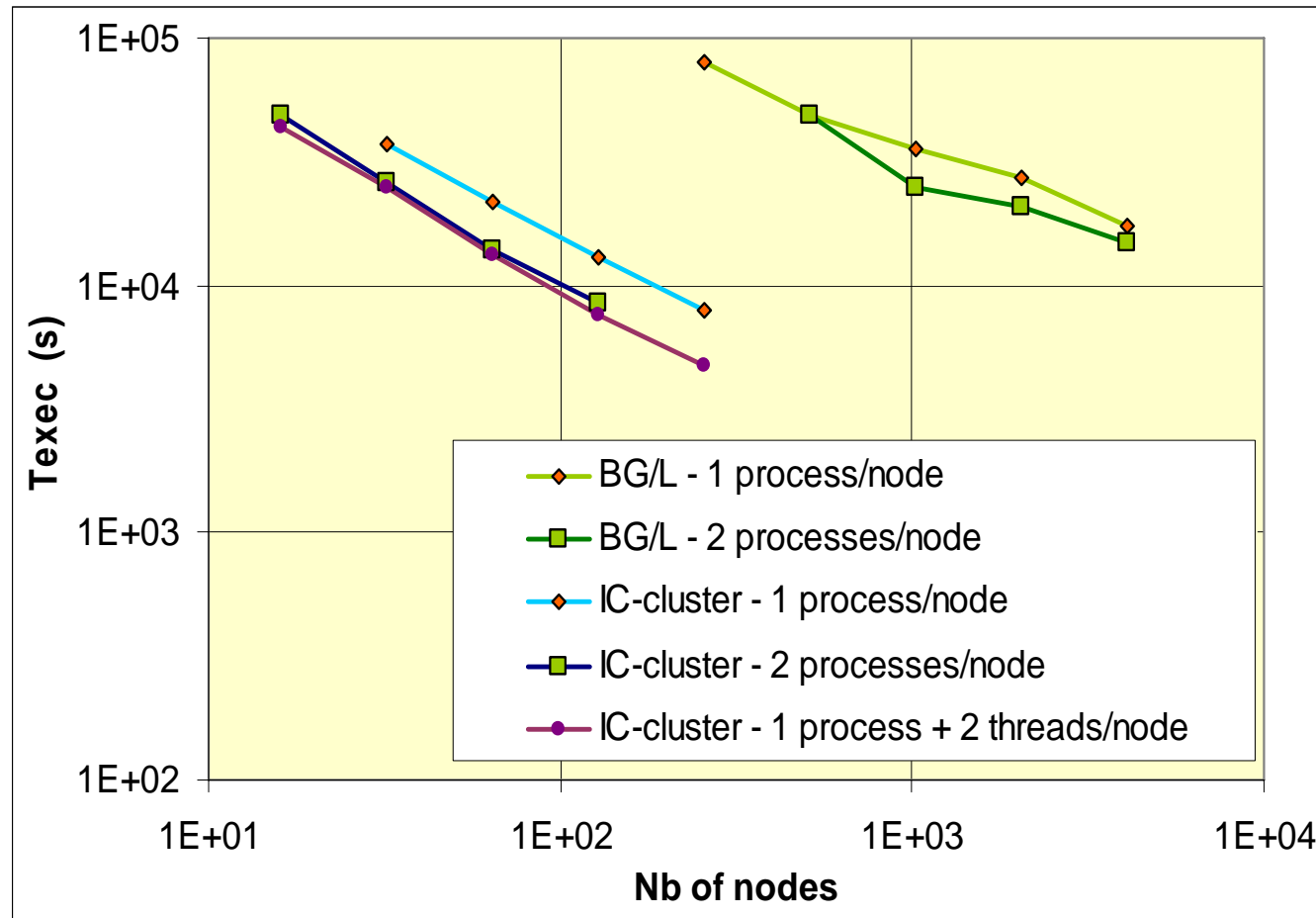
- 57000 lines of C++ code
- 10% for parallelization management
- Same source code on PC-cluster and Blue Gene/L



# 9.1 – Perf. of the optimization part

Performances of optimization part of a “7-stocks / 10-state-vars” computation  
 Using: **MPICH-1** and **TBB** on IC-cluster (a PC cluster), **IBM MPI** on BG/L

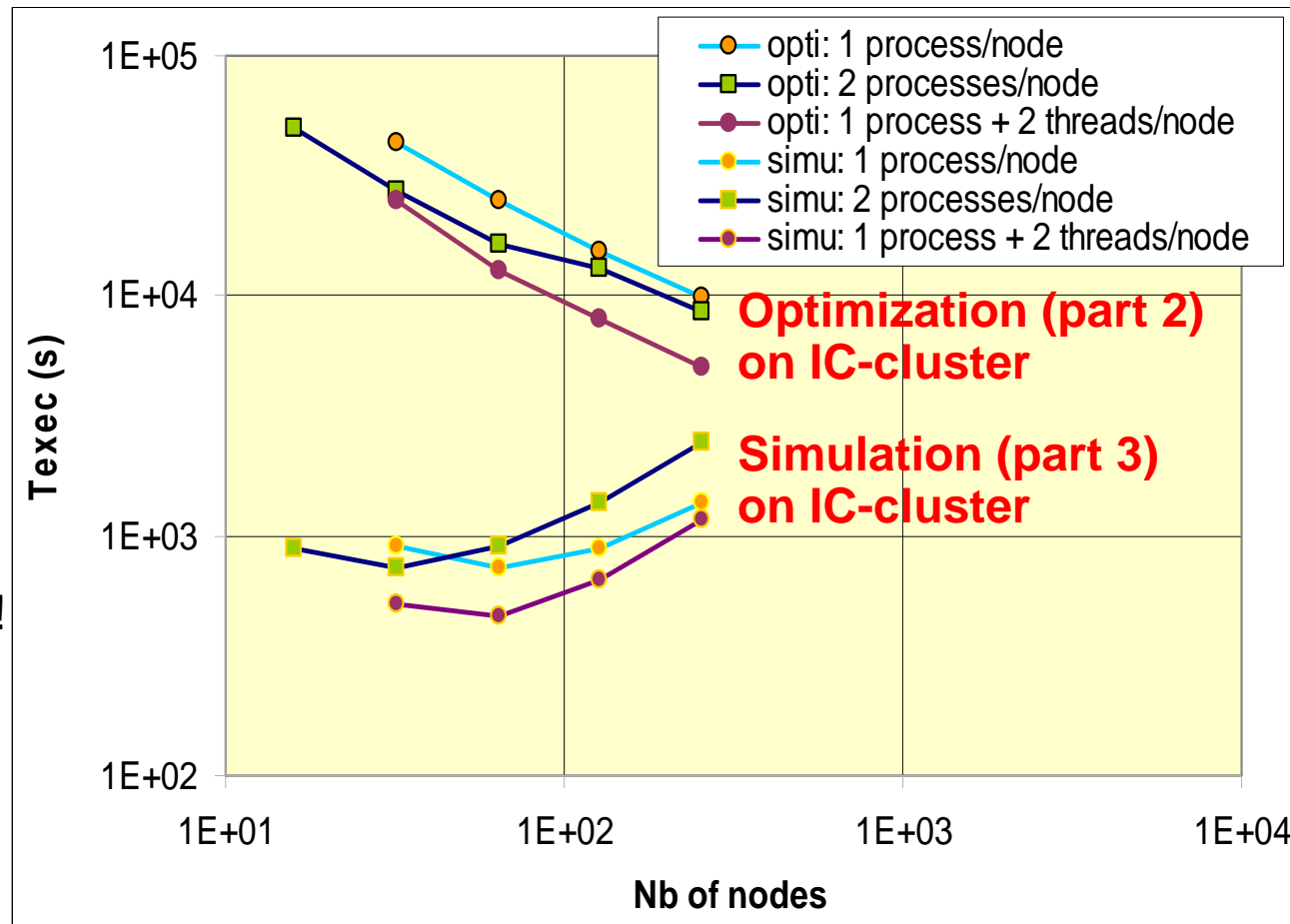
- Similar curves (compared to 1-D gas storage valuation)
- Performance on BG/L:
  - less important ...
  - high fluctuations ...Strange! We investigate.
- MPICH-1 on IC-cluster:
  - 1 MPI process + 2 threads: not better than 2 MPI processes,
  - many failures beyond 64 nodes.



# 9.2 – Perf. of optim. & simu. parts

Details of the performances of a “7-stocks / 10-state-vars” computation  
 Using: **OpenMPI** and **OpenMP** on IC-cluster (a PC-cluster)

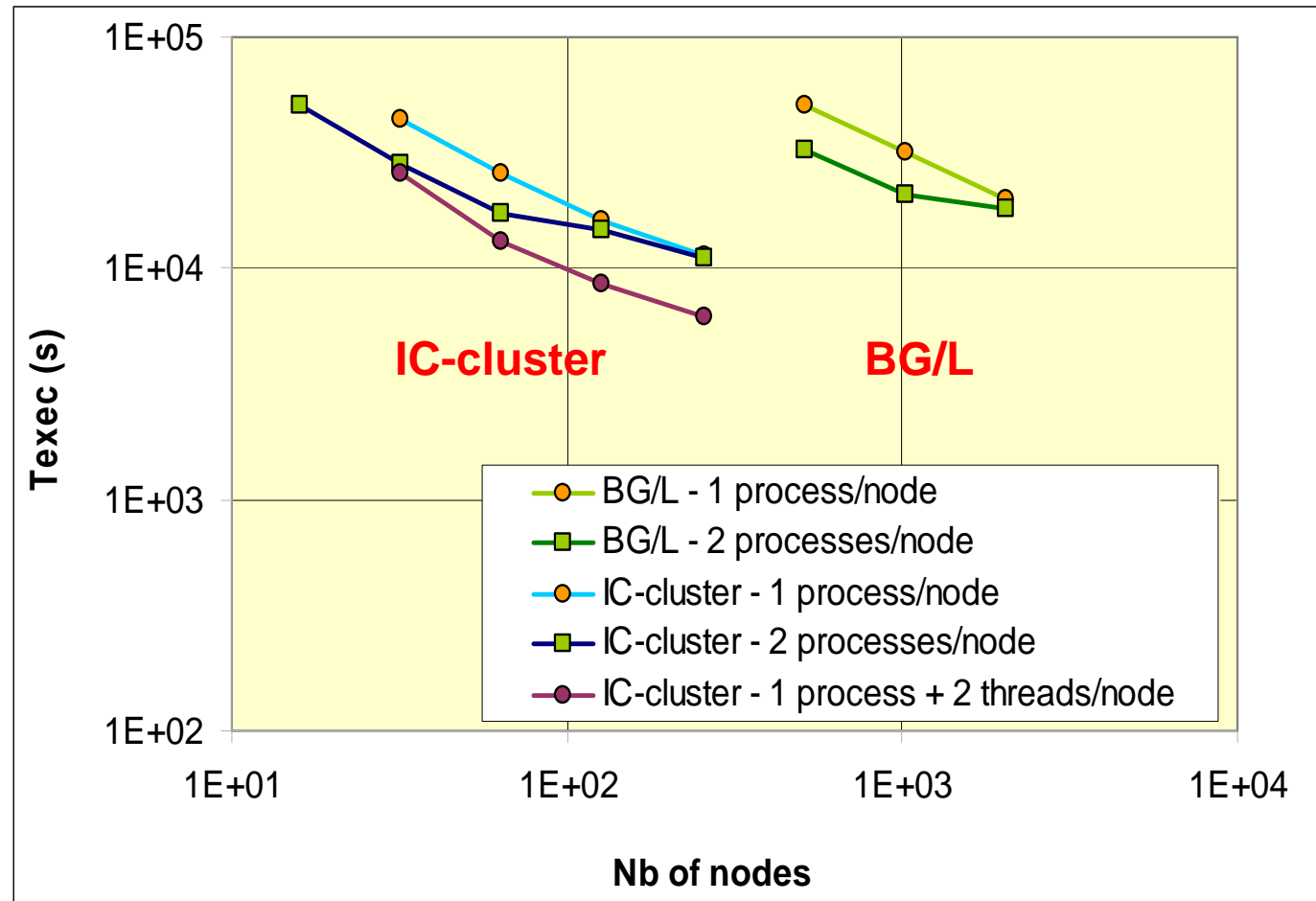
- OpenMPI on IC-cluster:
  - no failure beyond 64 nodes
  - slower than MPICH-1 (approx. +20%)
  - 1 MPI process + 2 threads: better than 2 MPI processes,
- Optimization: scales  
 Simulation: does not scale!
- Best implementation: 1 process + 2 threads, (both for opti & simu)



# 9.2 – Complete computation perf.

Performances of a “7-stocks / 10-state-vars” entire computation (parts 1, 2 and 3):  
 Using: **OpenMPI** and **OpenMP** on IC-cluster (a PC cluster), **IBM MPI** on BG/L

- Part 3 (simulation) limits the speedup
- 1 MPI process + 2 threads/node: best implementation
- Threads not available on BG/L,
- OpenMP threads on IC cluster
- Performance measurement on BG/L in progress...



# Conclusion and perspectives:

# 10.1 - Conclusion

## From parallel computing point of view:

Complex parallelization (with many comms. and IOs)

Speedup, size up and scalability on PC-cluster and BG/L supercomputers

More experimentations are required on BG/L

New experimentations are required on PC-cluster with different MPI libraries

Improvement of the stochastic simulation distributed algorithm is required...

## From a user point of view:

Usual limit is 4-5 state variables on sequential machines!

Our software :

- ✓ currently allows to process a « 7-stock/10-state-vars » problem in less than 2h on our PC-cluster.
- ✓ supports different theoretical models and allows to quickly experiment and evaluate new models

# 10.2 - Perspectives

## Next steps:

- More experiments on BG/L – 8192 processors (using IBM MPI)
- **Experiments on BG/P - 32000 processors (using IBM MPI + OpenMP)**
- Serial optimizations on BG, with the help of IBM
- Improvement of the stochastic simulations (part-3)



May-June 2008

## Exploitation for energy management:

- Quick implementation and large scale evaluation of new models
- Design and experimentation of new and larger use cases
- **Towards a global optimization of EDF energy production.**



May-December 2008

## Multi-site Grid experiments:

- Multi-site experiment on Grid'5000
- Impact of WAN on performances ?  
*collaboration with AIGorille/INRIA team (Nancy)*
- Improvement of process mapping on a multi-site Grid ?  
*collaboration with Reso/INRIA team (Lyon)*



June 2008 ...

# Stochastic control optimization & simulation applied to energy management: From 1-D to N-D problem distributions, on clusters, supercomputers and Grids.

**Questions ?**

