Parallel and Distributed Methods in Probabilistic Model Checker PRISM

Marta Kwiatkowska
School of Computer Science

UNIVERSITY OF BIRMINGHAM

www.cs.bham.ac.uk/~mzk
www.cs.bham.ac.uk/~dxp/prism

INRIA, Grenoble, Nov 2005
Overview

• **Context**  
  - The PRISM model checker

• **Probabilistic model checking**  
  - What does it involve?  
  - Symbolic MTBDD-based techniques

• **Parallel symbolic approaches**  
  - Parallel numerical solution  
  - Grid-based techniques

• **Simulation and sampling-based model checking**  
  - Distributed engine

• **Conclusion and future work**
With thanks to...

- **Main collaborators on probabilistic model checking**
  - Gethin Norman, Dave Parker, Jeremy Sproston, Christel Baier, Roberto Segala, Michael Huth, Luca de Alfaro, Joost-Pieter Katoen, Markus Siegle, Antonio Pacheco

- **PRISM model checker implementation**
  - Dave Parker, Andrew Hinton, Rashid Mehmood, Hakan Younes, Stephen Gilmore, Michael Goldsmith, Conrado Daws, Fuzhi Wang

- **Parallelisation & Grid-enabling**
  - Dave Parker, Yi Zhang, Rashid Mehmood

- **And many more...**
The e-Scientist of the future

Remote access to high-performance computers, via Internet
Remote access to visualisation facilities, via Internet
Computational steering from anywhere, via PDA
Visualisation, on your laptop
Fast, online, accurate, ...
Probabilistic model checking...

in a nutshell

Probabilistic model

Probabilistic Model Checker

Probabilistic temporal logic specification

send $\rightarrow P_{0.9} (◊ \text{deliver})$

State 5: 0.6789
State 6: 0.9789
State 7: 1.0
...
State 12: 0
State 13: 0.1245

The probability
Probabilistic model checking inputs...

- **Models**
  - discrete time Markov chains (DTMCs)
  - continuous time Markov chains (CTMCs)
  - Markov decision processes (MDPs)
  - (currently indirectly) probabilistic timed automata (PTAs)

- **(Yes/No) temporal logic specification languages**
  - Probabilistic temporal logic PCTL (for DTMCs/MDPs)
  - Continuous Stochastic Logic CSL (for CTMCs)
  - Probabilistic timed computation tree logic PTCTL (for PTAs)

- **Quantitative specification language variants**
  - Probability values for logics PCTL/CSL/PTCTL (for all models)
  - Extension with expectation operator (for all)
  - Extension with costs/rewards (for all)
Probabilistic model checking involves...

- **Construction of models:**
  - DTMCs/CTMCs, MDPs, and PTAs (with digital clocks)

- **Implementation of probabilistic model checking algorithms**
  - graph-theoretical algorithms, combined with
    - (probabilistic) reachability
    - qualitative model checking (for 0/1 probability)
  - numerical computation - iterative methods
    - quantitative model checking (plot probability values, expectations, rewards, steady-state, etc, for a range of parameters)
    - exhaustive
  - sampling-based - simulation
    - quantitative model checking as above, based on many simulation runs
    - approximate
The PRISM probabilistic model checker

• History
  - Implemented at the University of Birmingham
  - First public release September 2001, ~7 years development
  - Open source, GPL licence, available freely for research and teaching
  - >2600 downloads, many users worldwide, >100 papers using PRISM, connection to several tools, >40 case studies and 6 flaws, ...

• Approach
  - Based on symbolic, BDD-based techniques
  - Multi-Terminal BDDs, first algorithm [ICALP'97]

• www.cs.bham.ac.uk/~d xp/prism/
  - For software, publications, case studies, taught courses using PRISM, etc
Overview of PRISM

• **Functionality**
  - Implements temporal logic probabilistic model checking
  - Construction of models: discrete and continuous Markov chains (DTMCs/CTMCs), and Markov decision processes (MDPs)
  - High-level model description language, state-based, also PEPA
  - Probabilistic temporal logics: PCTL and CSL
  - Extension with costs/rewards, expectation operator

• **Underlying computation combines graph-theoretical algorithms**
  - Reachability, qualitative model checking, BDD-based with numerical computation - iterative methods
  - Linear equation system solution - Jacobi, Gauss-Seidel, ...
  - Uniformisation (CTMCs)
  - Dynamic programming (MDPs)
  - Explicit and symbolic (MTBDDs, etc.)
PRISM real-world case studies

- **CTMCs**
  - Dynamic power management [HLDVT’02, FAC 2005]
  - Dependability of embedded controller [INCOM’04]
  - RKIP-inhibited ERK pathway (by Calder et al) [CMSB’01]
  - thinkteam (by ter Beek, Massink & Latella) [DSVIS’05]

- **MDPs/DTMCs**
  - Bluetooth device discovery [ISOLA’04]
  - Crowds anonymity protocol (by Shmatikov) [CSFW’02, JSC 2003]
  - Randomised consensus [CAV’01,FORTE’02]
  - Contract signing protocols (by Norman & Shmatikov) [FASEC’02]
  - Reliability of NAND multiplexing (with Shukla) [VLSI’04,TCAD 2005]

- **PTAs**
  - IPv4 ZeroConf dynamic configuration [FORMATS’03]
  - Root contention in IEEE 1394 FireWire [FAC 2003, STTT 2004]
  - IEEE 802.11 (WiFi) Wireless LAN MAC protocol [PROBMIV’02]
PRISM property specifications

- **PCTL/CSL (true/false) formula examples:**
  - \( P<0.001 \) \( [ \text{true U} \leq 100 \text{ error} ] \)
    “the probability of the system reaching an error state within 100 time units is less than 0.001”

- **Can also write query formulae:**
  - \( P=? \) \( [ \text{true U} \leq 10 \text{ terminate} ] \)
    “what is the probability that the algorithm terminates successfully within 10 time units?”

- **Instantaneous rewards, state-based, e.g. “queue size”:**
  - \( R=? \) \( [ I=T ] \), expected reward at time instant \( T \)?

- **Cumulative rewards, state/transition, e.g. “power consumed”:**
  - \( R=? \) \( [ C=T ] \), expected reward by time \( T \)?
  - \( R=? \) \( [ S ] \), expected long-run reward per unit time?
PRISM technicalities

- **(New) Simulator and sampling-based model checking**
  - allows to “execute” the model step-by-step or randomly
  - avoids state-space explosion, trading off accuracy

- **GUI implementation**
  - integrated editor for PRISM language
  - automatic graph plotting

- **Support for “experiments”**
  - e.g. \( P=? \begin{equation*} [\text{true } U \leq T \text{ error}] \end{equation*} \) for \( N=1..5, T=1..100 \)
  - repeats model checking for a range of model/formula parameters
  - plots on single graph
Screenshot: Graphs
Screenshot: Graphical input language
Simulator output: Workstation cluster

Formula label highlighted:
"premium" = (left_n>=left_mx&Topleft_n) | (right_n>=right_mx&Toright_n) | ((left_n+right_n)>=left_mx&Topleft_n&line_n&Toright_n)

<table>
<thead>
<tr>
<th>Step</th>
<th>left_n</th>
<th>left</th>
<th>right_n</th>
<th>right</th>
<th>r</th>
<th>line</th>
<th>line_n</th>
<th>Topleft</th>
<th>Topleft_n</th>
<th>Tonight</th>
<th>Tonight_n</th>
<th>Time</th>
<th>Rate</th>
<th>Reward</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1991</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1992</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1993</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1994</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1995</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1996</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1997</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1998</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1999</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scalability with sampling-based method

Cell cycle control
PRISM model size limit
What we have learnt from practice

• Probabilistic model checking
  - Is capable of finding ‘corner cases’ and ‘unusual trends’
  - Good for worst-case scenarios, for all initial states
  - Benefits from quantitative-style analysis for a range of parameters
  - Is limited by state space size
  - Useful for real-world protocol analysis, power management, performance, biological processes, …

• Simulation and sampling-based techniques
  - Limited by accuracy of the results, not state-space explosion
  - May need to rerun experiments for each possible start state, not always feasible
  - Statistical methods in conjunction with sampling help
  - Nested formulas may be difficult
New directions and challenges

- Often not practical to analyse the full system beforehand
  - Adaptive methods
    - Genetic algorithm-based methods [Jarvis et al, 2005]
  - Online methods
    - Continuous approximations using ODEs [Gilmore et al, 2005]
    - Applicable for sufficiently large numbers of entities
    - Work with PEPA, derivation of ODEs

- Scalability challenge
  - State-space explosion has not gone away...
  - Parallelisation
  - Distributed computation
  - Compositionality
  - Abstraction
  - Approximate methods
Why parallelise?

- **Experience with PRISM indicates**
  - Symbolic representation very compact, $>10^{10}$ states for CTMCs
  - Extremely large models feasible depending on regularity
  - Numerical computation often slow
  - Sampling-based computation can be even slower

- **Can parallelise the symbolic approach**
  - Facilitates extraction of the dependency information
  - Compactness enables storage of the full matrix at each node
  - Focus on steady-state solution for CTMCs, can generalise
  - Use wavefront techniques to parallelise

- **Easy to distribute sampling-based computation**
  - Individual simulation runs are independent
Numerical Solution for CTMC/DTMCs

- Steady-state probability distribution can be obtained by solving linear equation system:
  - \( \pi Q = 0 \) with constraint \( \sum_i \pi_i = 1 \)

- Consider the more general problem of solving:
  - \( Ax = b \),
  - where \( A \) is \( n \times n \) matrix, \( b \) vector of length \( n \)

- Numerical solution techniques
  - Direct, not feasible for very large models
  - Iterative stationary (Jacobi, Gauss-Seidel), memory efficient
  - Projection methods (Krylov, CGS, ...), fastest convergence, but require several vectors

- Transient probabilities similar
  - Computed via an iterative method (uniformisation)
Symbolic techniques for CTMCs

- **Explicit matrix representation**
  - Intractable for very large matrices

- **Symbolic representations**
  - e.g. Multi-Terminal Binary Decision Diagrams (MTBDDs), matrix diagrams and Kronecker representation
  - Exploit regularity to obtain compact matrix storage
  - Also faster model construction, reachability, etc
  - Sometimes also beneficial for vector storage

- **This paper uses MTBDDs (Clarke et al) and derived structures**
  - Underlying data structure of the PRISM model checker
  - Enhanced with caching-based techniques that substantially improve numerical efficiency
MTBDD data structures

- Recursive, based on Binary Decision Diagrams (BDDs)
  - Stored in *reduced* form (DAG), with isomorphic subtrees stored only once
  - Exploit regularity to obtain *compact* matrix storage
Matrices as MTBDDs

- **Representation**
  - Root represents the whole matrix
  - Leaves store matrix entries, reachable by following *paths* from the root node
Matrices as MTBDDs

- Recursively descending through the tree
  - Divides the matrix into submatrices
  - One level, divide into two submatrices
Matrices as MTBDDs

- Recursively descending through the tree
  - Provides a convenient block decomposition
  - Two levels, divide into four blocks
A two-layer structure from MTBDDs

- Can obtain block decomposition, store as two sparse matrices
  - Enables fast row-wise access to blocks and block entries

(Parker’02, Mehmood’05)
Parallel symbolic numerical engine

- **MTBDDs**
  - Provide a convenient block decomposition of the matrix (computation) into submatrices

- **Parallel symbolic solution techniques for \( Ax = b \)**
  - Store full matrix at each node (see also Kemper et al)
  - Solve in block form
  - **Tasks**, each determined by a matrix block
  - The execution order determined by computational dependency

- **Techniques**
  - Parallel Jacobi, **CGS**
  - **Gauss-Seidel** more difficult
    - Non-symbolic (Joubert et al) relies on permutation, not feasible in the symbolic context
    - Here, first symbolic parallelisation of Gauss-Seidel
Gauss-Seidel

- Computes one matrix row at a time
- Updates $i^{th}$ element using most up-to-date values
- Computation for a single iteration, $n \times n$ matrix:
  1. for ($0 \leq i < n$)
  2. $x_i := (b_i - \sum_{0 \leq j < n, j \neq i} A_{ij} x_j) / A_{ii}$

- Can be reformulated in block form, $N \times N$ blocks, length $M$
  1. for ($0 \leq p < N$)
  2. $v := b_p$
  3. for each block $A_{pq}$ with $q \neq p$
  4. $v := v - A_{pq} x_q$
  5. for ($0 \leq i < M$, $i \neq j$)
  6. $x_{(p)i} := (v_i - \sum_{0 \leq j < M} A_{(pp)ij} x_{(p)j}) / A_{(pp)ii}$

- Computes one matrix block at a time
Parallelising Gauss-Seidel

- Inherently sequential for dense matrices
  - Uses results from current and previous iterations

- Permutation has no effect on correctness of the result
  - Can be exploited to achieve parallelisation for certain sparse matrix problems, e.g. Koester, Ranka & Fox 1994

- The block formulation helps, although
  - Requires row-wise access to blocks and block entries
  - Need to respect computational dependencies, i.e. when computing vector block $x_{(p)}$, use values from current iteration for blocks $q < p$, and from previous iteration for $q > p$

- Idea: propose to use wavefront techniques
  - Extract dependency information and form execution schedule
Wavefront techniques

• An approach to parallel programming, e.g. Joubert et al ‘98
  - Divide a computation into many tasks
  - Form a schedule for these tasks

• A schedule contains several wavefronts
  - Each wavefront comprises tasks that are algorithmically independent of each other
  - i.e. correctness is not affected by the order of execution

• The execution is carried out from one wavefront to another
  - Tasks assigned according to the dependency structure
  - Each wavefront contains tasks that can be executed in parallel
Dependency graph from MTBDD

- By traversal of top levels of MTBDD, as for top layer
Generating a Wavefront Schedule

- By colouring the dependency graph

- Can generate a schedule to let the computation perform from one colour to another
Wavefront with MTBDDs

- **Our parallelisation of Gauss-Seidel**
  - Allows much larger CTMC models to be solved
  - Has good overall speedup

- **Symbolic approach particularly well suited to Wavefront parallelisation of Gauss-Seidel**
  - Easy to extract task dependency information
  - Reduced memory requirement and communication load

- **Gauss-Seidel excellent candidate to solve very large linear equation systems**
  - Small memory requirement (only requires one iteration vector, vs 2 for Jacobi and 6 for CGS)
  - Method generalises to other symbolic techniques and application domains
Implementation

• Implemented on a Ethernet and Myrinet-enabled PC cluster
  – Use MPI (the MPICH implementation)
  – Prototype extension for PRISM, uses PRISM numerical engines and CUDD package for MTBDDs (Somenzi)
  – 32 nodes available

• Evaluated on a range of benchmarks
  – Kanban, FMS and Polling system

• Optimisations for PC-cluster environments
  – Non-blocking inter-processor communication is used to interleave communication and computation
  – Load-balancing is implemented to distribute computation load evenly between processors and minimise communication load
  – Cache mechanism is used to reduce communication further
Experimental results: models

- Parameters and statistics of models
  - Include Kanban 9,10 and FMS 13, previously intractable
  - All compact, requiring less than 1GB

<table>
<thead>
<tr>
<th>Model</th>
<th>States</th>
<th>Transitions</th>
<th>Blocks (N)</th>
<th>Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MTBDD</td>
<td>Sparse</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMS (K=11)</td>
<td>54,682,992</td>
<td>518,030,370</td>
<td>1,365</td>
<td>297</td>
</tr>
<tr>
<td>FMS (K=12)</td>
<td>111,414,940</td>
<td>1,078,917,632</td>
<td>1,820</td>
<td>558</td>
</tr>
<tr>
<td>FMS (K=13)</td>
<td>216,427,680</td>
<td>2,136,215,172</td>
<td>2,380</td>
<td>1,005</td>
</tr>
<tr>
<td>Kanban (K=7)</td>
<td>41,644,800</td>
<td>450,455,040</td>
<td>120</td>
<td>18</td>
</tr>
<tr>
<td>Kanban (K=8)</td>
<td>133,865,325</td>
<td>1,507,898,700</td>
<td>165</td>
<td>43</td>
</tr>
<tr>
<td>Kanban (K=9)</td>
<td>384,392,800</td>
<td>4,474,555,800</td>
<td>220</td>
<td>95</td>
</tr>
<tr>
<td>Kanban (K=10)</td>
<td>1,005,927,208</td>
<td>12,032,229,352</td>
<td>286</td>
<td>195</td>
</tr>
<tr>
<td>Polling (K=20)</td>
<td>31,457,280</td>
<td>340,787,200</td>
<td>308</td>
<td>65</td>
</tr>
<tr>
<td>Polling (K=21)</td>
<td>66,060,288</td>
<td>748,683,264</td>
<td>324</td>
<td>141</td>
</tr>
<tr>
<td>Polling (K=22)</td>
<td>138,412,032</td>
<td>1,637,875,712</td>
<td>340</td>
<td>307</td>
</tr>
</tbody>
</table>
Experimental results: time

- Total execution times (in seconds) with 1 to 32 nodes
  - Termination condition maximum relative difference $10^{-6}$
  - Block numbers selected to minimise storage

<table>
<thead>
<tr>
<th>Num. nodes</th>
<th>FMS $K=11$</th>
<th>FMS $K=12$</th>
<th>FMS $K=13$</th>
<th>Kanban $K=7$</th>
<th>Kanban $K=8$</th>
<th>Kanban $K=9$</th>
<th>Kanban $K=10$</th>
<th>Polling $K=20$</th>
<th>Polling $K=21$</th>
<th>Polling $K=22$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15,990</td>
<td>35,637</td>
<td>O/M</td>
<td>4,683</td>
<td>19,417</td>
<td>O/M</td>
<td>O/M</td>
<td>8,764</td>
<td>14,195</td>
<td>45,485</td>
</tr>
<tr>
<td>2</td>
<td>10,349</td>
<td>22,986</td>
<td>O/M</td>
<td>3,351</td>
<td>16,419</td>
<td>O/M</td>
<td>O/M</td>
<td>6,451</td>
<td>10,834</td>
<td>37,713</td>
</tr>
<tr>
<td>4</td>
<td>6,548</td>
<td>15,264</td>
<td>O/M</td>
<td>1,925</td>
<td>8,099</td>
<td>34,755</td>
<td>O/M</td>
<td>4,906</td>
<td>6,301</td>
<td>21,553</td>
</tr>
<tr>
<td>8</td>
<td>3,991</td>
<td>9,212</td>
<td>O/M</td>
<td>1,106</td>
<td>4,474</td>
<td>16,271</td>
<td>O/M</td>
<td>2,123</td>
<td>3,463</td>
<td>11,287</td>
</tr>
<tr>
<td>12</td>
<td>3,218</td>
<td>7,148</td>
<td>O/M</td>
<td>806</td>
<td>3,314</td>
<td>11,452</td>
<td>45,206</td>
<td>1,488</td>
<td>2,433</td>
<td>8,338</td>
</tr>
<tr>
<td>16</td>
<td>2,446</td>
<td>5,642</td>
<td>12,544</td>
<td>611</td>
<td>2,555</td>
<td>9,522</td>
<td>29,674</td>
<td>1,153</td>
<td>1,807</td>
<td>5,929</td>
</tr>
<tr>
<td>24</td>
<td>1,860</td>
<td>4,419</td>
<td>9,657</td>
<td>503</td>
<td>1,915</td>
<td>6,741</td>
<td>20,560</td>
<td>769</td>
<td>1,335</td>
<td>4,546</td>
</tr>
<tr>
<td>28</td>
<td>1,623</td>
<td>4,038</td>
<td>8,173</td>
<td>450</td>
<td>1,679</td>
<td>5,753</td>
<td>18,599</td>
<td>736</td>
<td>1,203</td>
<td>3,491</td>
</tr>
<tr>
<td>32</td>
<td>1,504</td>
<td>3,689</td>
<td>7,693</td>
<td>351</td>
<td>1,526</td>
<td>5,134</td>
<td>15,750</td>
<td>650</td>
<td>858</td>
<td>3,086</td>
</tr>
</tbody>
</table>
Experimental results: FMS speed-up

![Graph showing speedup vs. number of processors for FMS 11 and FMS 12.]
Experimental results: Kanban speed-up
Experimental results: Polling speed-up
What we have learnt...

• **Experience with PRISM indicates**
  - Symbolic representation very compact, \(>10^{10}\) states for CTMCs
  - Extremely large models feasible depending on regularity
  - Numerical computation often slow
  - Sampling-based computation can be even slower

• **Can parallelise the symbolic approach**
  - Facilitates extraction of the dependency information
  - Compactness enables storage of the full matrix at each node
  - Focus on steady-state solution for CTMCs, can generalise
  - Use **wavefront** techniques to parallelise

• **Easy to distribute sampling-based computation**
  - Individual simulation runs are independent
A Grid engine

• Integrate a parallel numerical engine into PRISM
  - Access cluster from desktop
  - Manage remote computation resources for end users
  - Free end users from learning remote scheduling systems
  - Handling data transfer on behalf of end users
  - Monitoring job execution on remote computation resources

• Implemented parallel symbolic numerical engine
  - Based on MTBDD data structures
  - Solving linear equation systems for analysis of CTMC and DTMC
  - A Parallel Gauss-Seidel Iterative Method
    • for shared memory machines.
    • for message passing machines.
The Role of Globus Toolkits

- **Globus version 4 (GT4)**
  - Web-services and HPC standard

- **Provide building blocks for our middleware**
  - GSI for security
  - GRAM for job management
  - GSI-OpenSSH for file transfer
  - Grid services for data handling and job monitoring

- **Offers convenience and high-performance to end users**
  - Linear equations generated by PRISM
  - Matrices and vectors transferred to cluster
  - Solution can be queries, transferred back
Structure of Grid-enabled PRISM
PRISM with Globus

• **Job Submission Component**
  - Based on WS-GRAM
  - Generates job description files
  - Communicates with WS-GRAM services at remote resources

• **Data Transfer**
  - Using GSI-OpenSSH for file transfer
    - Matrices
    - Vectors
  - Create grid services for fine-grained data access
    - Block by block

• **Job Monitoring (under development)**
  - Information about job status
  - Runtime information
  - Convergence rate information
### Experimental evaluation

**Table 2. Total computation time (seconds) for model checking of Kanban models.**

<table>
<thead>
<tr>
<th>Num. nodes</th>
<th>Kanban $K=8$</th>
<th>$K=9$</th>
<th>$K=10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19,868</td>
<td>O/M</td>
<td>O/M</td>
</tr>
<tr>
<td>4</td>
<td>8,666</td>
<td>38,451</td>
<td>O/M</td>
</tr>
<tr>
<td>8</td>
<td>4,698</td>
<td>18,120</td>
<td>O/M</td>
</tr>
<tr>
<td>16</td>
<td>2612</td>
<td>10,172</td>
<td>32,459</td>
</tr>
<tr>
<td>24</td>
<td>2,006</td>
<td>7,123</td>
<td>22,971</td>
</tr>
</tbody>
</table>
Summary

• **Grid-based middleware for PRISM**
  - Allows much larger CTMC models to be solved
  - Provides easy access of remote parallel computation resources for end users
  - A foundation for future parallelisation work in PRISM

• **Symbolic approach particularly well suited to Wavefront parallelisation of Gauss-Seidel**
  - Easy to extract task dependency information
  - Reduced memory requirement and communication load
  - Good speed-up

• **Sampling-based Monte Carlo**
  - Easy to parallelise, simulation traces independent
  - Implementing SSH based scheduling
Challenges for future

• State-space explosion has not gone away...
  - Are Grid techniques the answer?

• Exploiting structure
  - Abstraction, compositionality...
  - Parametric probabilistic verification?

• Efficient methods for continuous models
  - Continuous PTAs? Continuous time MDPs? LMPs?
  - Sampling-based, approximate model checking

• Proof assistant for probabilistic verification?

• Real software, not models!
For more information...

J. Rutten, M. Kwiatkowska, G. Norman and D. Parker

**Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems**

P. Panangaden and F. van Breugel (editors), CRM Monograph Series, vol. 23, AMS
March 2004

[www.cs.bham.ac.uk/~dxp/prism/](http://www.cs.bham.ac.uk/~dxp/prism/)

- Case studies, statistics, group publications
- Download, version 2.0 (approx. 2200 users)
- Linux/Unix, Windows, Macintosh versions
- Publications by others and courses that feature PRISM...