On-the-Fly Equivalence Checking using Distributed Local Resolution of Boolean Equation Systems

Christophe Joubert

(joint work with Radu Mateescu and Nicolas Descoubes)

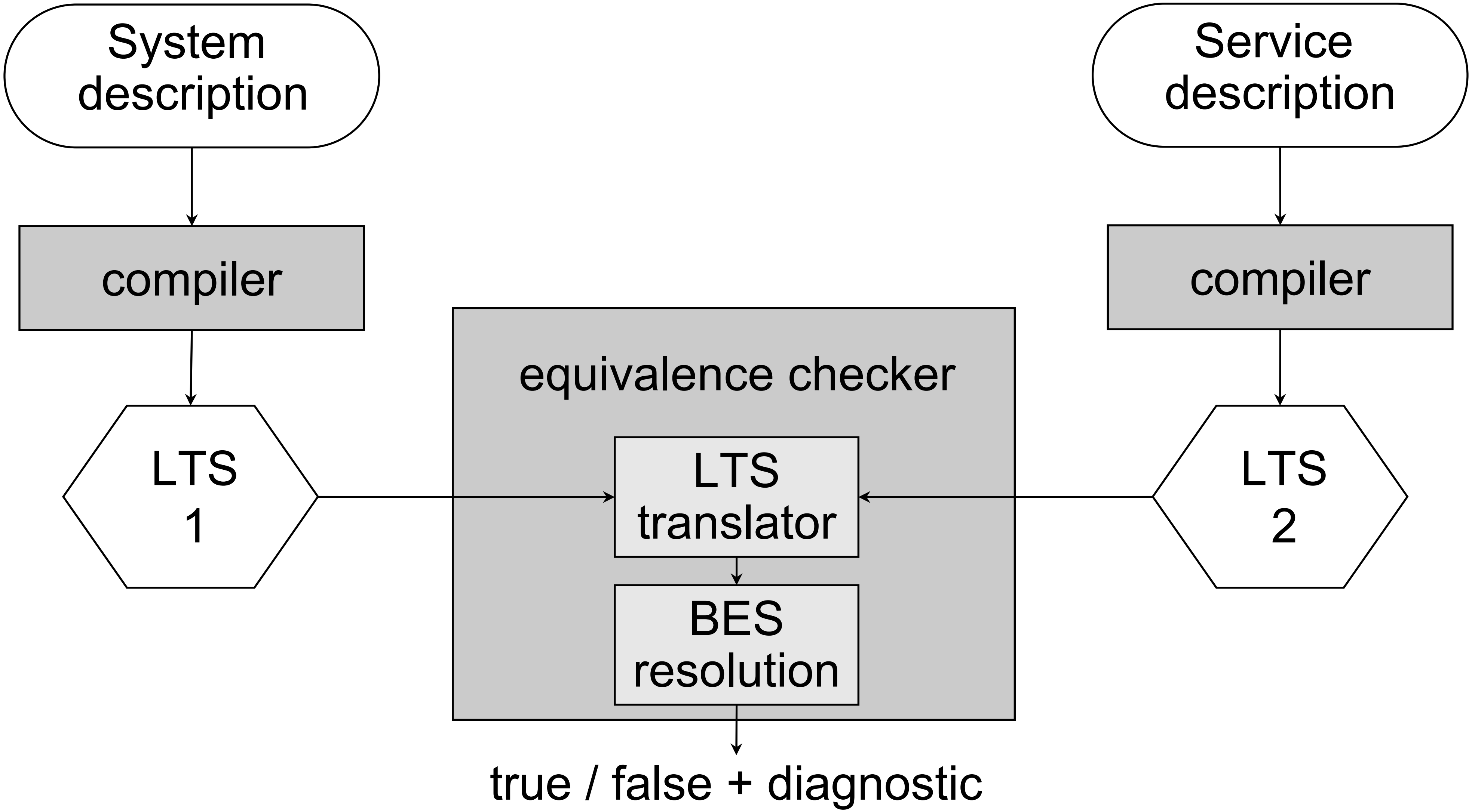
INRIA / VASY

http://www.inrialpes.fr/vasy

Outline

- Introduction
- Distributed local resolution of BES
- Implementation and experiments
- Conclusion and future work

Equivalence checking using BES resolution



Equivalence relation in terms of BES

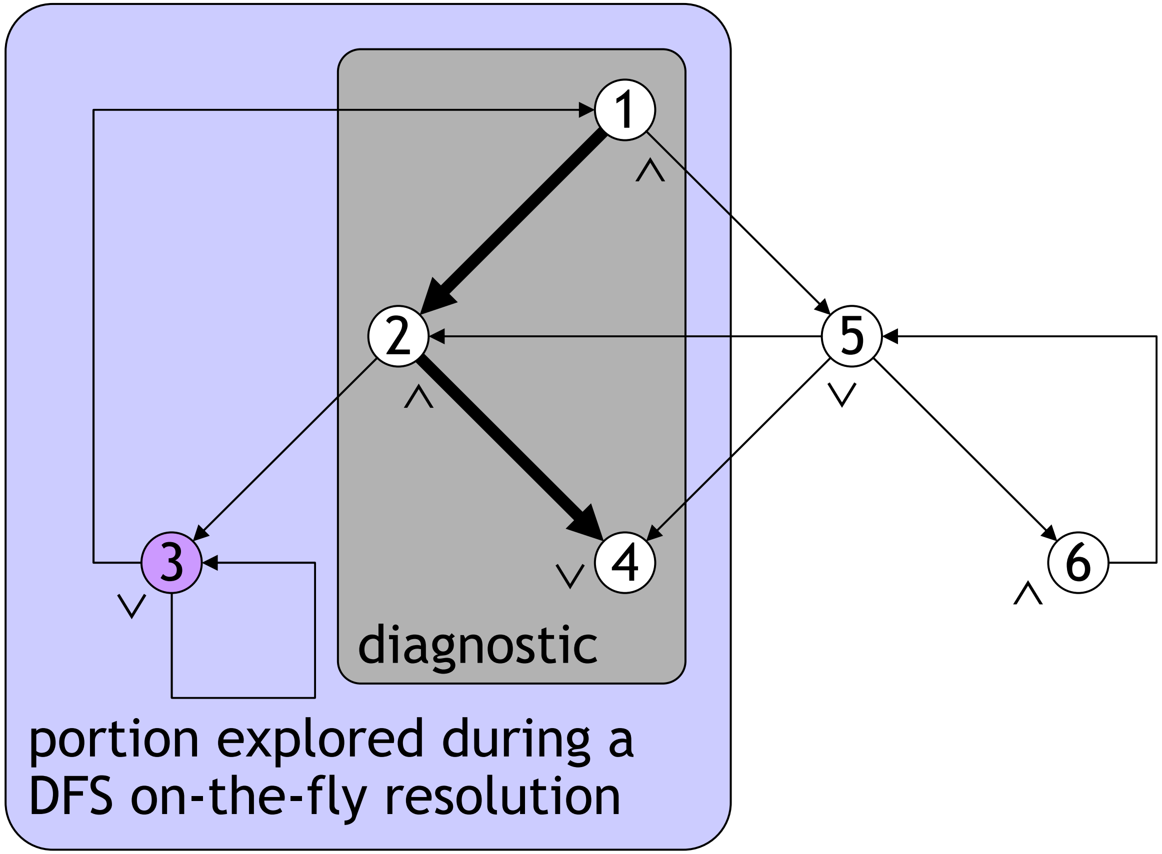
Relation	Encoding
Strong	$X_{p,q} =_\nu (\wedge_{p \to_a p'} \vee_{q \to_a q'} X_{p',q'}) \wedge (\wedge_{q \to_a q'} \vee_{p \to_a p'} X_{p',q'})$
Observational	$X_{p,q} =_\nu (\wedge_{p \to_\tau p'} \vee_{q \to_{\tau^*} q'} X_{p',q'}) \wedge (\wedge_{p \to_a p'} \vee_{q \to_{\tau^*.a.\tau^*} q'} X_{p',q'})$ $\wedge (\wedge_{q \to_\tau q'} \vee_{p \to_{\tau^*} p'} X_{p',q'}) \wedge (\wedge_{q \to_a q'} \vee_{p \to_{\tau^*.a.\tau^*} p'} X_{p',q'})$
Tau*.a	$X_{p,q} =_\nu (\wedge_{p \to_{\tau^*.a} p'} \vee_{q \to_{\tau^*.a} q'} X_{p',q'}) \wedge (\wedge_{q \to_{\tau^*.a} q'} \vee_{p \to_{\tau^*.a} p'} X_{p',q'})$
Safety	$X_{p,q} =_\nu Y_{p,q} \wedge Y_{q,p}$ $Y_{p,q} =_\nu (\wedge_{p \to_{\tau^*.a} p'} \vee_{q \to_{\tau^*.a} q'} Y_{p',q'})$

Boolean graphs (running example)

BES

$$
\begin{cases}
x_1 =_\nu x_2 \wedge x_5 \\
x_2 =_\nu x_3 \wedge x_4 \\
x_3 =_\nu x_1 \vee x_3 \\
x_4 =_\nu F \\
x_5 =_\nu x_2 \vee x_4 \vee x_6 \\
x_6 =_\nu x_5
\end{cases}
$$

boolean graph



Distributed local BES resolution algorithm

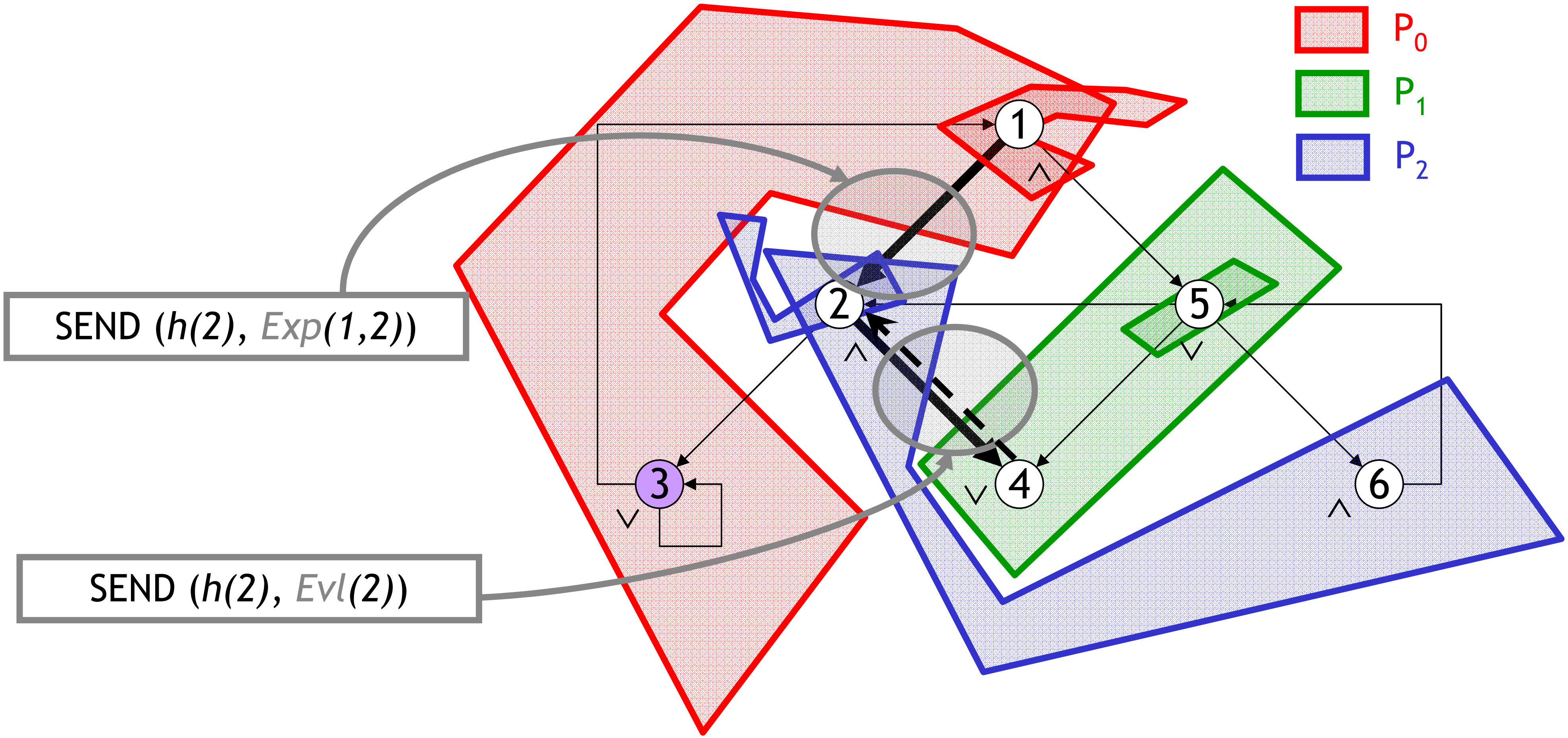
Distributed environment

- P computers (with own CPU and memory)
  - NOWs and clusters of PCs
- Strongly connected network topology
- P processes performing the distributed BES resolution (SPMD model) + 1 *coordinator* process (configuration, launching, collection of statistical data, termination detection)

Distributed algorithm

- DSOLVE (x, (V,E,L), h, i) => Bool
  - Inputs:
    - Variable of interest *x*
    - Implicit boolean graph *(V,E,L)* (successor function)
    - Static hash function *h*
    - Index of current node *i* (i $\in$ [0, P-1]
  - Principle:
    - BFS forward exploration of boolean graph (V,E,L) starting at $x \in V$
    - Backward propagation of stable (computed) variables
    - Distribution (communication) of remote data
    - Termination when *x* is stable or the entire boolean graph has been explored
    - Diagnostic by keeping relevant successors
  - Ouput:
    - Boolean value of *x*

Distributed execution



Synchronization and communication

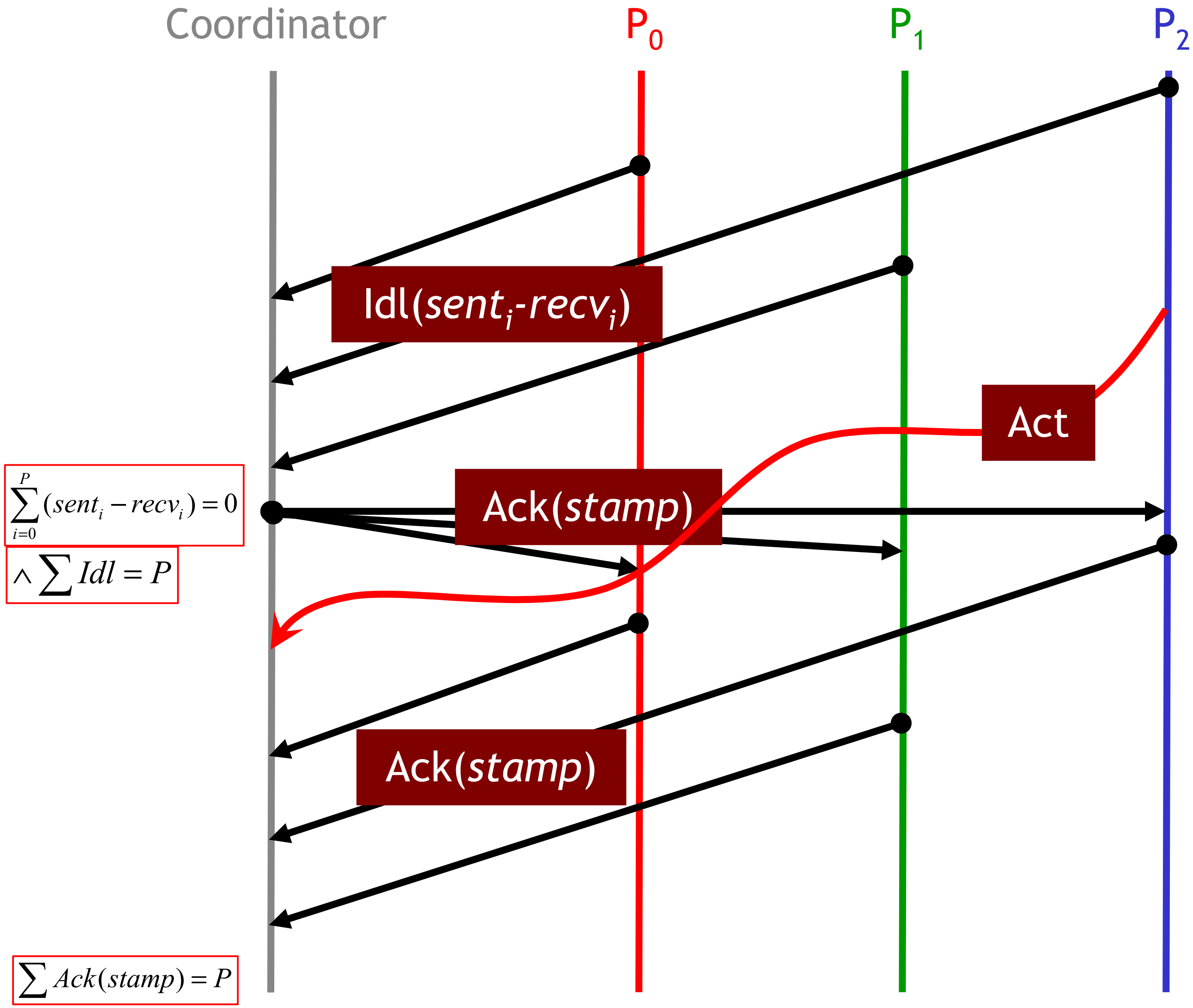
- Asynchronous (overlapping of communication with computations)

- Both blocking and non-blocking communication (avoiding synchronization and busy waiting)

- Fine tuned loosely coupled distributed communication library (CAESAR_NETWORK)
  - UNIX sockets with bounded buffers
  - TCP/IP protocol

=> Reducing memory consumption

Termination detection



Conditions of termination:

- Stabilized variable of interest *x*
- Boolean graph completely explored =
  - all local working sets of variables empty
  - No more messages transiting through the network ($\sum_{i=0}^{P}(sent_i - recv_i) = 0$)

=> 2 broadcast waves of global inactivity detection between the coordinator and the resolution processes

Complexity

- Theory of boolean graphs [Andersen-Vergauwen-95][Vergauwen-Lewi-94]
  - Worst case time complexity = O (|V|+|E|)
    - 2 intertwined graph traversals (forward and backward)
  - Worst case memory complexity = O (|V|+|E|)
    - Dependencies stored during graph exploration
  - Worst case message complexity = O (|E|)
    - 2 messages (expansion and stabilization) exchanged by edges
  - Distributed termination detection = O (|E|)
    - Practically, only 0.01% of total exchanged messages used for termination detection

Implementation and experiments

Parallel architecture

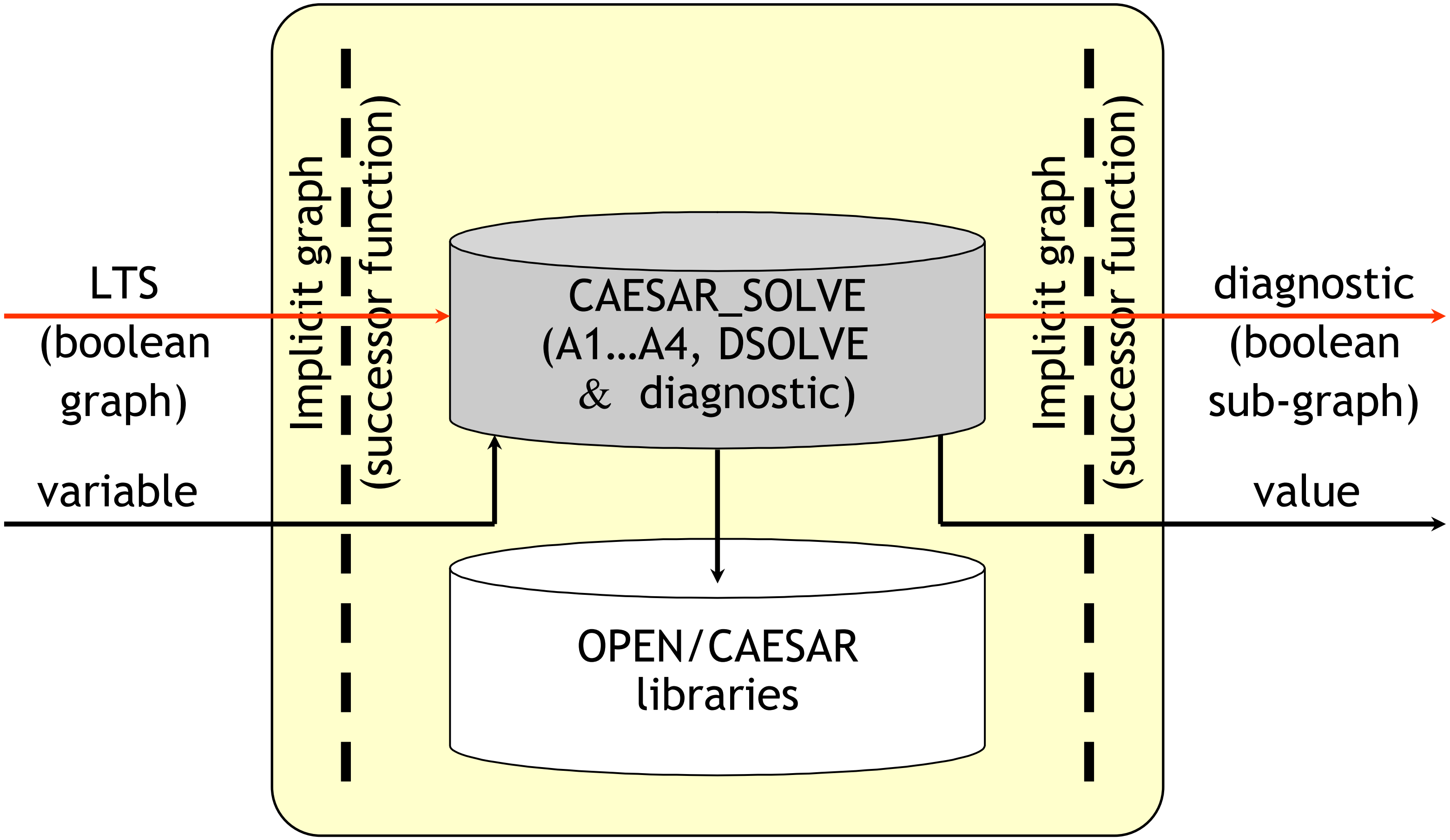
- 48 * Bi-Xeon 2.4 GHz + 1.5 GB of RAM + 80 GB
- 1 * switch 48 ports Gigabit
- 1 * switch 10 ports Gigabit
- Debian 2.4.26
- OAR batch scheduler
- http://idpot.imag.fr/

Software architecture

- Highly modular, allowing to separate:
  - The front-end (encoding of the equivalence relations as BESs), from
  - The back-end (BES resolution)

- DSOLVE :
  - 7500 lines of C code
  - Integrated to the BES resolution library CAESAR_SOLVE
  - Developed using the OPEN/CAESAR environment
  - Gives a immediate distributed version of BISIMULATOR which uses CAESAR_SOLVE as verification engine

CAESAR_SOLVE library



Random generation of BESs

- Small application (400 lines of C code)

- Successor function of a BES *(edges going out of a variable in the boolean graph)* characterized by a set of parameters:
  - % of variable kind alternation (i.e. proportion of $\wedge$ (resp. $\vee$) variables going out of a $\vee$ (resp. $\wedge$) variable)
  - % of boolean constants
  - Minimum number of variables
  - Average boolean equation length (branching factor of the boolean graph)
  - Random generation seed used for generating index and type of variables

- Function cost negligible w.r.t. distributed BES resolution

Speedup (Classes of BESs) - 1

- $S_P = T_s / T_P$,

P number of nodes,

T_s sequential execution time (CAESAR_SOLVE A2),

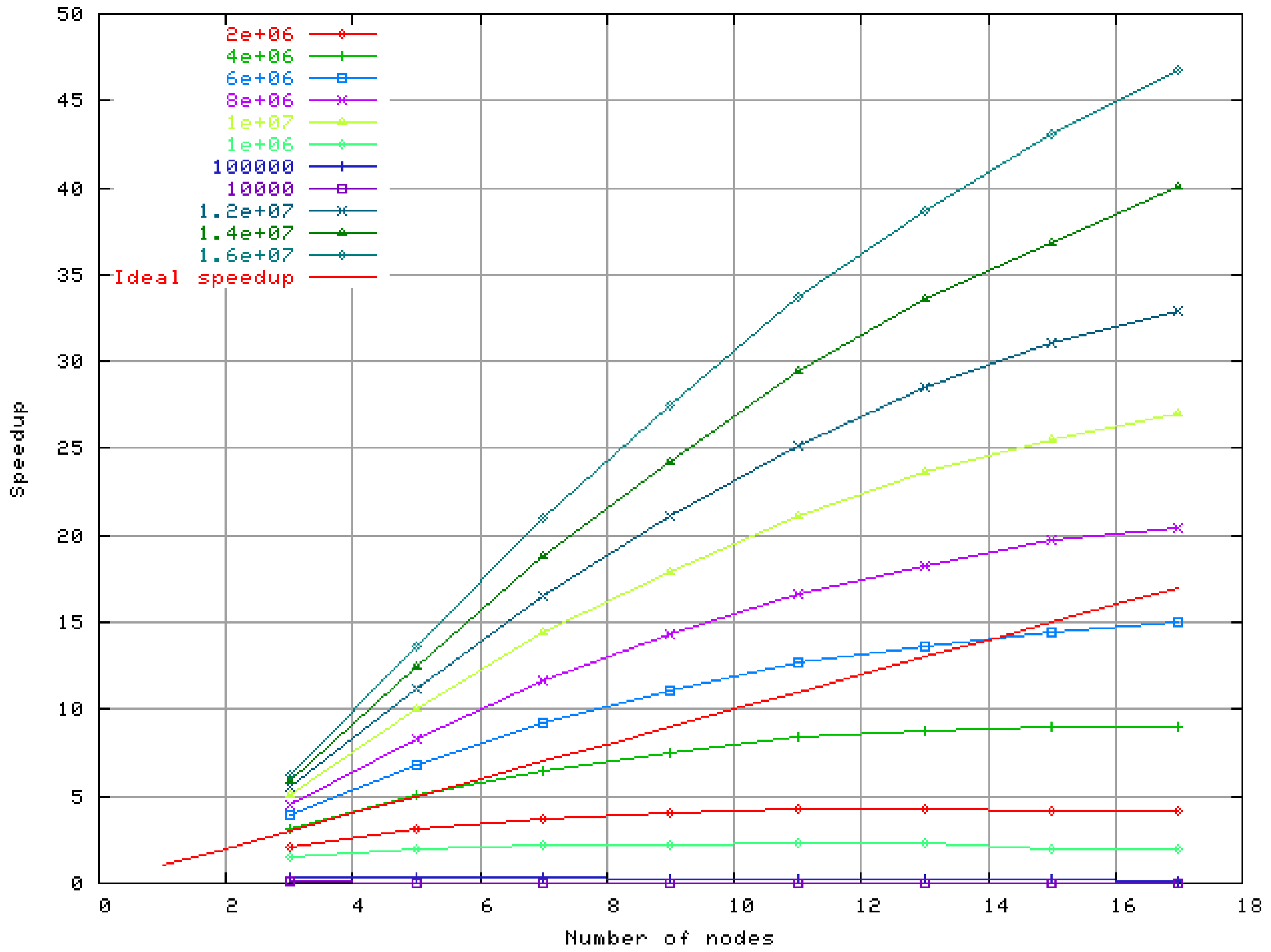
T_P parallel execution time,

Node = 1 machine (=1 cpu)

- 0% of variable kind alternation, 0% of boolean constants, boolean equations with 10 variables on average

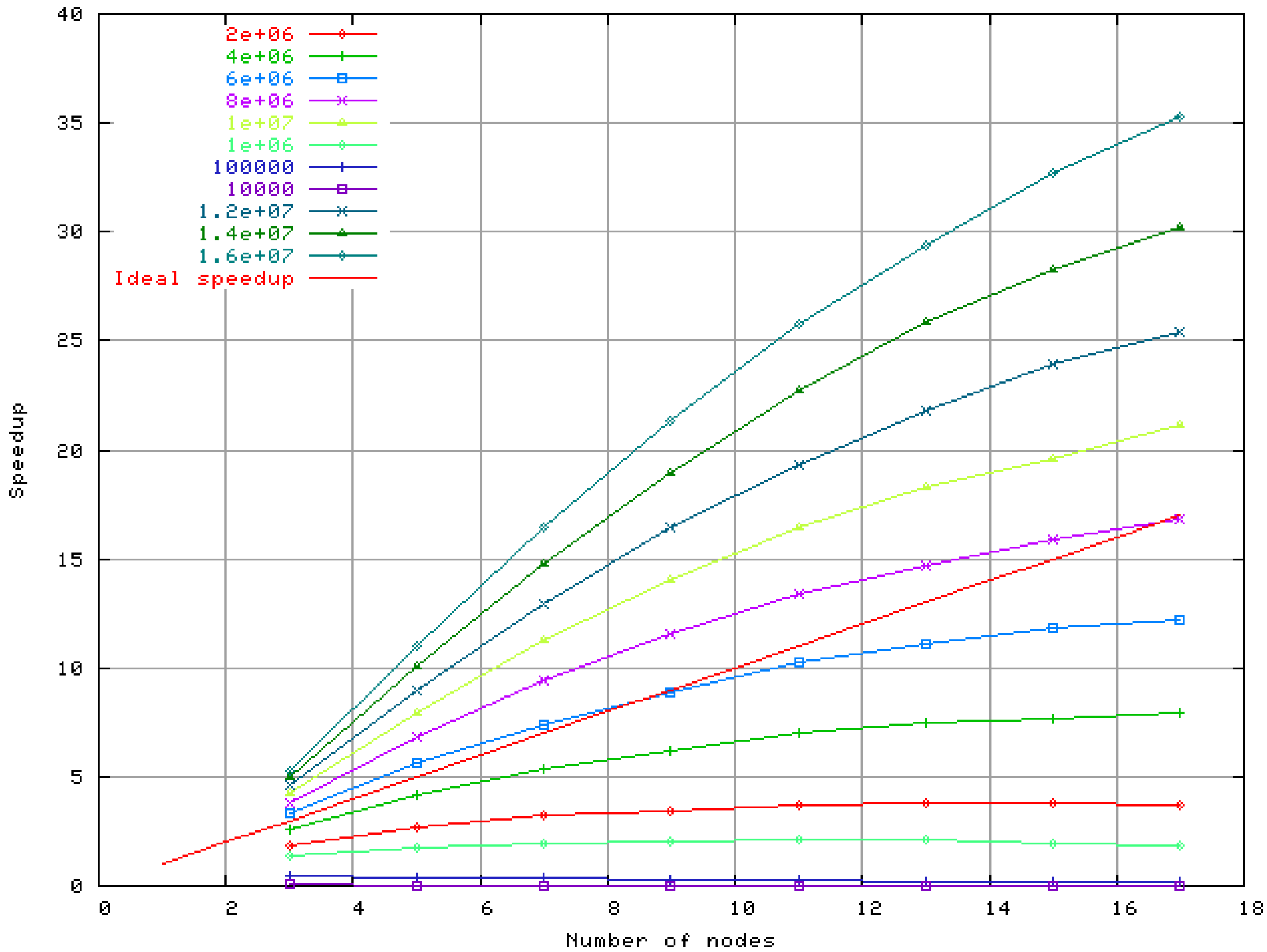
- Resolution = forward exploration of the boolean graph

- Superlinearity = cost of updating hash tables divided by P^2 in the distributed solution



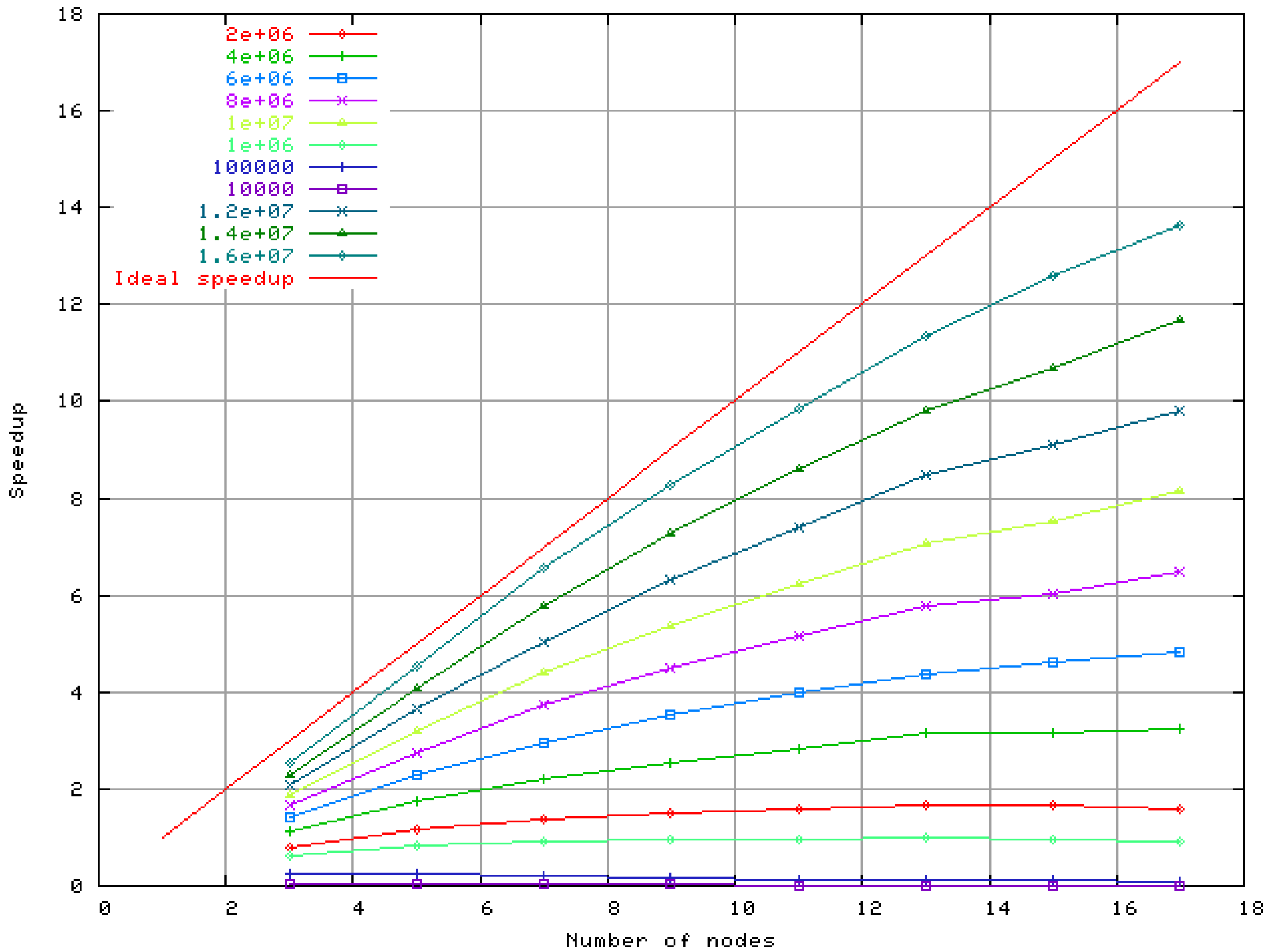
Speedup (Classes of BESs) - 2

- 100% of variable kind alternation, 10% of boolean constants
- ⟺ Verification of non-deterministic systems (equivalence checking and partial order reduction)
- Overall communication cost doubled due to stabilization messages
- Stabilization bounded to immediate predecessors (e.g. a ∨-variable stabilized to T will not necessarily stabilize its ∧-predecessors)



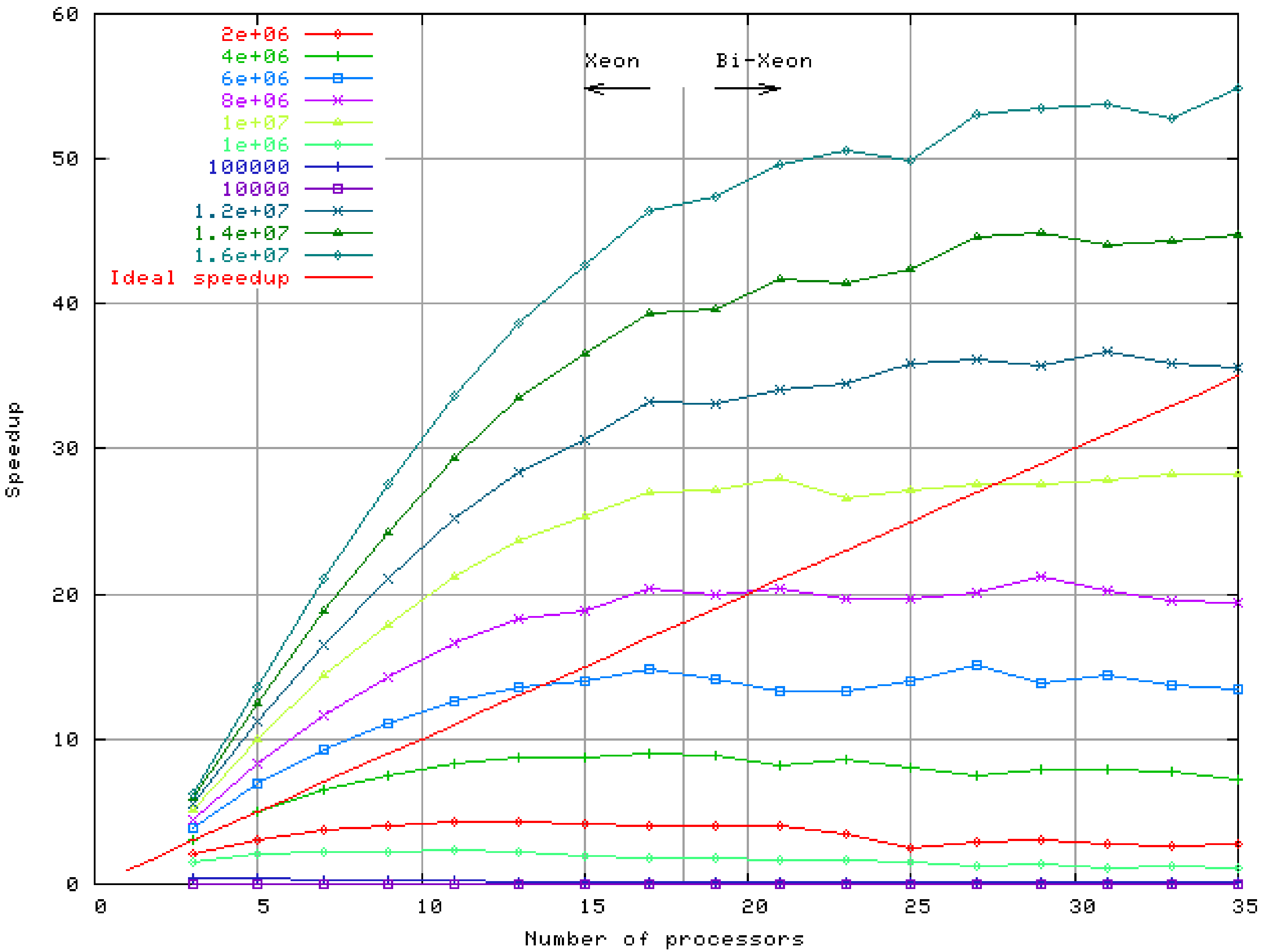
Speedup (Classes of BESs) - 3

- 2% of variable kind alternation, 1% of boolean constants
- $\Longleftrightarrow$ Equivalence checking of deterministic systems and model-checking
- Long paths of $\vee$-variable ended by T constants ($\wedge$-sinks)
- Better propagation mechanism in sequential algorithm (all information about predecessor dependencies stored locally)



Speedup (Classes of BESs) - 4

- 0% of variable kind alternation, 0% of boolean constants
- 1 processor/machine up to 17 processors
- 1 processor/machine and few 2 processors/machine from 19 to 35 processors
- Noise and irregularities on graph due to :
  - cluster maintenance
  - asymmetric hardware configuration (few nodes with 1 running cpu and others with 2 running cpus)

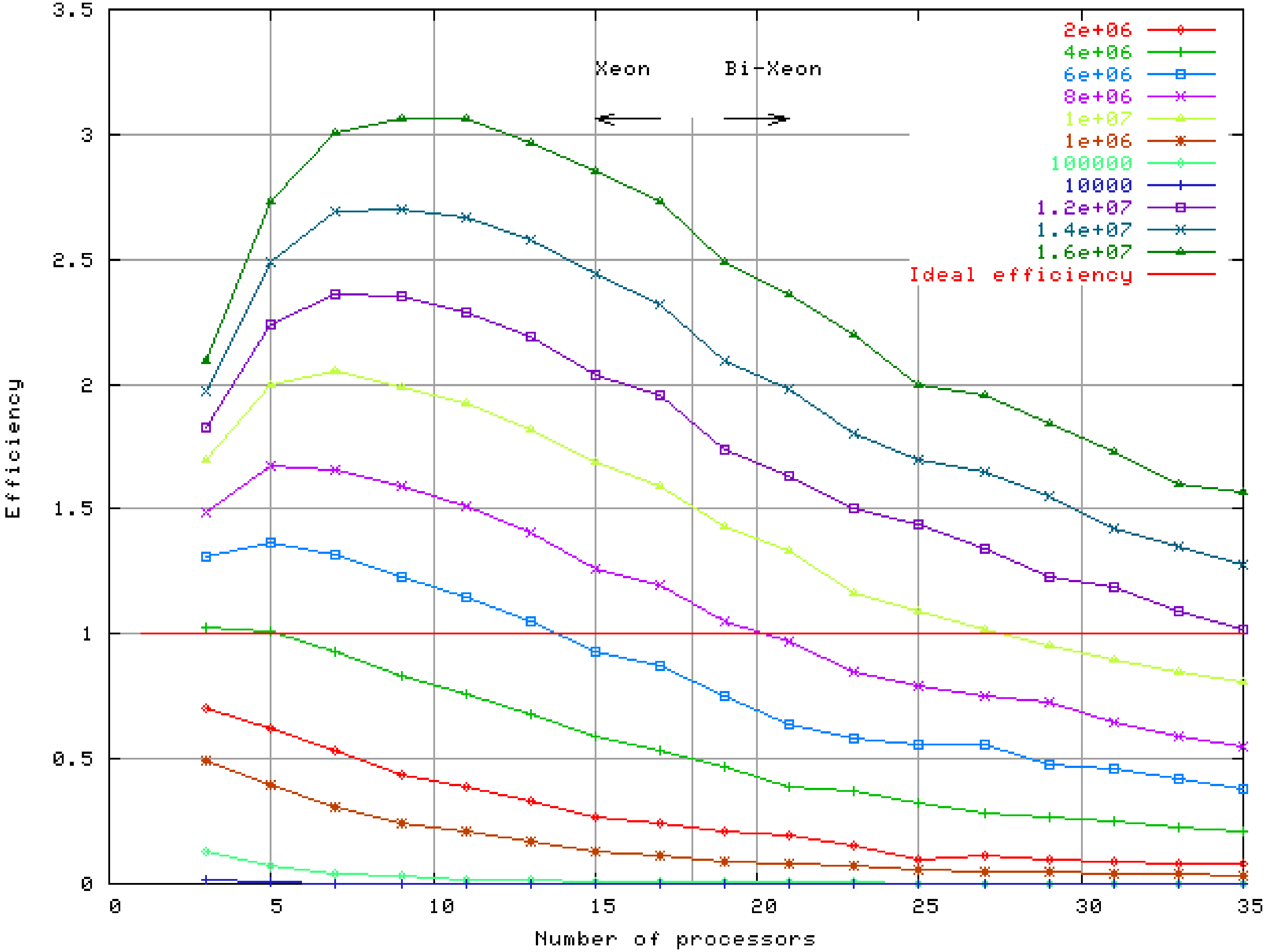


Efficiency (Classes of BESs) - 5

- $E_P = T_s / (T_P * P)$

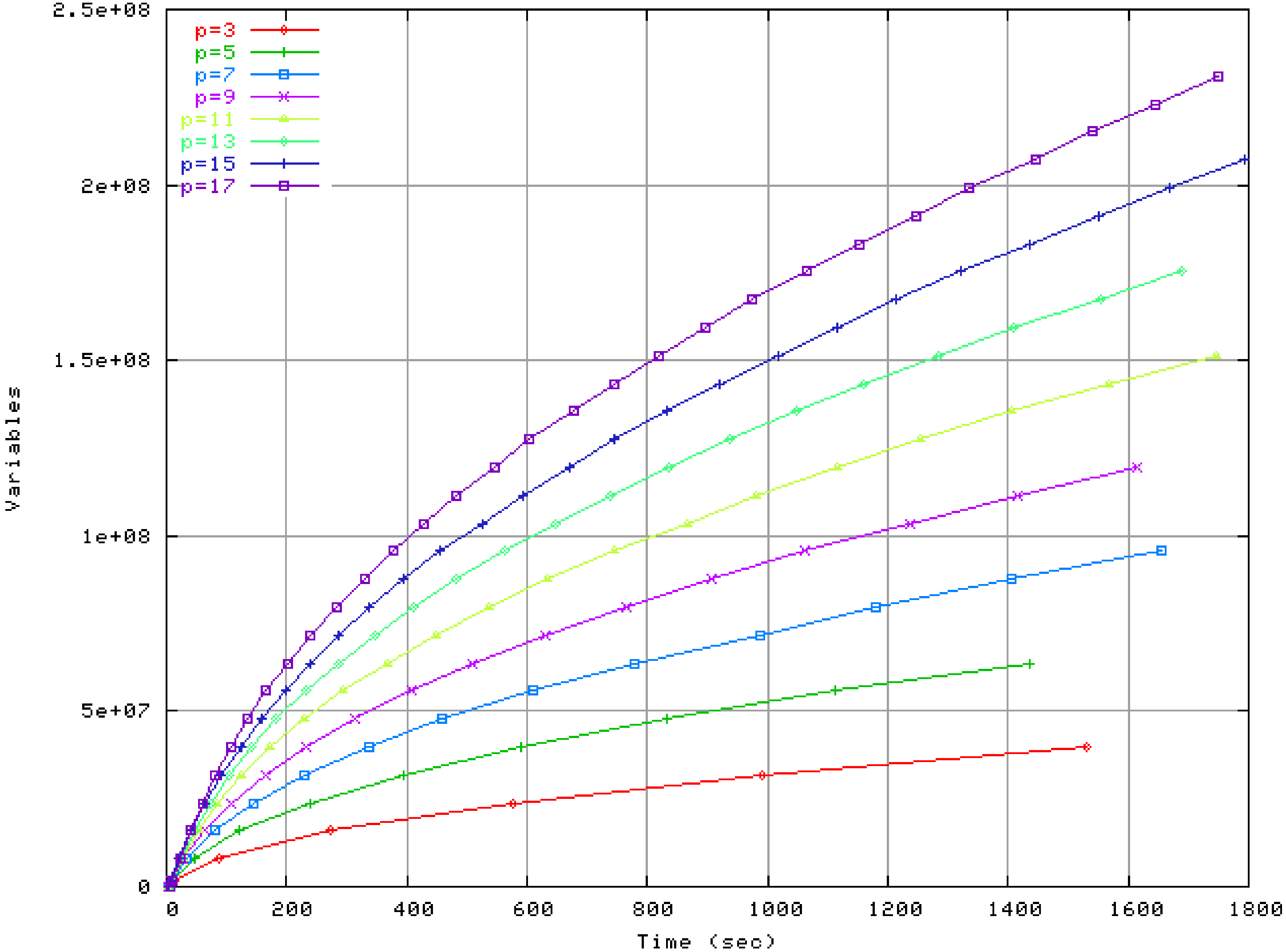
P, T_s, and T_P same as previous

- No particular decrease in efficiency when using bi-processors
- Irregularities due to the same reasons
- BESs size from $2*10^6$ to $1.6*10^7$ variables

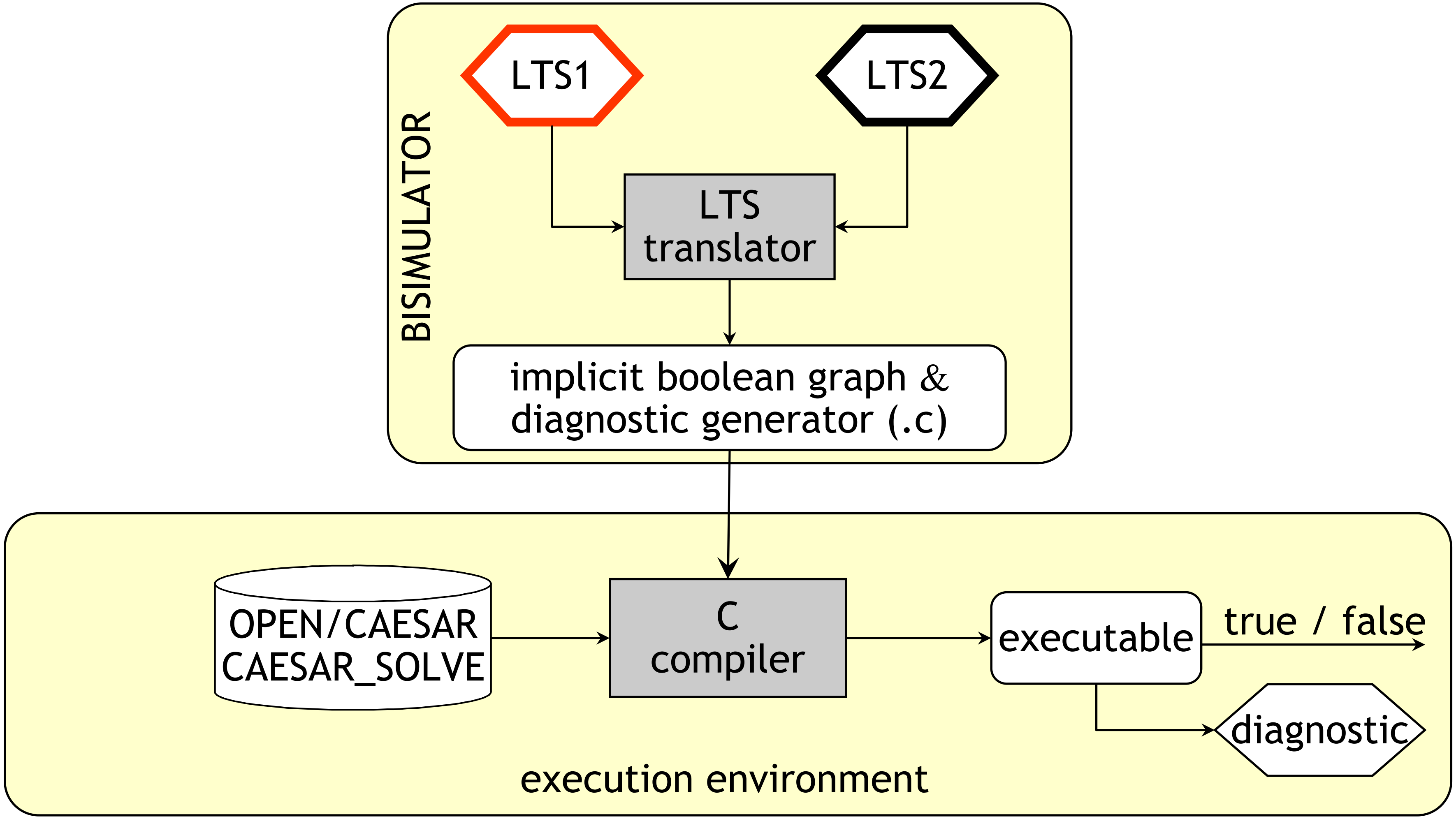


Scalability (Classes of BESs)

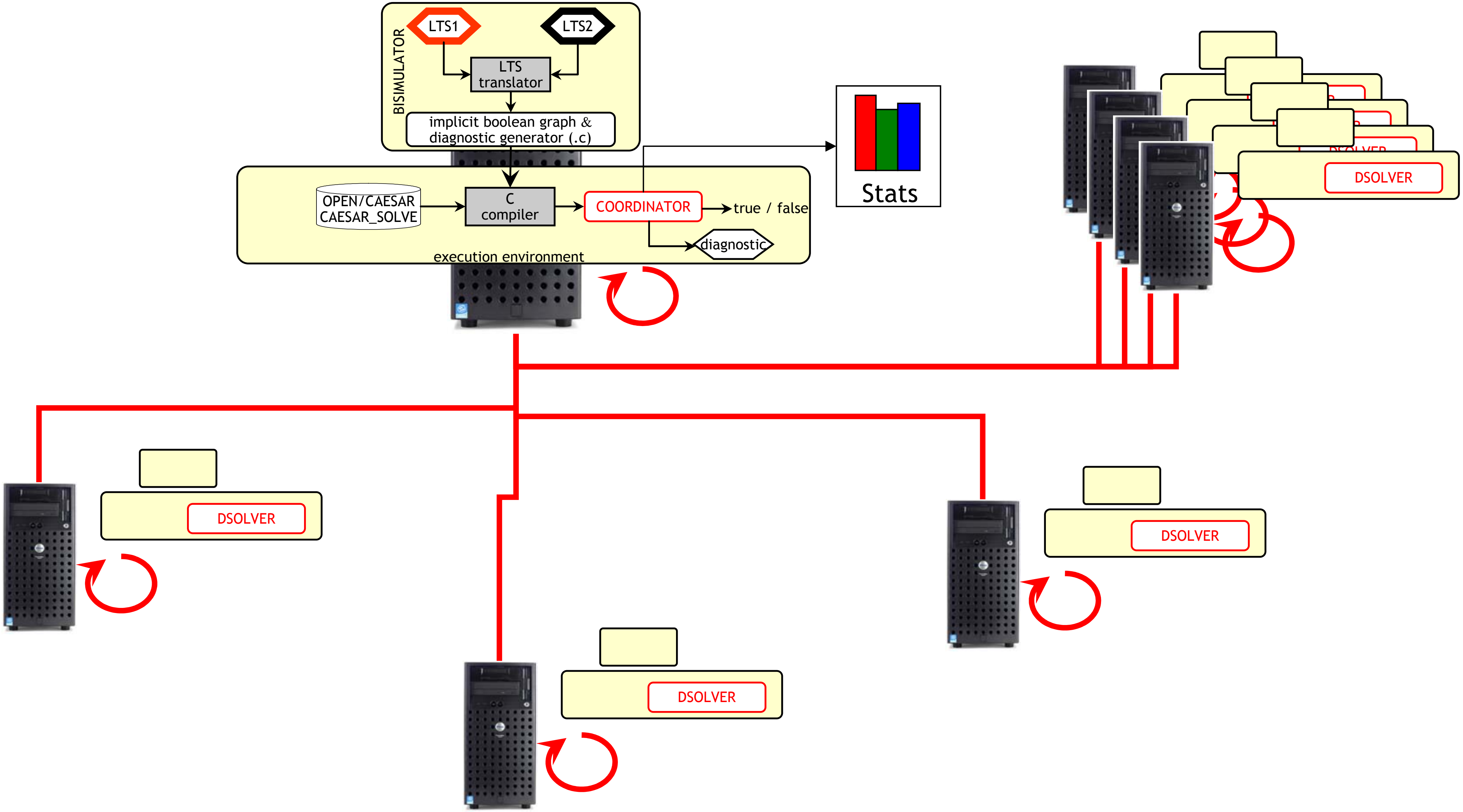
- Variation of processing speed (increasing the BES size on a fixed set of nodes)
- Execution time (increasing the number of nodes on a fixed BES size)
- 0% of variable kind alternation, 0% of boolean constants
- Curves shape close to linear ⟹ good scalability on increasing BES size (up to 2.5×10^8 variables !)



BISIMULATOR

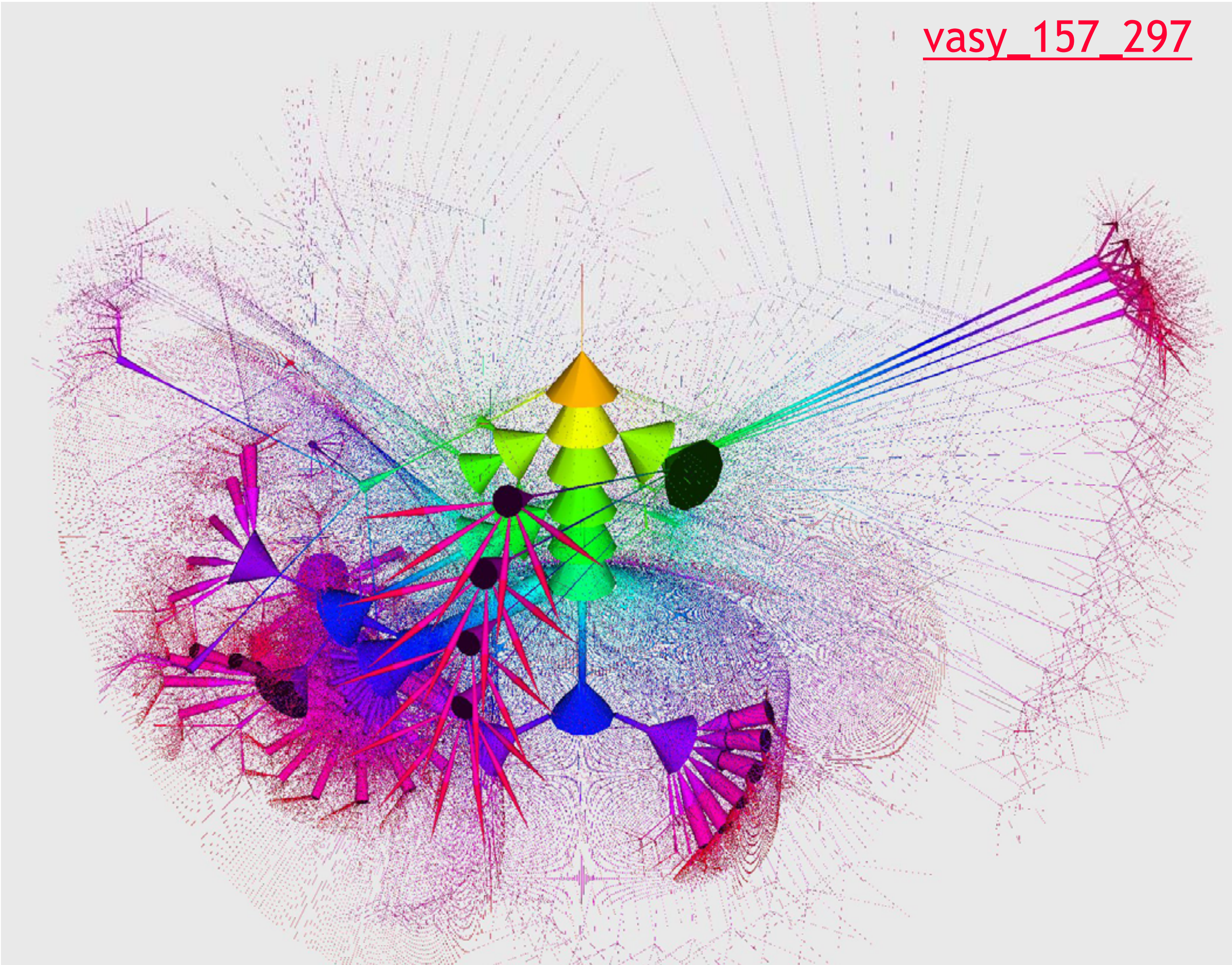


Distributed BISIMULATOR



The VLTS benchmark suite

- *Very Large Transition Systems* (VLTS)
  - *joint project of CWI/SEN2 and INRIA/VASY*
  - collection of *Labelled Transition Systems* (in BCG format)
  - case studies about the modelling of communication protocols and concurrent systems
  - 40 real life, industrial systems with up to 33,949,609 states, 165,318,222 transitions

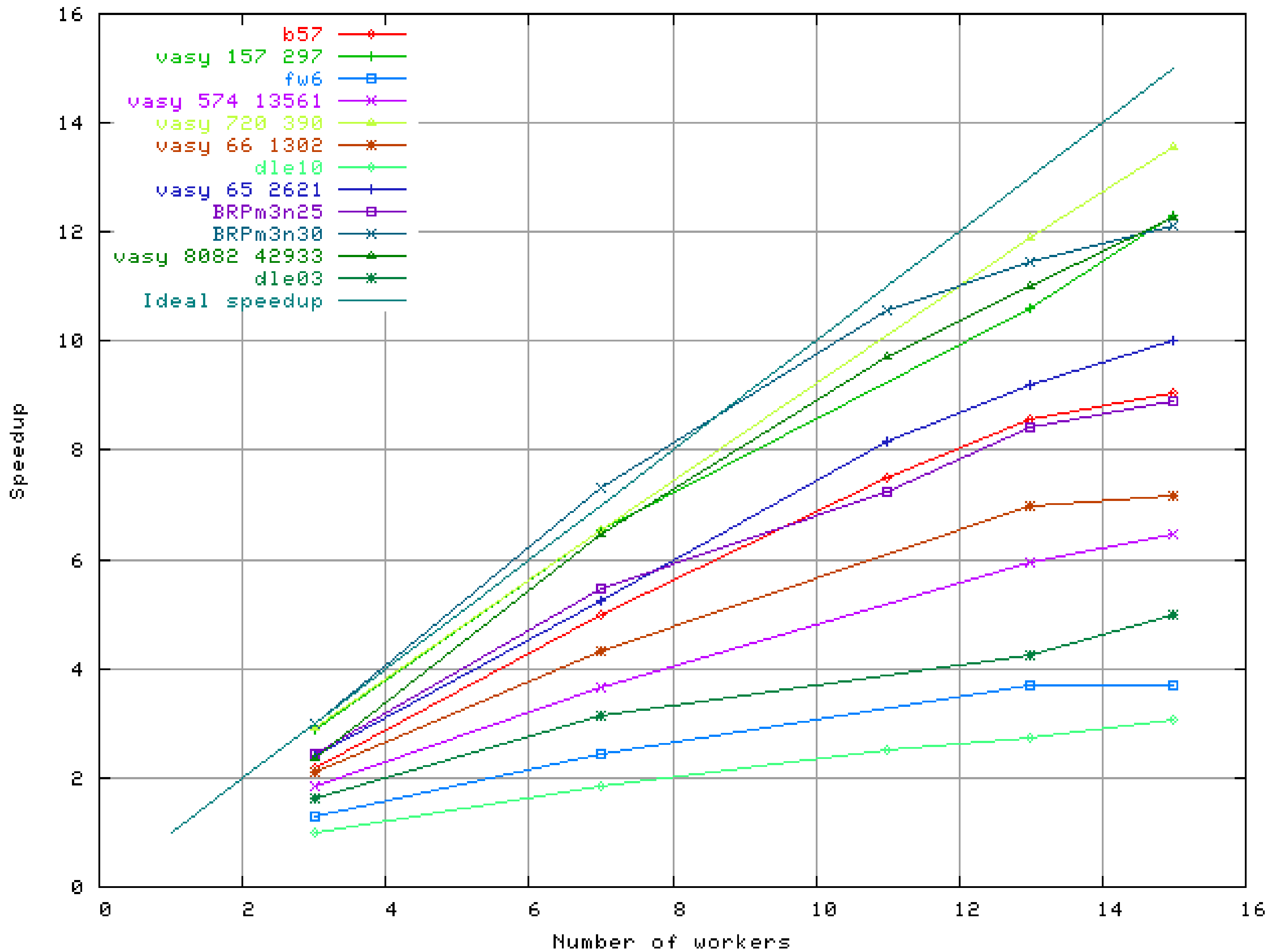


Pictures courtesy of Jan Friso Groote and Frank van Ham (Technical University of Eindhoven)

http://www.inrialpes.fr/vasy/cadp/resources/benchmark_bcg.html

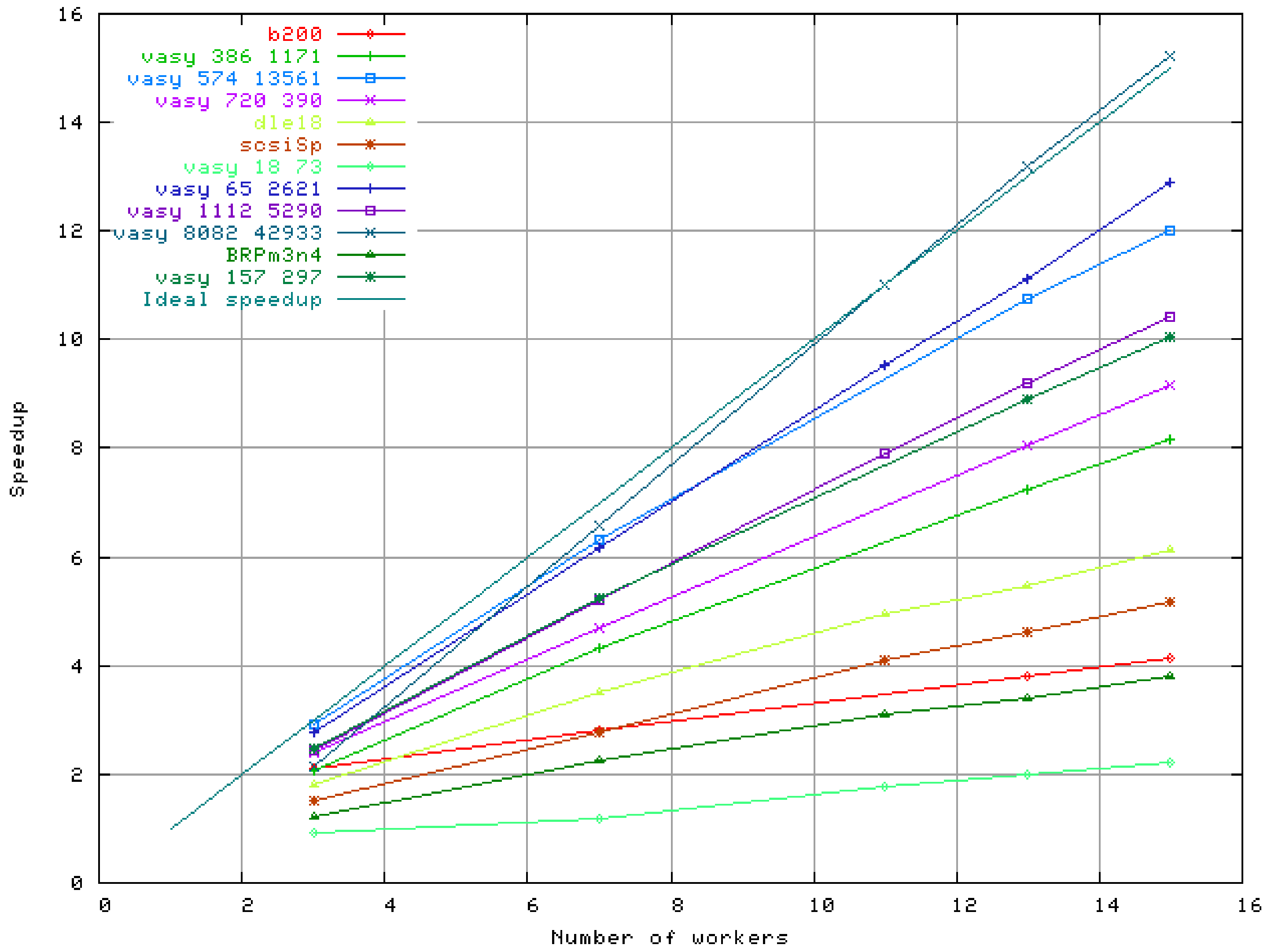
Speedup (Bisimulation) - 1

- 3 factors:
  - Size of LTSs
  - % of Tau transitions
  - Degree of non-determinism
- Strong equivalence
  - Best behavior (very few time spent in the front-end)
  - Linear speedups
  - BRPm3n30:
    - 332.53 s. in seq
    - 29.06 s. with 13 processors (speedup of 11.5)



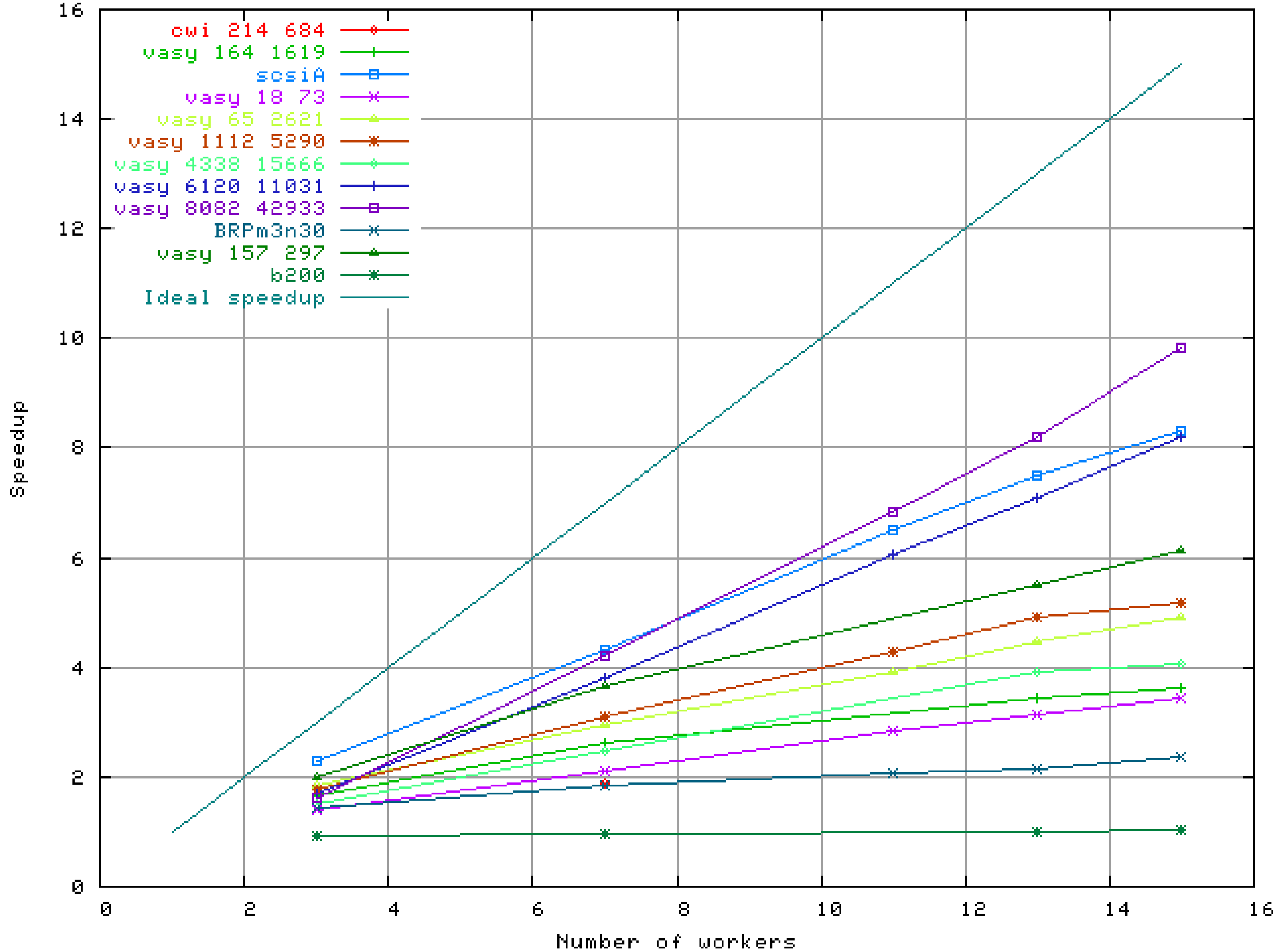
Speedup (Bisimulation) - 2

- Observational equivalence
  - Large BES encoding
  - Vasy_8082_42933:
    - Speedup of 10.99 with 11 processors
  - Branching equivalence not yet implemented but similar results expected

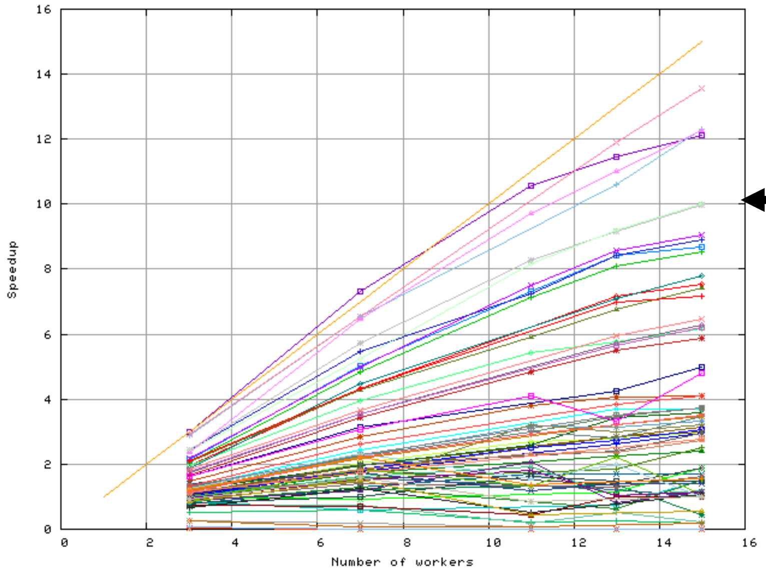


Speedup (Bisimulation) - 3

- Tau*.a equivalence
  - Similar results for safety equivalence
  - Worst behavior (extensive transitive closures on Tau transitions)
  - Very small BES encoding for high % of Tau transitions
  - Vasy_6120_11031:
    - Speedup of 8.22 with 13 processors

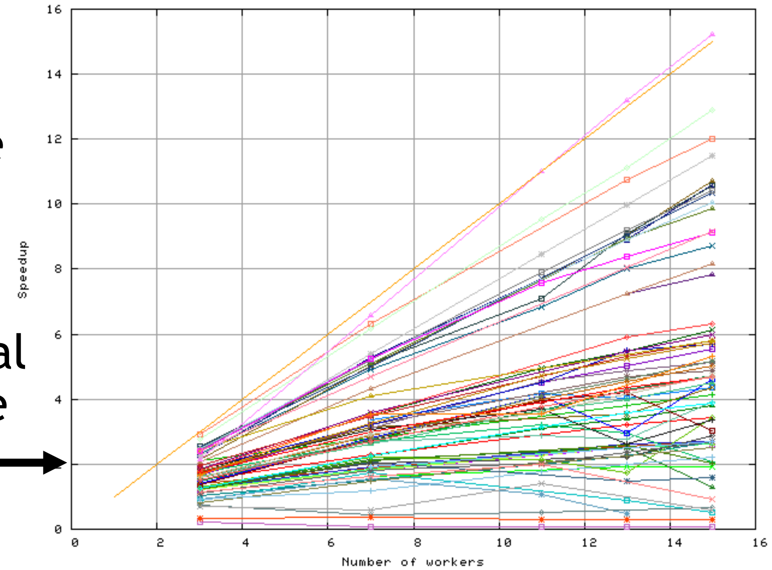


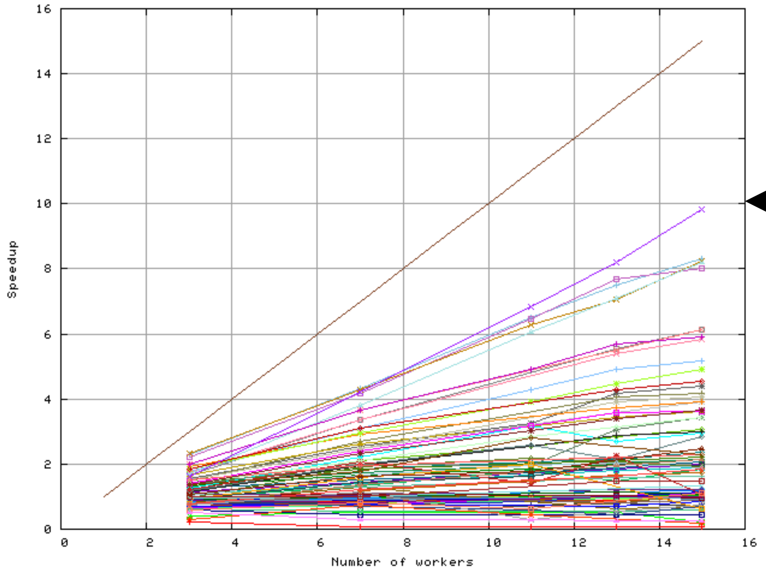
Speedup (VLTS Bisimulation) - 4



Strong equivalence

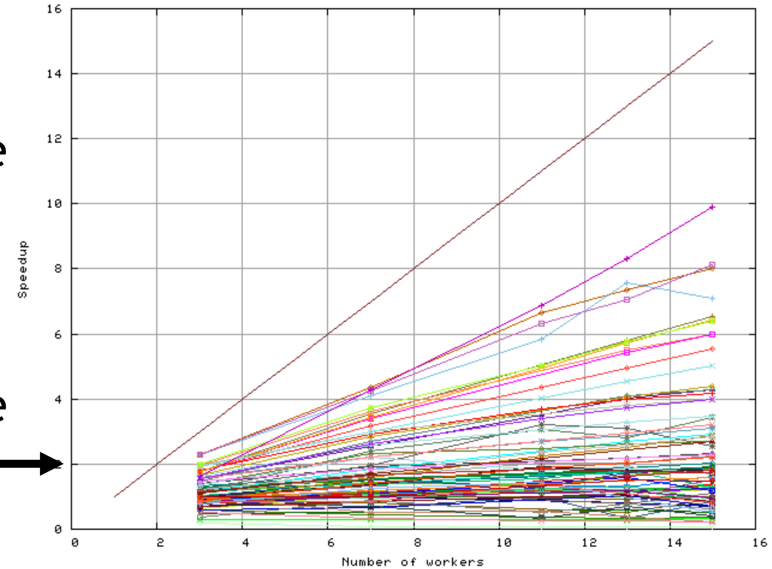
Observational equivalence





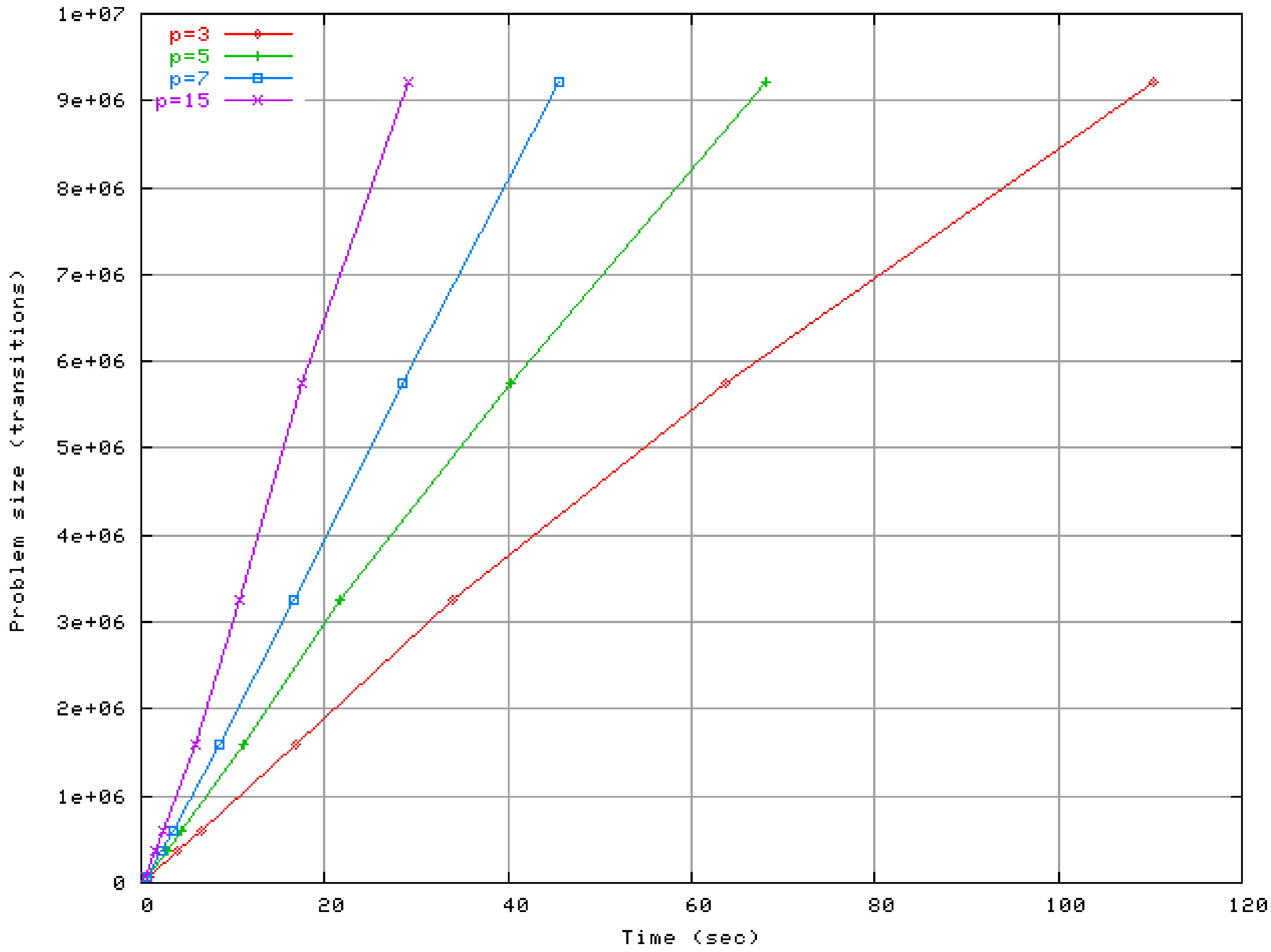
Safety equivalence

Taustar equivalence

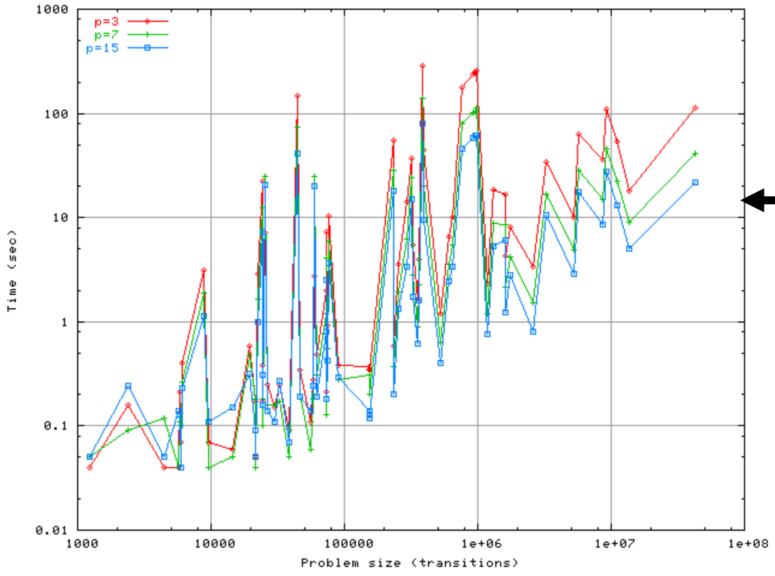


Scalability (Bisimulation) - 1

- BRPm3nK (K ∈ [4,30]):
  - Strong equivalence
  - Fixed p number of processors (p ∈ [3,15])
  - Adapted to increases in problem size
- B200:
  - 2.4 10^8 variables (max of 1.6 10^7 achieved in seq)
  - 24 minutes
  - 15 processors

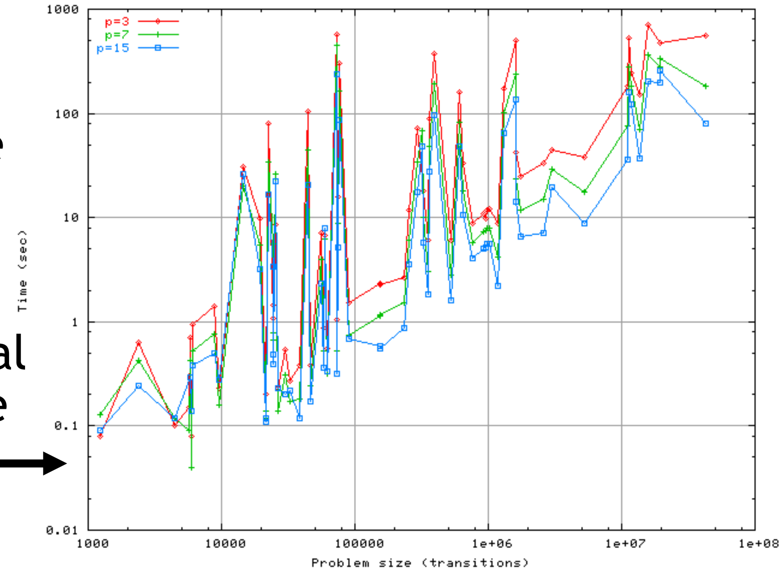


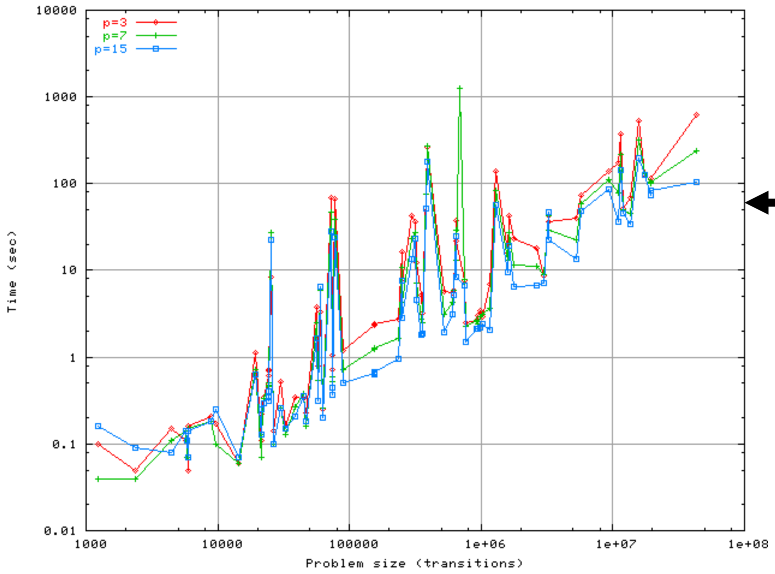
Scalability (Bisimulation) - 2



Strong equivalence

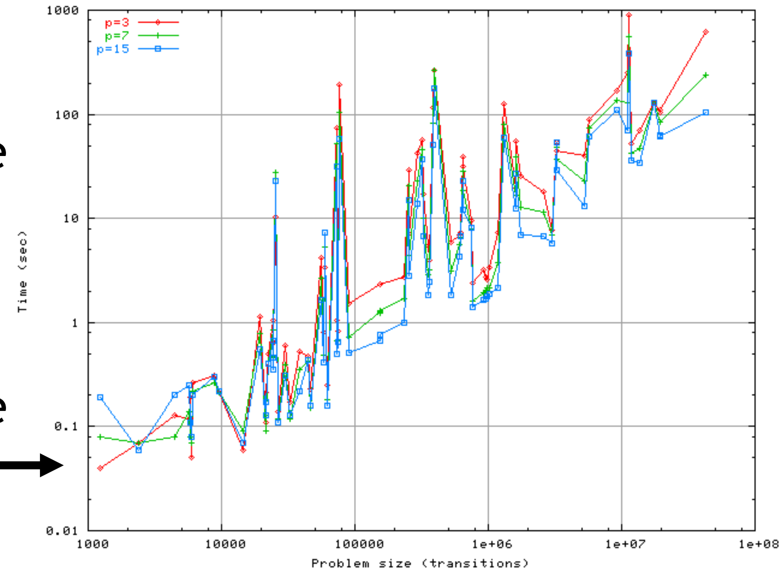
Observational equivalence





Safety equivalence

Taustar equivalence



Conclusion

- DSOLVE, a distributed algorithm for local resolution of BESs
- A distributed version of BISIMULATOR and a distributed generation of diagnostic for equivalence checking
- Generic implementation running on widely-used loosely-coupled parallel machines (clusters and NOWs)
- Extensive set of experiments performed on large BESs (VLTS benchmark suite)
  - Linear speedups (even superlinear for large BESs with particular forms)
  - Scalability w.r.t. BES size and number of processors

Future work

- Verification:
  - Tau-confluence reduction
  - Mu-calculus model-checking
  - Markovian bisimulation

- Other applications:
  - Horn clauses resolution
  - Abstract interpretation
  - Data flow analysis

For more information …

Christophe
Joubert

Radu
Mateescu

Nicolas
Descoubes

http://www.inrialpes.fr/vasy