P systems simulations on massively parallel architectures

José M. Cecilia[1], José M. García[1], Ginés D. Guerrero[1], Miguel A. Martínez del Amor[2], Mario J. Pérez-Jiménez[2], Manuel Ujaldón[3]

[1]Parallel Computer Architecture Research Group
University of Murcia (Spain)

[2]Research Group on Natural Computing
University of Seville (Spain)

[3]Computer Architecture Department
University of Malaga (Spain)

09-03-2012

Outline

Outline

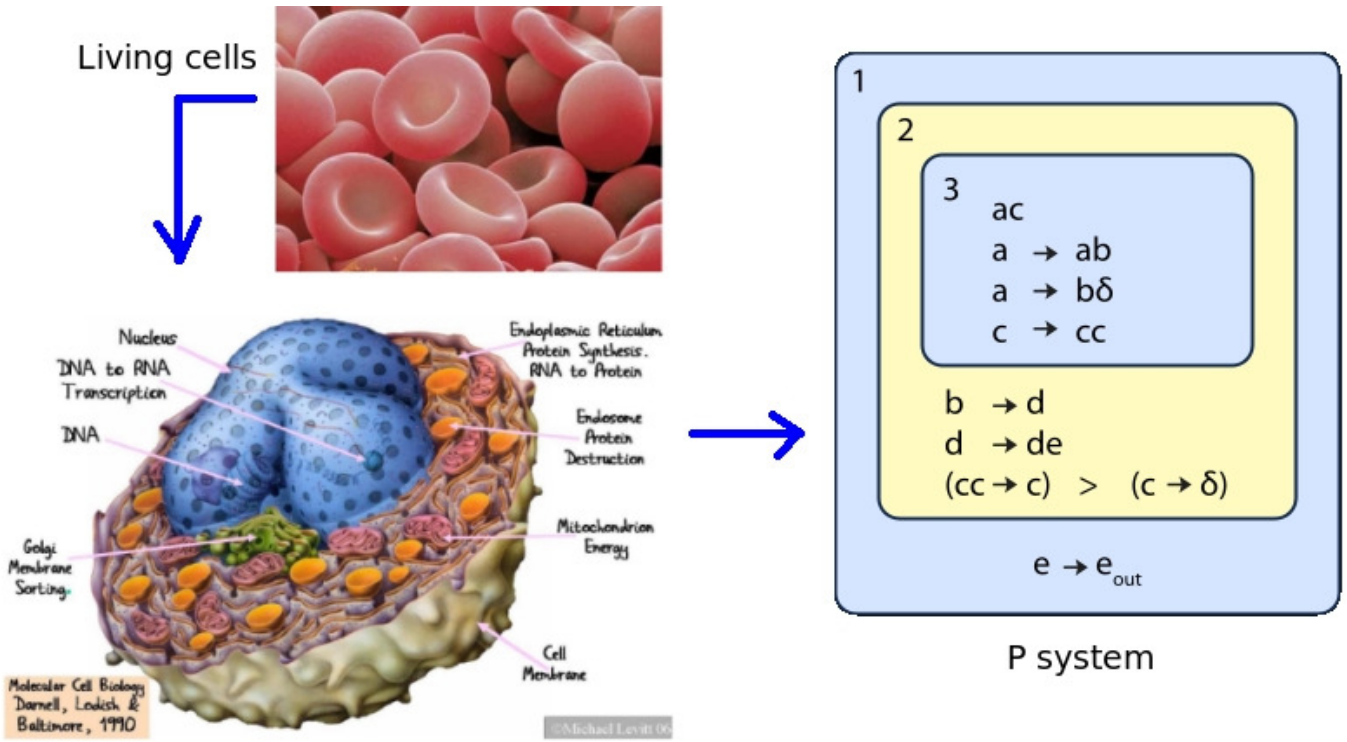
Motivation

P system definition

- Membrane computing is an emergent branch in Natural Computing.
- It is based on the behaviour of living cells to define bio-inspired computation devices, called **P systems**.
- P systems computational features include:
  - Polynomial time solutions to NP-Complete problems by trading time for space.
  - Modeling of biological phenomena.
- We focus on the family of **recognizer P systems with active membranes**.

Motivation

P system apperance



Motivation

P system simulation

- No P systems implementations neither *in vivo* nor *in vitro*.
- Only P systems simulations on silicon.
- Simulation of P systems poses challenges in three different aspects:
  - The intrinsic massively parallelism (like cells).
  - The exponential memory requirements (trading time for space).
  - The non-intensive floating point nature (for recognizer P systems).

Work Motivation: Which parallel platform is best suited for the P system simulation?

Outline

The SAT problem

- The first-known NP-complete problem (Stephen Cook, 1971 [3]).
- Formula in CNF with a conjuction of clauses formed by a disjuntion of literals.
- Literal is either a variable or its negation.
- To determine whether exists a truth assignment to its n variables.
- Paramount importance in many computer science areas: Hardware design, algorithmc, etc.

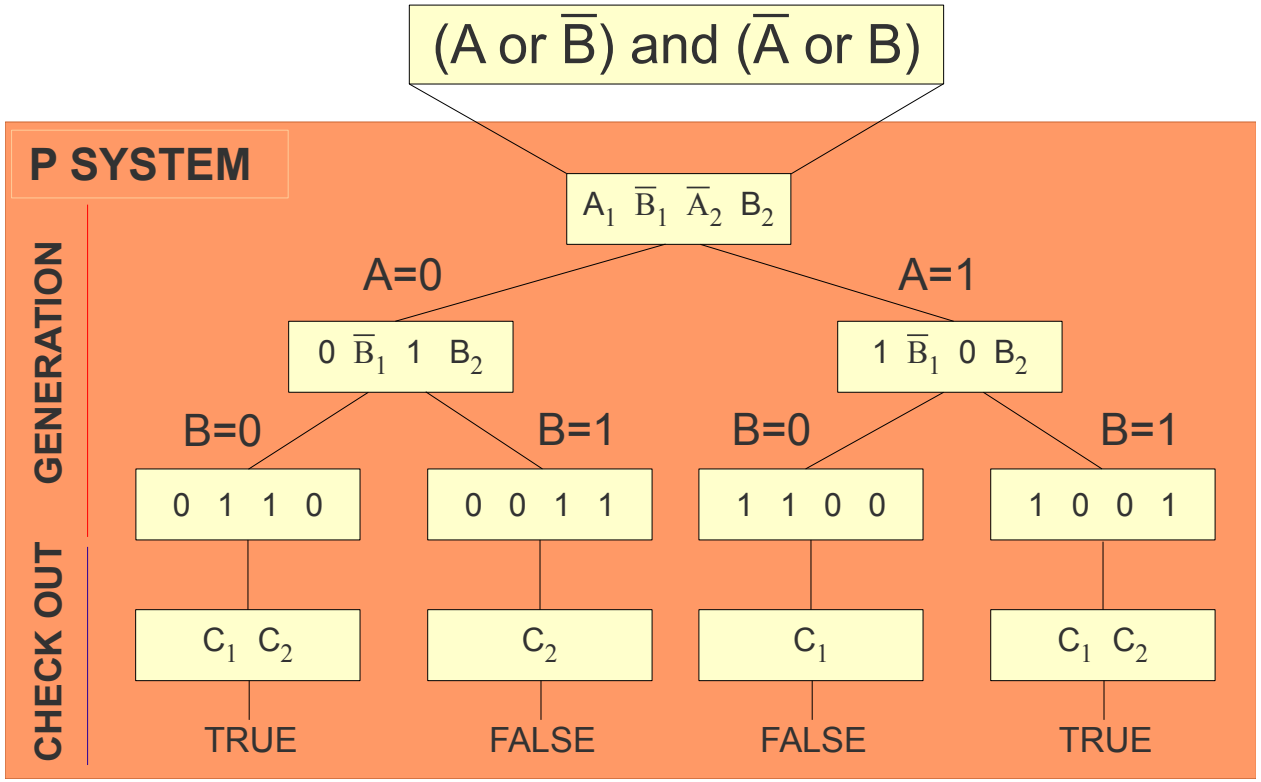
(A or ¬B) and (¬A or B)

Solving the SAT problem through P systems

- The P system computation for SAT is described in [4].
- It can be summarized in four stages:
  1. **Generation.** All possible truth assignments to the variables are generated by using P systems rules. 2^n internal membranes are created and each one encodes a truth assignment to the variables of the formula.
  2. **Synchronization.** The objects encoding a true clause (a partial solution to the CNF formula) are unified in the membrane.
  3. **Check out.** The goal here is to determine how many (and which) clauses are *true* in every internal membrane (that is, by the assignment that represents).
  4. **Output.** Internal membranes encoding a solution send an object to the skin. If the skin has such object from some membrane, the object *Yes* is sent to the environment. Otherwise, the object *No* is sent.

Solving the SAT problem through P systems (II)

A small example



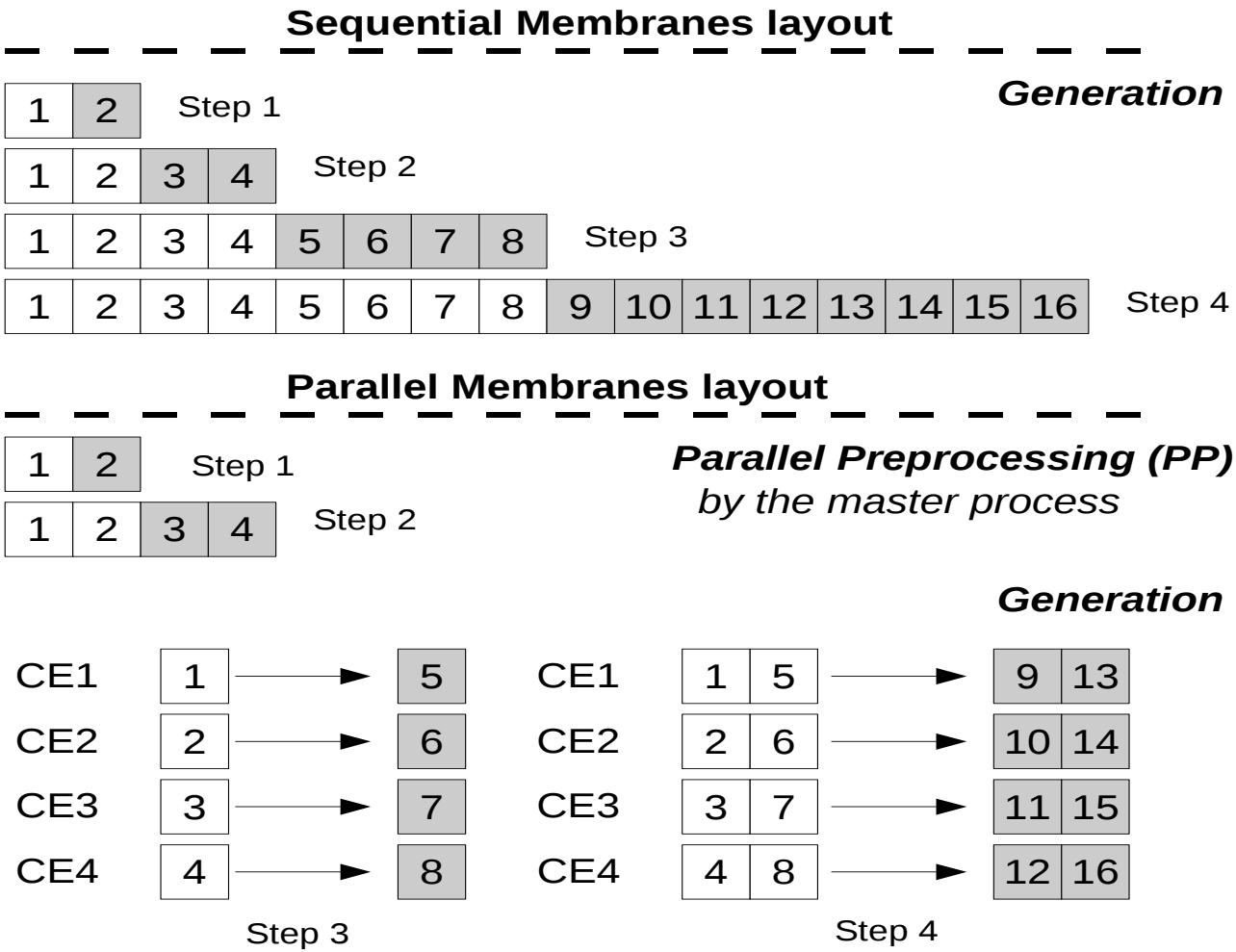
Outline

Characteristics of P system simulation

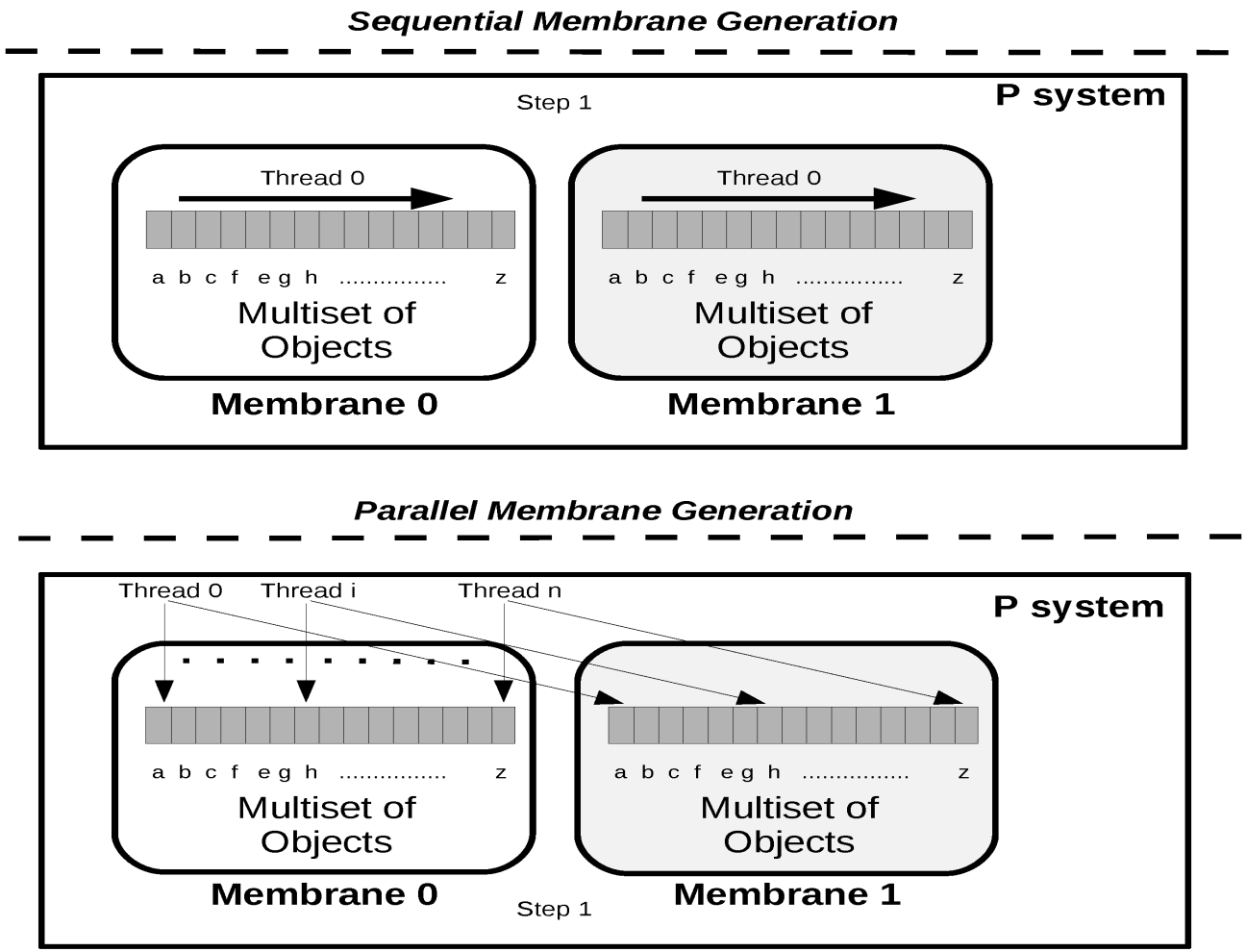
1. P systems are massively parallel, having a double level of parallelism:
   - The first level of parallelism is among membranes (coarse-grained).
   - The second level of parallelism is within each membrane (fine-grained).
2. P systems create an exponential workspace (for the SAT equation 1).

$$Size = 2^n(membranes) * k(objects) * 4(uint)Bytes. \quad (1)$$

Parallelism among membranes

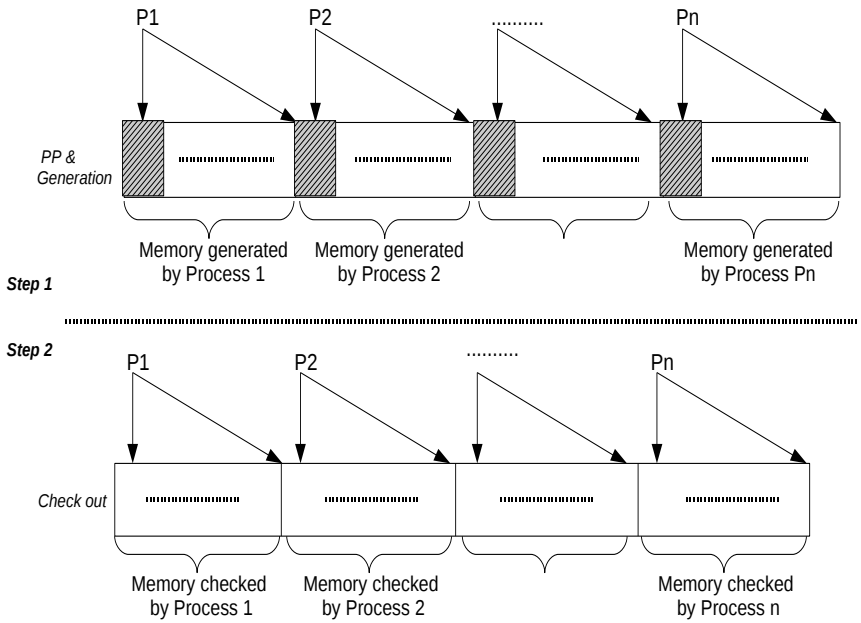


Parallelism within each membrane



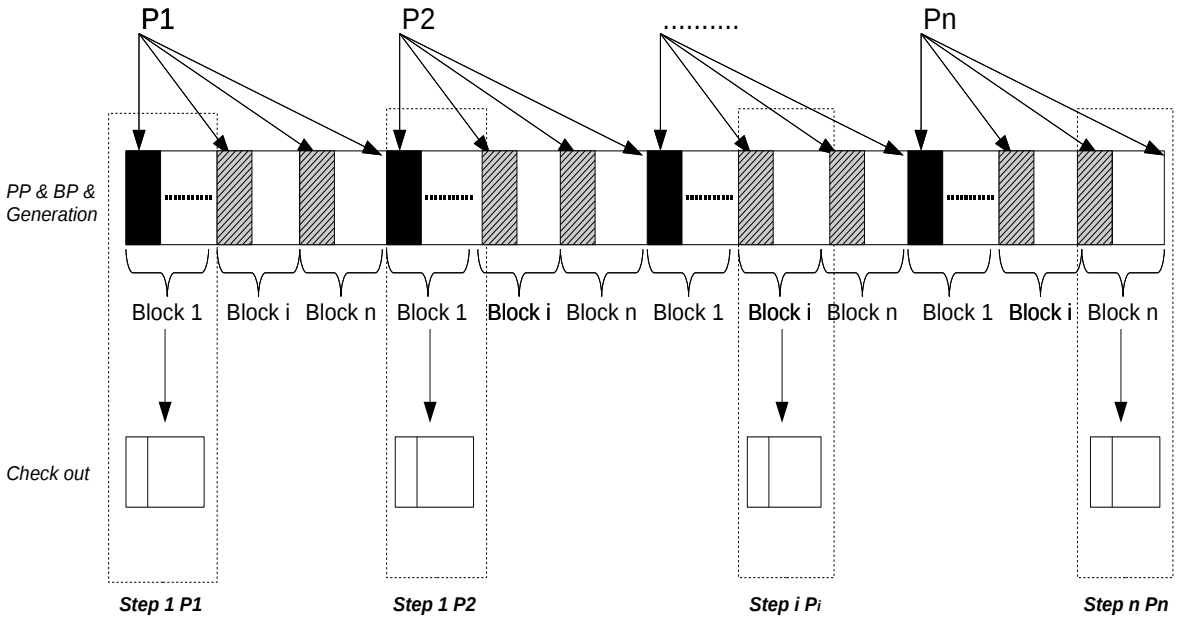
Shared memory implementation (OpenMP)

- The shared memory space is equally distributed among processes.
- Master process creates as many membranes as OpenMP processes.
- Initial membranes are located at the beginning of each memory space for each process.
- Each process performs generation stage in parallel, and then Check out.



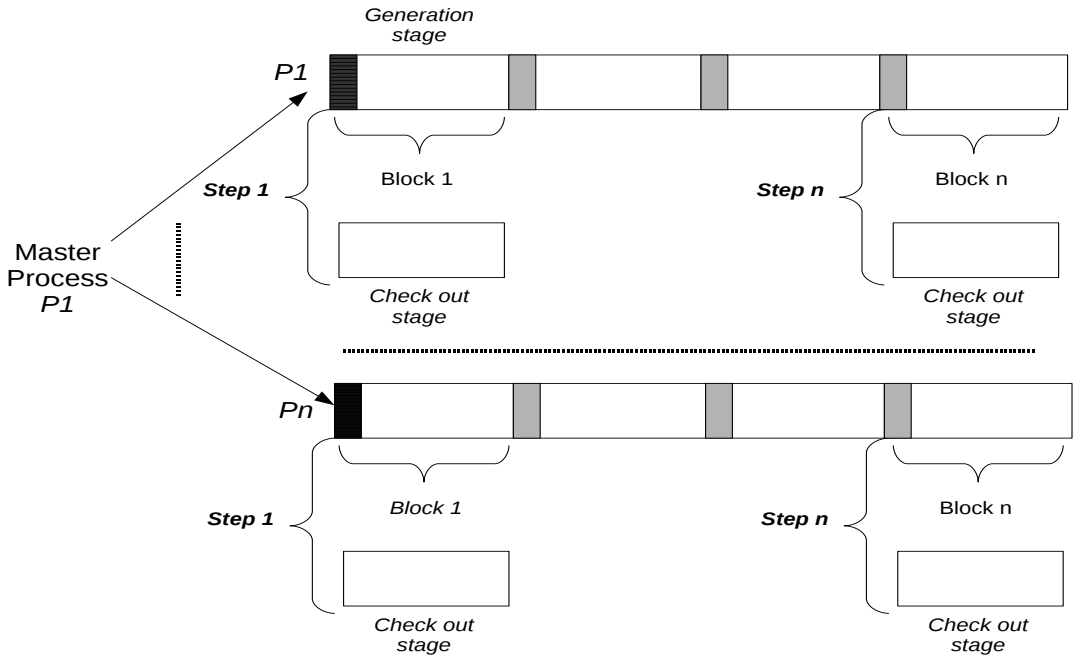
Data locality on the Shared memory implementation (OpenMP)

- Many cache misses with the previous (first writes and then reads).
- Block-based data layout to improve locality.



Distributed memory implementation(MPI)

- Similar to the previous one but in a distributed memory.
- Both implementations: Blocking and non-Blocking.

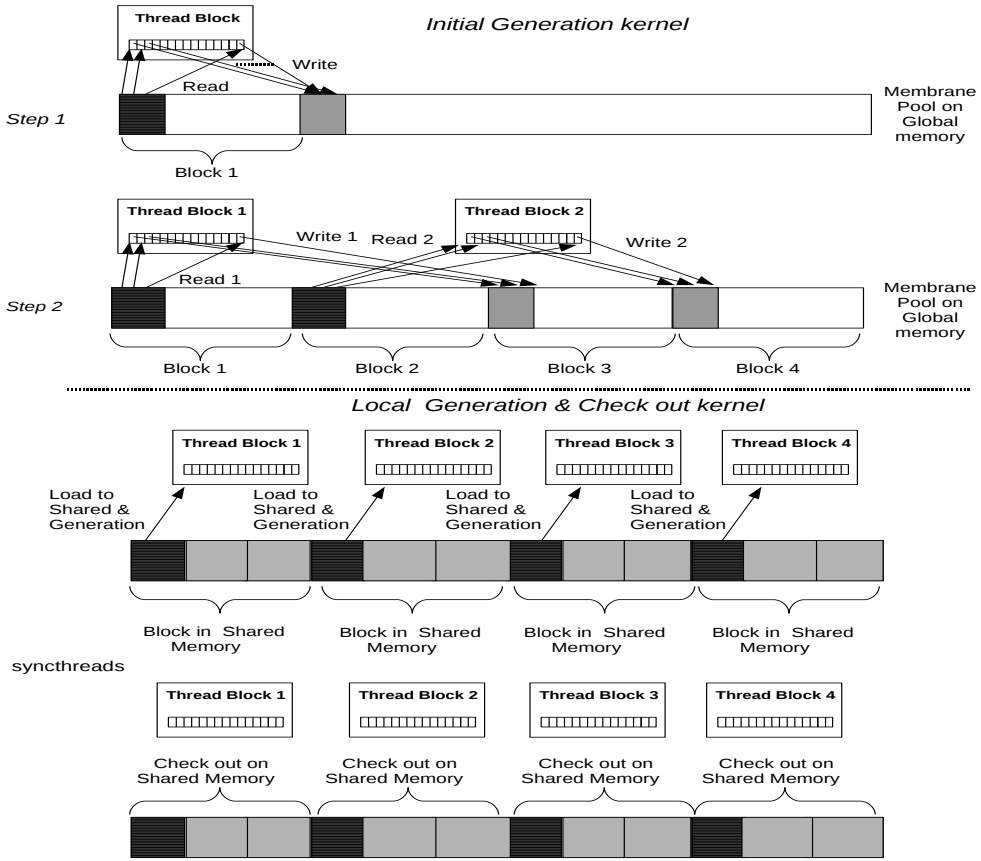


Implementation on GPUs (CUDA)

- A thread block per membrane and a thread per object (or set of objects).
- CPU creates as many membranes as GPUs in the system.
- First attempt with two kernels: First Generation and then Check out.
- Need of global synchronization among blocks.
- Performance issues: Only global memory accesses

Blocking on GPUs (CUDA)

Taking advantage of the shared memory.

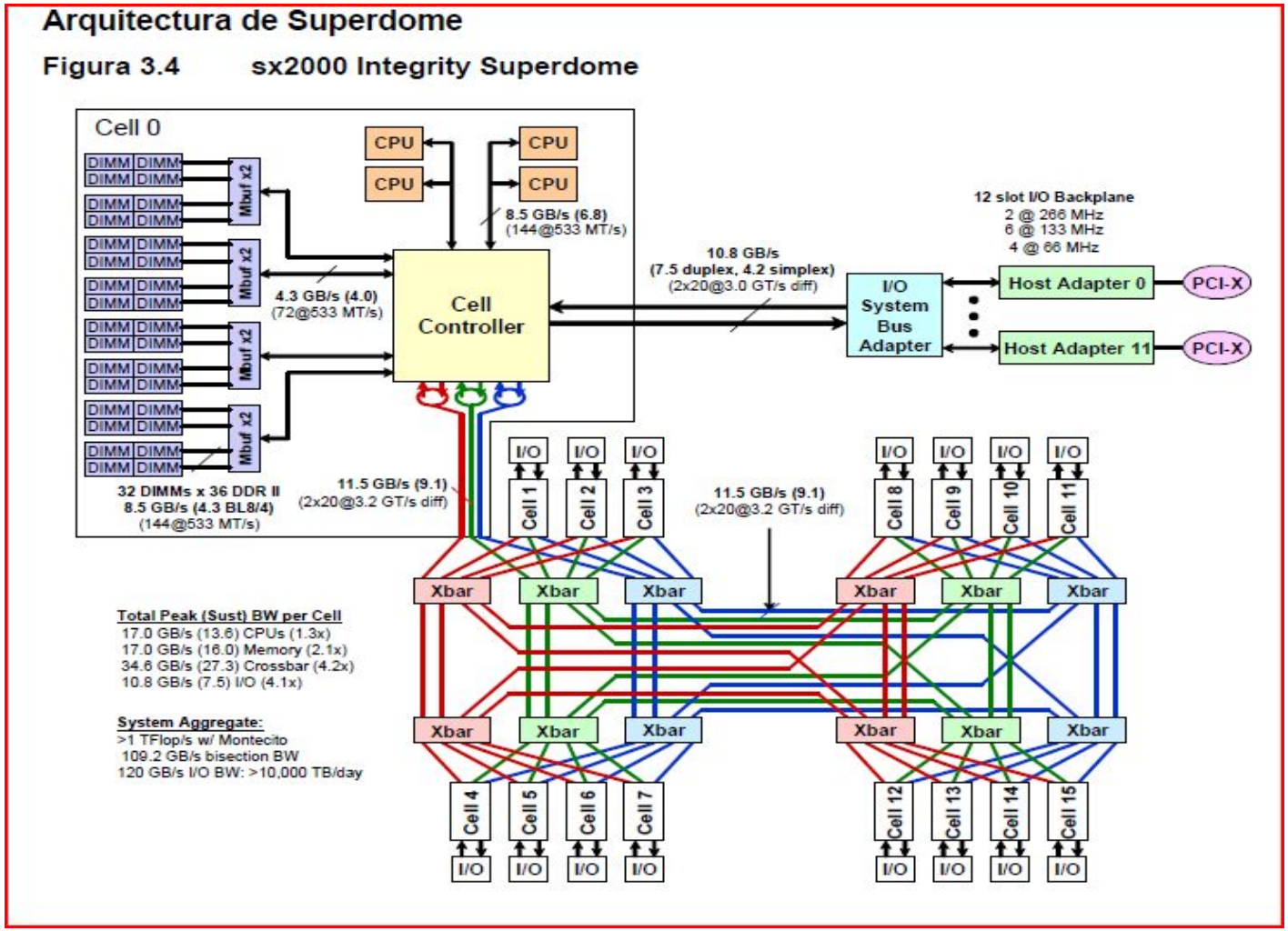


Outline

Hardware environment

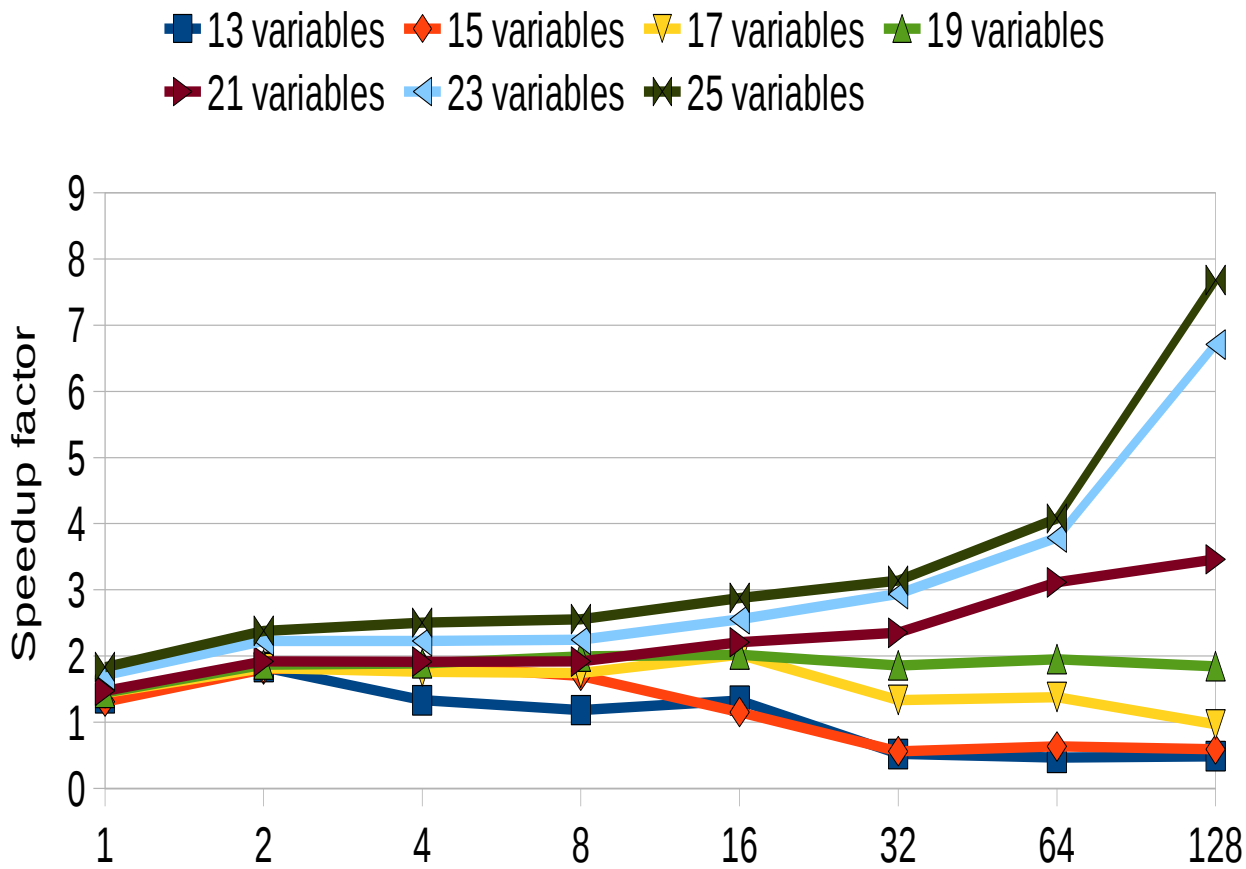
- Shared memory platform is a HP Integrity Superdome SX2000 endowed with 64 CPUs
  - Intel Itanium 2 dual-core Montvale.
  - 16 Kbytes L1, 256 Kbytes L2, 18 Mbytes L3.
  - Total DRAM memory available is 1.5 Tbytes.
  - Interconnection network is a 4x DDR Infiniband.
- Distributed memory system is a HP BladeSystem with 102 nodes
  - Dual socket containing a quad-core Intel Xeon E5450.
  - 12 MBytes L2 cache.
  - DRAM memory capacity for the whole system is 1072 Gbytes.
  - Interconnection network is also a 4x DDR Infiniband.
- GPU-based platform includes
  - Four-socket, quad-core Intel Xeon E5530 socket.
  - 8 MBytes L2 cache.
  - DRAM memory capacity 16 Gbytes
  - 4 Tesla C1060 with up to 16Gbytes of video memory.

Superdome SX2000 architecture

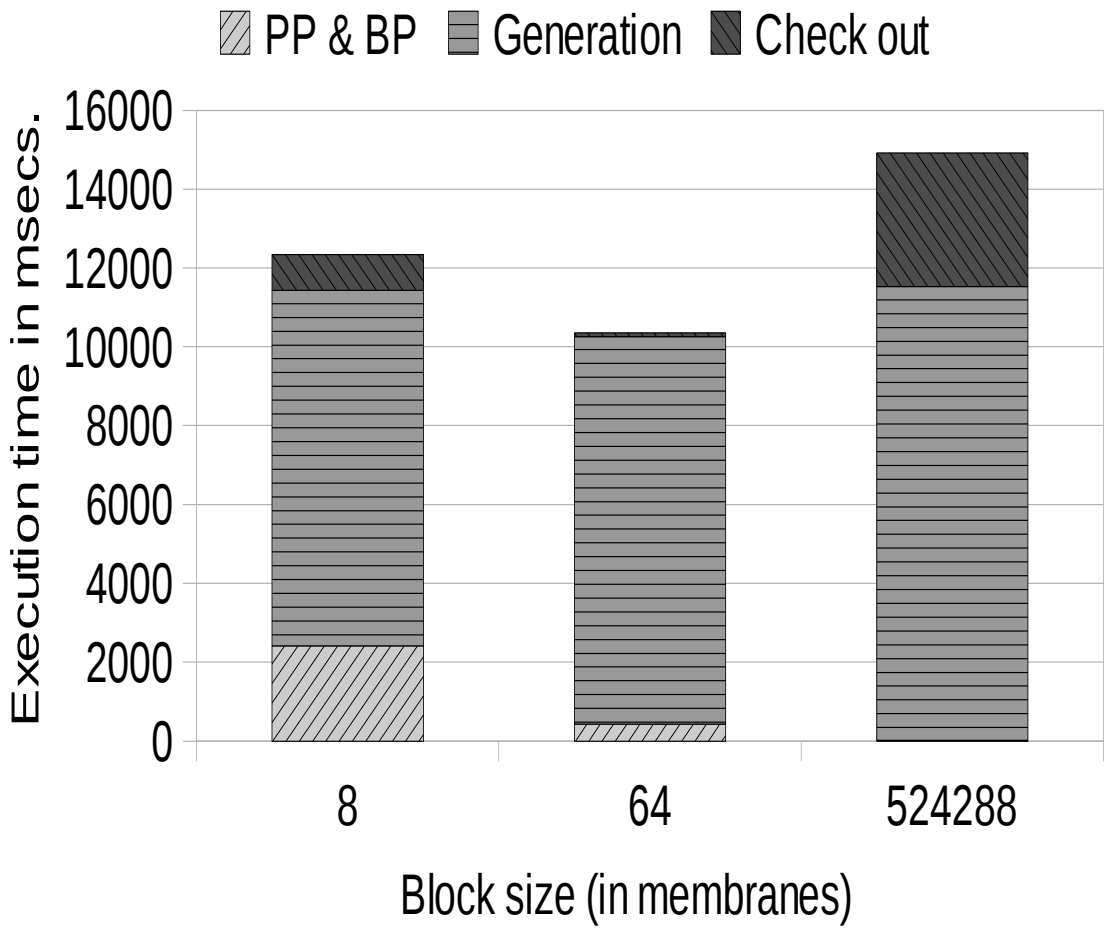


Blocking Vs Non-blocking algorithm

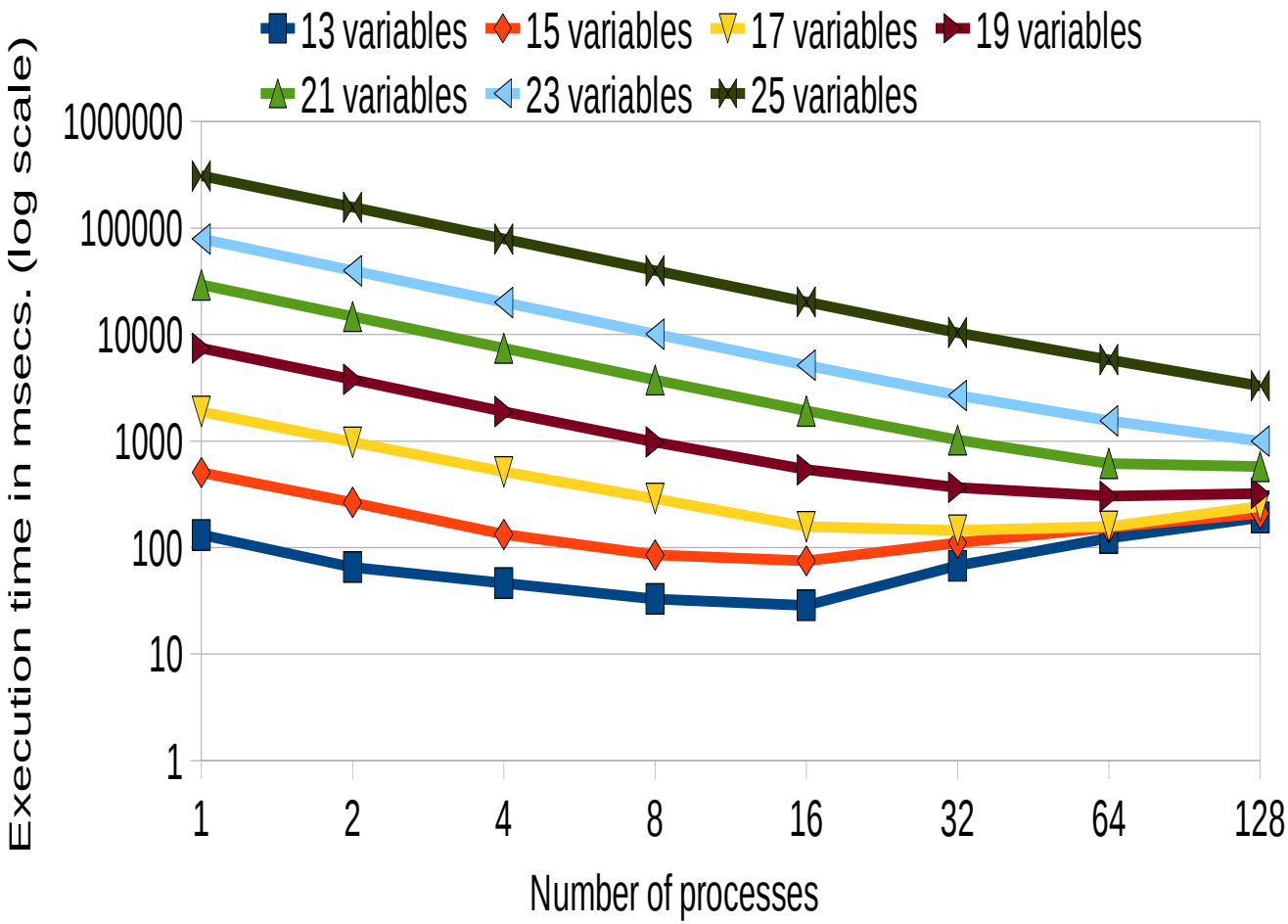
Performance increases with both: Problem size and number of processes



Effects of the blocking size



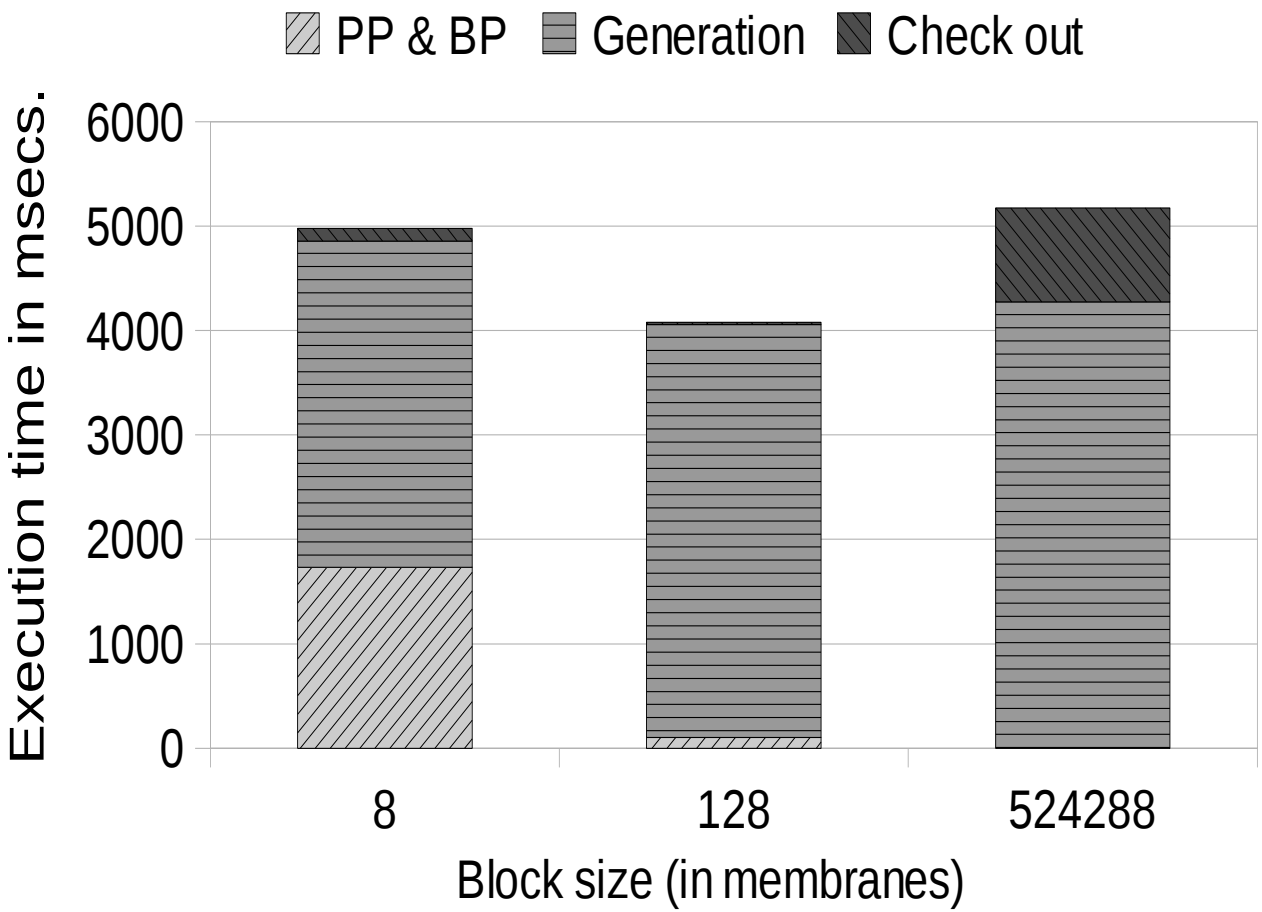
Execution time depending on the problem size



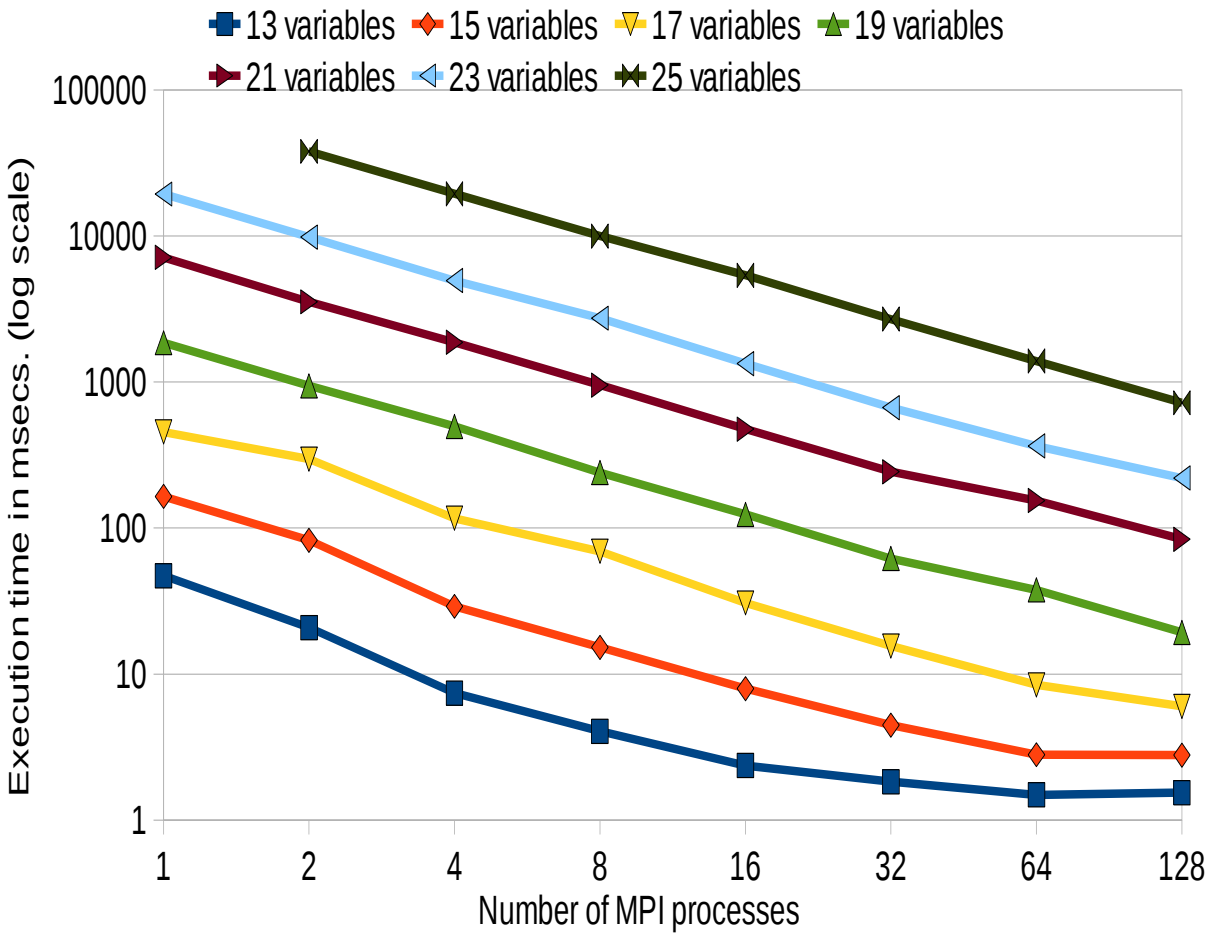
Blocking Vs non-Blocking

- The blocking technique reaches up to 2x versus non-blocking alternative (25 variables)
- Memory banks are independent on this platform.
- Blocking algorithm ONLY takes advantage of data locality to improve memory bandwidth.

Effects of the blocking size



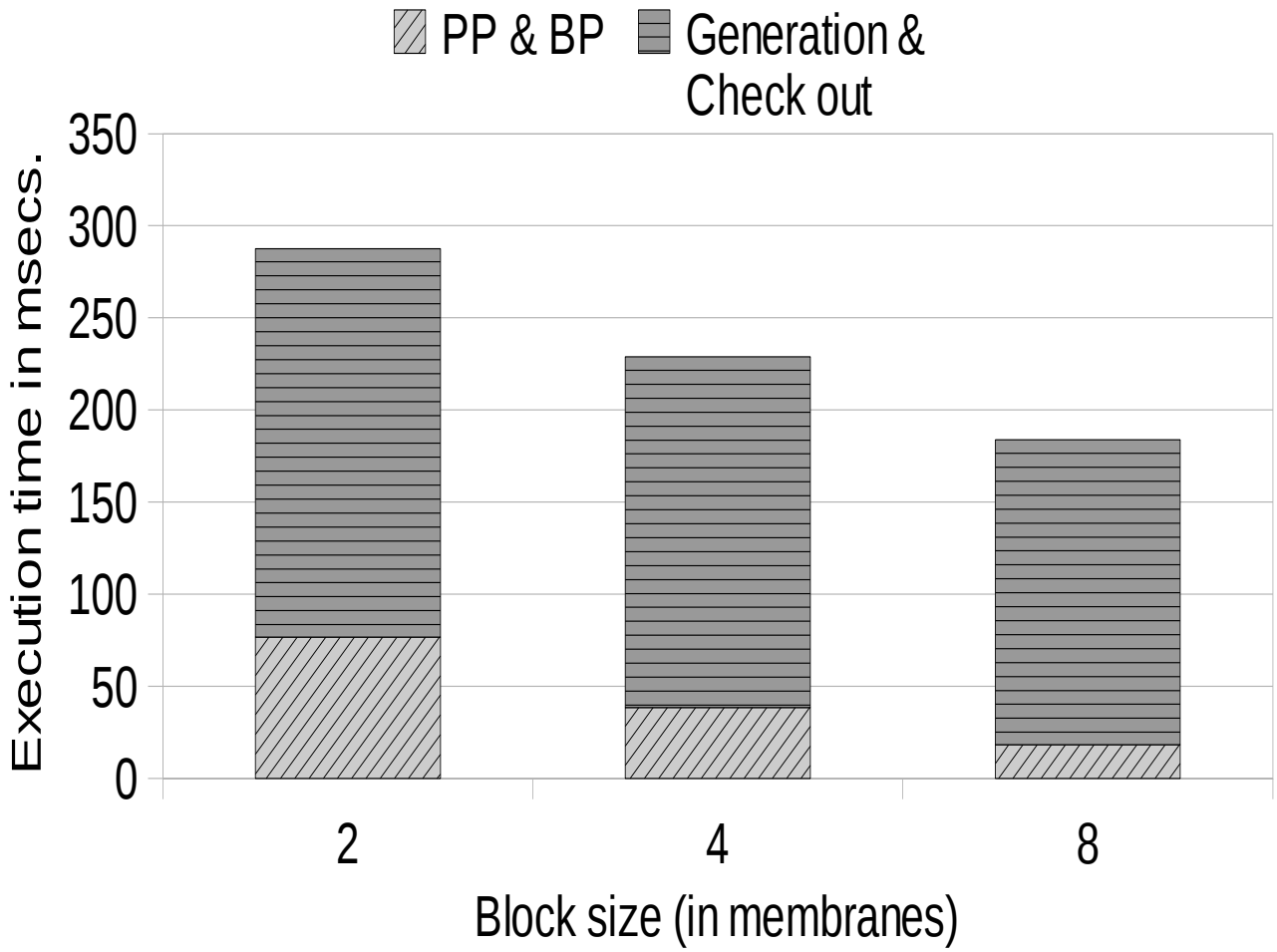
Execution time depending on the problem size



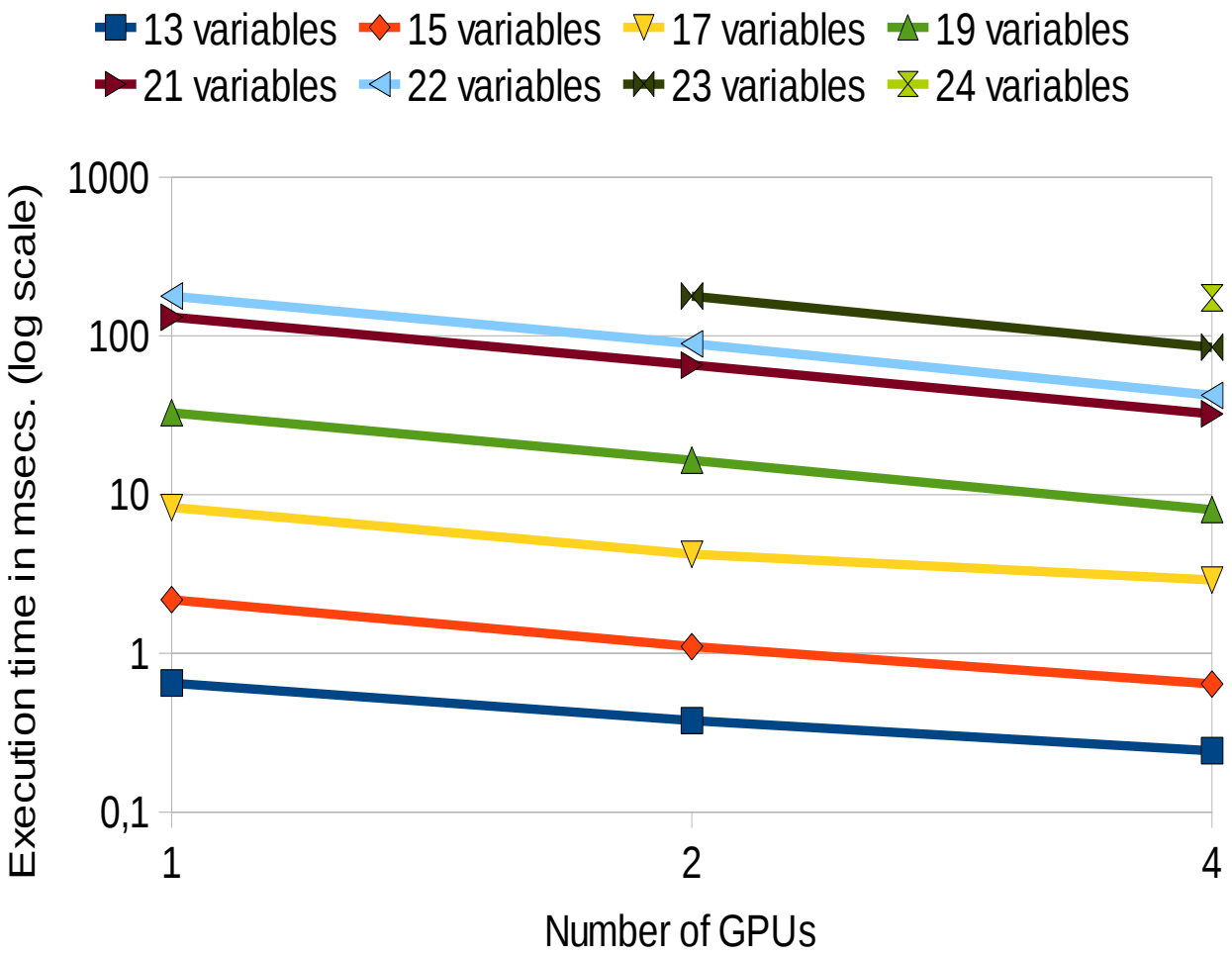
Blocking Vs non-Blocking

- The tiling alternative obtains up to 1.75x speed up versus non-tiling version.

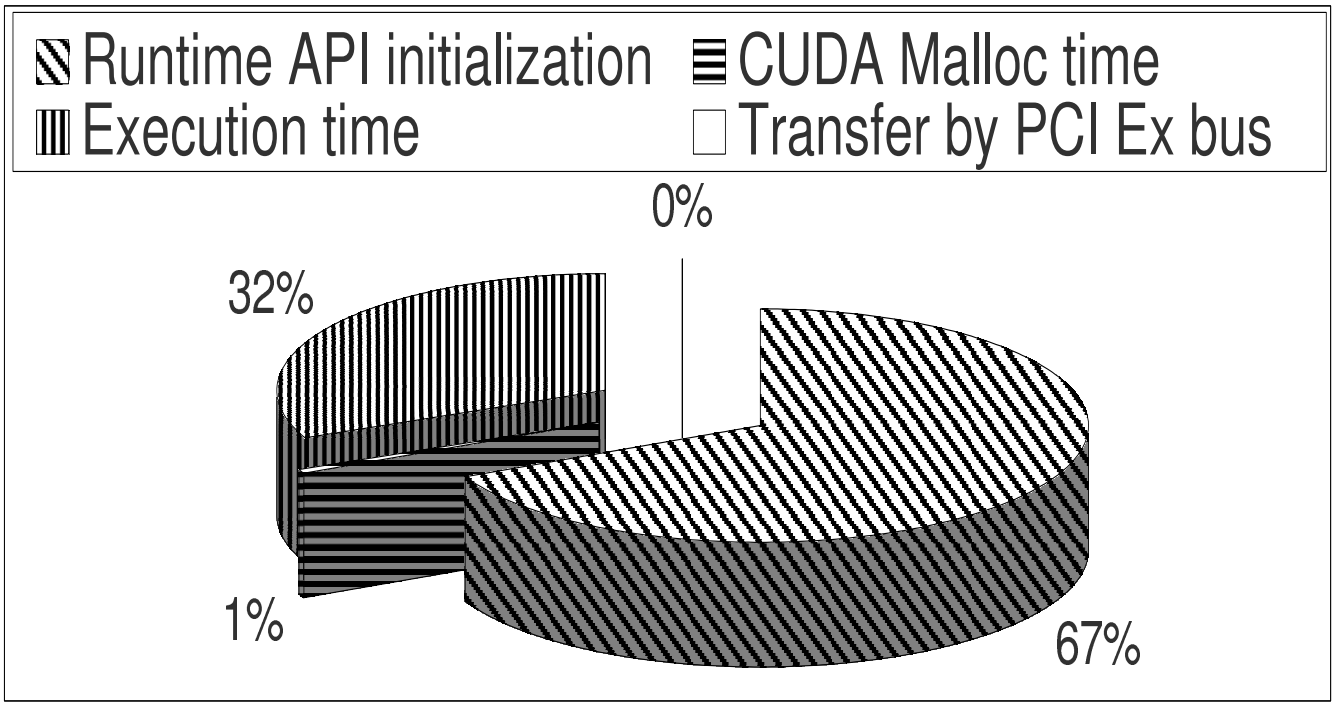
Effects of the blocking size



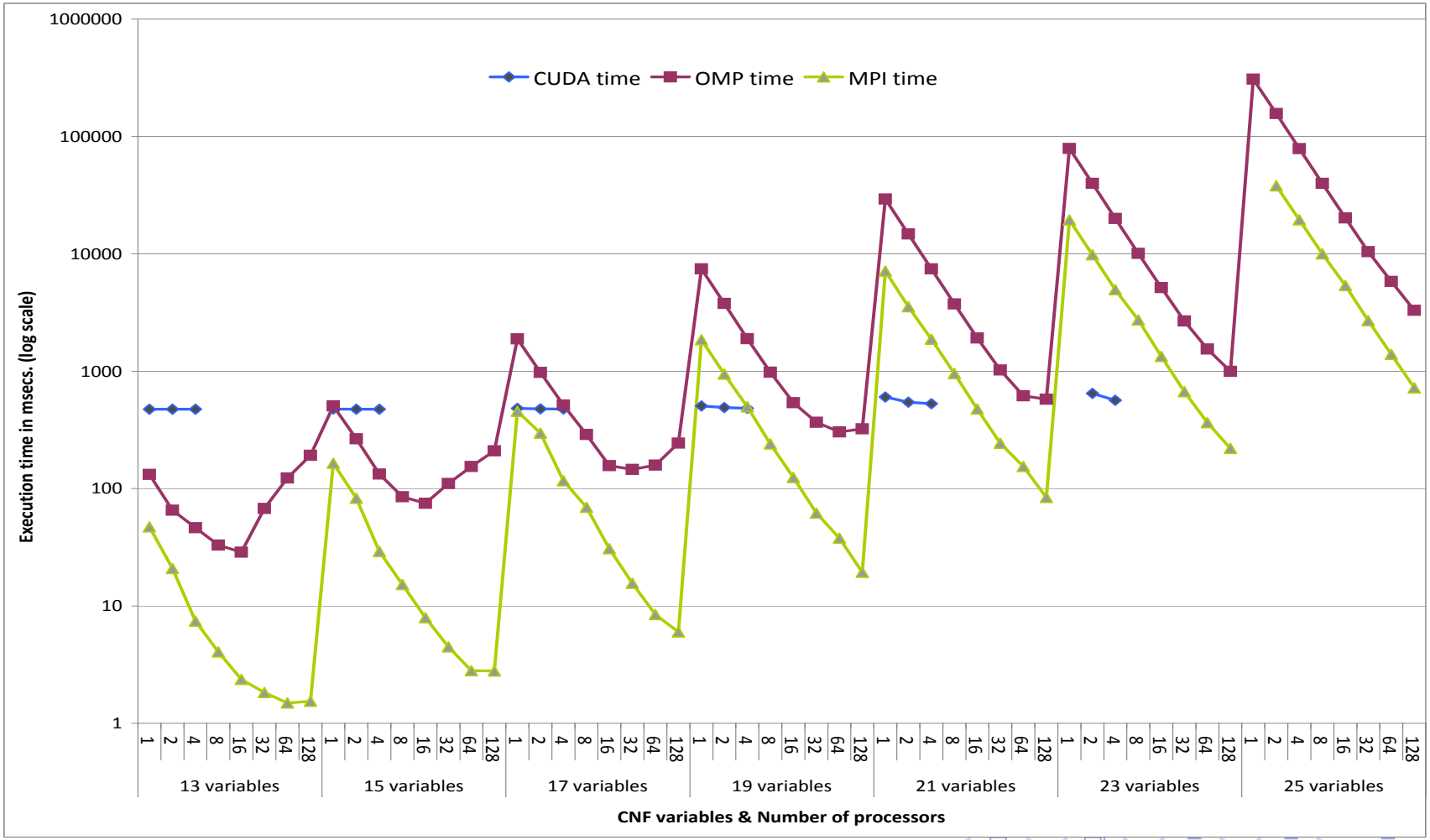
Execution time depending on the problem size



GPUs associated overheads



Overall comparison



Outline

Conclusions

- Simulation of a recognizer P systems with active membranes for the SAT.
- Three parallel architectures: Shared Memory, Distributed Memory and GPUs.
- Blocking increases the bandwidth in all targeted systems (data locality).
- OpenMP sim is the lowest performance, increasing overheads along with the number of processors.
- OpenMP sim is the only one able to execute all our benchmarks due to the memory requirements.
- MPI sim exhibits good scalability with the number of processors.
- GPU sim is the best platform in terms of execution time, reaching up to 10x speed up versus MPI and 40x speed up versus OpenMP.

Future work

- Newest generation of GPUs, such as Fermi, increase performance and flexibility.
- Cloud computing and Heterogeneous computing increase memory space without sacrifying performance.
- Other P system models for modeling ecosystems can be simulated on HPC solution.

Thanks for your attention

Questions?

Bibliography for this presentation


J. M. Cecilia, J. M. García, G. D. Guerrero, M. A. M. del Amor, I. Pérez-Hurtado and M. J. Pérez-Jiménez. Simulating a P system based efficient solution to SAT by using GPUs. *Journal of Logic and Algebraic Programming*, 79(6):317–325, 2010.


J. M. Cecilia, J. M. García, G. D. Guerrero, M. A. M. del Amor, I. Pérez-Hurtado and M. J. Pérez-Jiménez. Simulation of P systems with active membranes on CUDA. *Briefings in Bioinformatics*, 11(3):313–322, 2010.

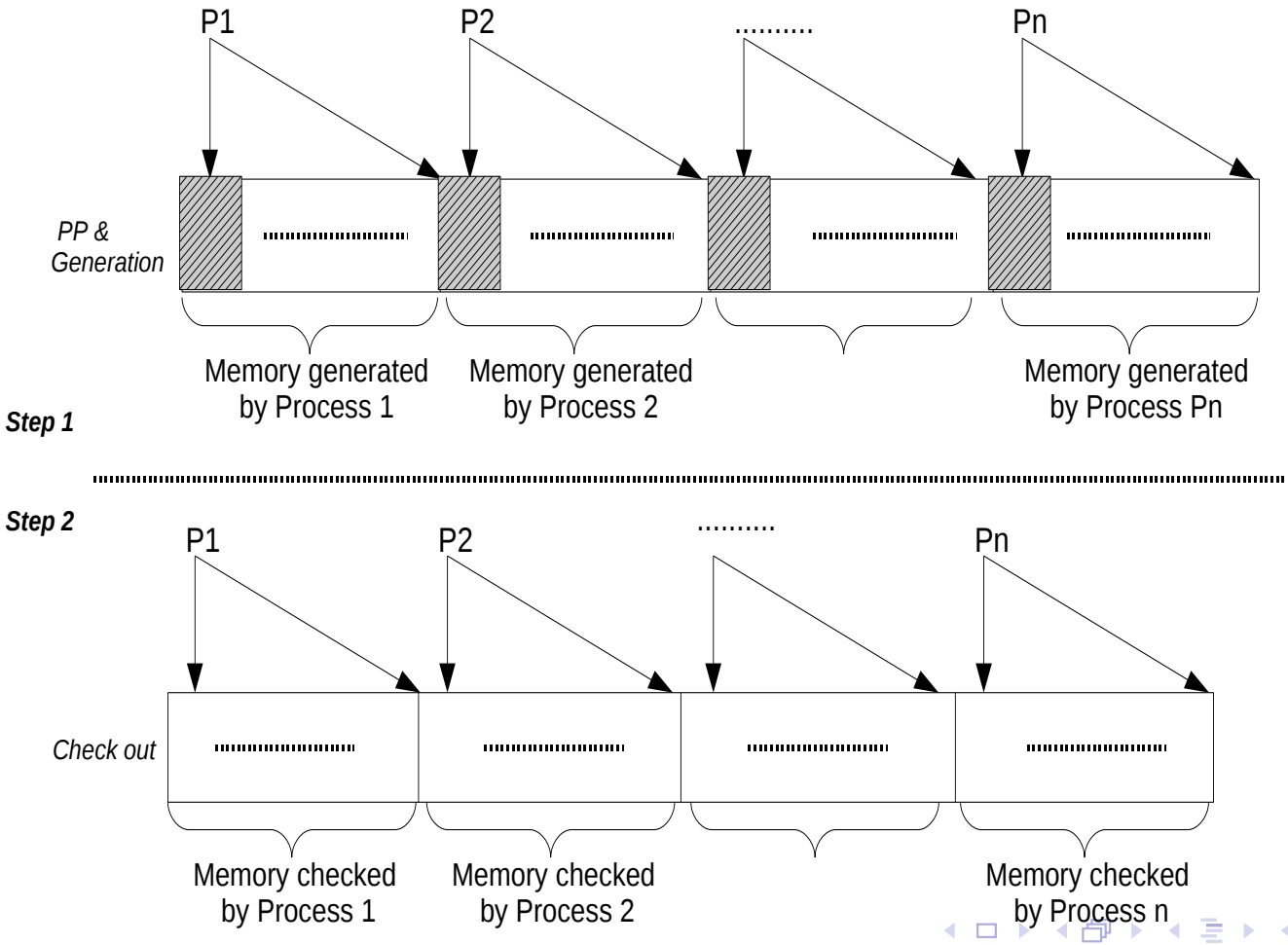

S. A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM.


M. J. Pérez-Jiménez, Á. Romero-Jiménez and F. Sancho-Caparrini. Complexity classes in models of cellular computing with membranes. *J. Natural Computing*, 2(3):265–285, 2003.

Outline

General data policy description

Non-block based implementation



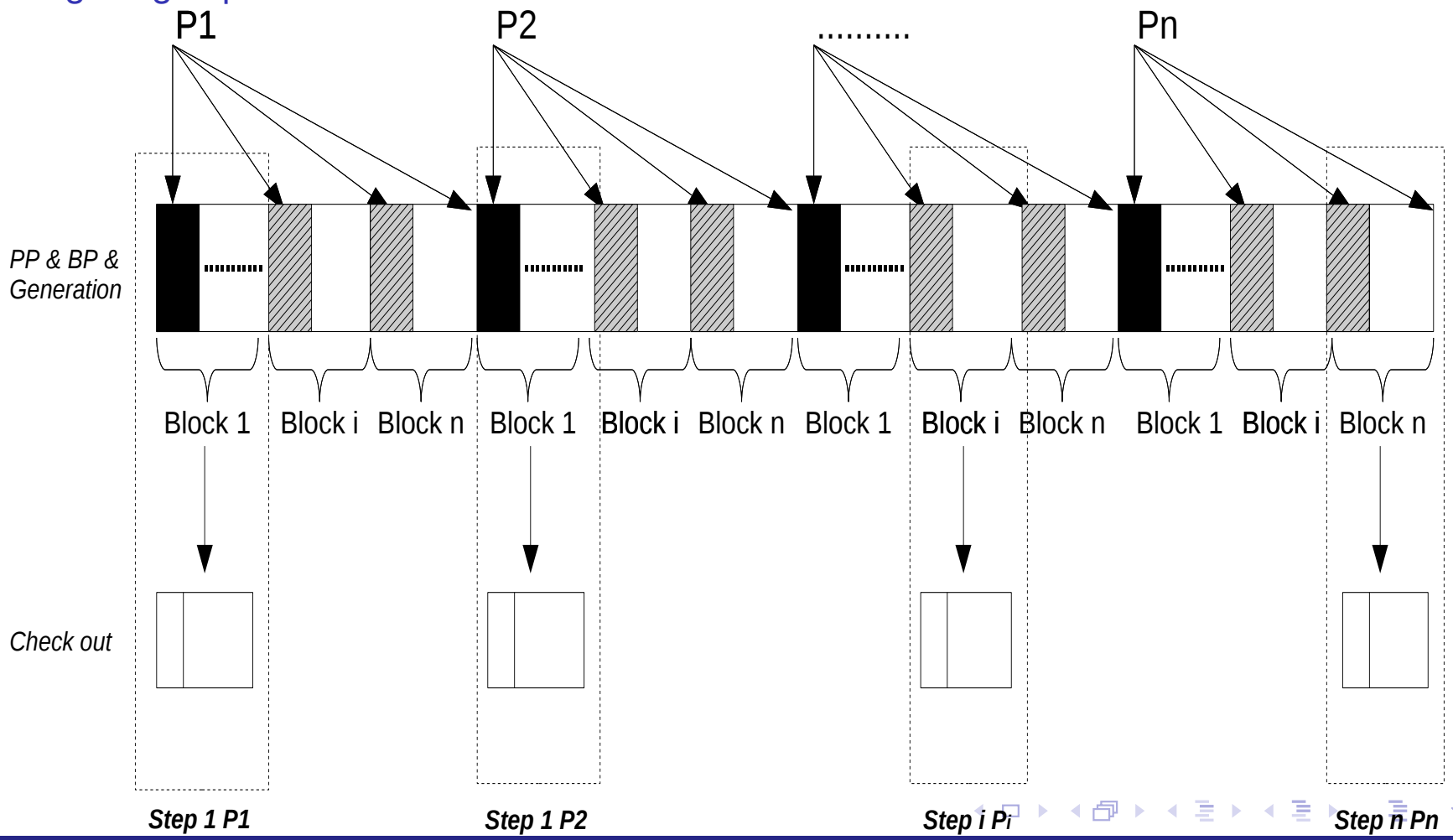
General data policy description

Reduced bandwidth

- Many cache misses (first writes and then reads).
- Block-based data layout to improve locality.
- Intel Vtune profiler to see memory behaviour, and Cuda Visual profiler for GPUs.
- Needs efficent handling of the exponential workspace.

General data policy description

A blocking/tiling implementation



Extra bibliography

V. Nguyen, D. Kearney and G. Gioiosa. An extensible, maintainable and elegant approach to hardware source code generation in reconfig-p. *J. Logic and Algebraic Programming*, 79(6):383–396, 2010.

G. Paun. Membrane computing. An introduction. *Springer-Verlag*, pages 9–419, 2002.