Analyzing Cluster Scheduling Schemes by Probabilistic Model Checking[*]

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
Wolfgang.Schreiner@risc.jku.at

Tamás Bérczes
Faculty of Informatics
University of Debrecen, Hungary
berczes.tamas@inf.unideb.hu

Ádám Tóth
Faculty of Informatics
University of Debrecen, Hungary
adamtoth102@gmail.com

September 22, 2014

Abstract

We report in this paper on initial results of modeling and analyzing with the probabilistic model checker PRISM various cluster scheduling schemes that were introduced by Do, Vu, Tran, and Nguyen in their paper "A generalized model for investigating scheduling schemes in computational clusters" and analyzed by simulation there. The preliminary results are encouraging, but there also remain some open issues that need to be addressed.

[*]Supported by the project HU 10/2012 of the Austrian Academic Exchange Service (ÖAD) and by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. This project has been supported by the European Union, co-financed by the European Social Fund.

Contents

1. Introduction

We report in this paper on our initial results of modeling and analyzing various scheduling schemes for computational clusters with the help of the probabilistic model checker PRISM [2, 4]. These models were originally presented in [1] and analyzed there by simulation; our goal is to validate and to potentially generalize these results by constructing and solving precise mathematical system models: from a description of the model in the language of PRISM, the tool first automatically generates a continuous-time Markov chain (CTMC) which it then subsequently solves by application of a numerical solver. Previously we have applied this technique to the validation of simulation results from the domain of mobile networks [5]; with this report we extend our experience to the domain of cluster scheduling.

The system presented in [1] consists of K classes of servers; the servers in each class have identical performance and energy consumption; typically, the higher the performance of a server is, the more energy it consumes. The devised scheduling schemes are based on a ranking of server classes, either by high performance (HP) or by energy efficiency (EE). Roughly speaking, when a new job arrives, highly ranked servers are preferred over lowly ranked ones. In more detail, there are three schemes:

1. *Separate Queue Scheme:* each server has a separate queue from which it accepts and processes jobs. A new job is placed into the shortest queue; if there are multiple such queues, the queue of the most highly ranked server is chosen.

2. *Class Queue Scheme:* each server class has a separate queue from which the servers of this class accept and process jobs. A new job is placed into the shortest queue; if there are multiple such queues, the queue of the most highly ranked class is chosen.

3. *Common Queue Scheme:* there is a single common queue from which all servers accept and process jobs. A new job is forwarded to the most highly ranked free server; if there is no such server, the job is placed into the common queue.

The various schemes are compared in [1] with respect to various criteria for both the HP and the EE ranking. The paper assumes an infinite source model where jobs arrive with a constant rate λ that is determined to establish a desired system utilization U in the range of 50% to 90%.

In this paper, we devise analogous PRISM models for the three scheduling schemes with infinite sources and both the HP and the EE ranking. We analyze these models with respect to some of the criteria reported in [1] (service, response, and waiting times), and compare the results. Furthermore, we also provide finite source versions of the models (which were not investigated in [1]) and perform corresponding analyses.

The rest of the paper is organized as follows: in Section 2, we present the infinite source models and their analysis while in Section 3 we present the finite source models and their analysis. In Section 4, we present our conclusions and discuss further work. Appendix A contains the full listing of the infinite source models and of the properties that were used for the analysis; Appendix B contains the corresponding listings for the finite source variants. Appendix C gives alternative descriptions of some of the models that (while yielding the same analysis results) demonstrate another style of model descriptions in PRISM.

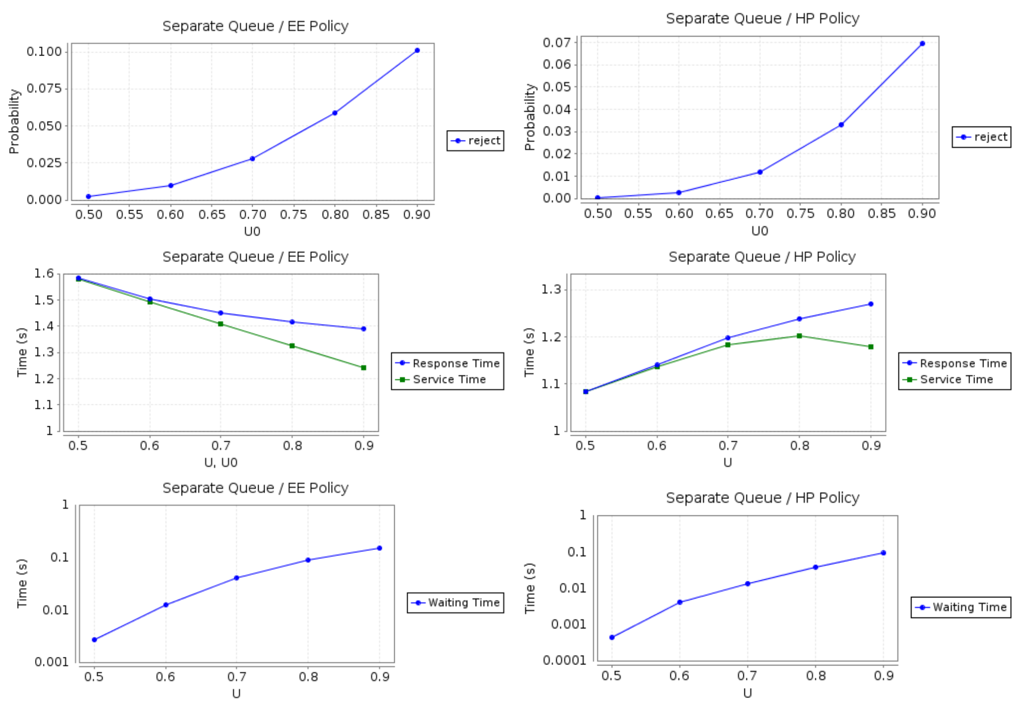

Figure 1: Separate Queue ($3 \cdot 8$ Servers, $Q = 1$)

2. Infinite Source Models

In this section, we present and analyze the infinite source versions of PRISM models that correspond to those in [1].

2.1. Separate Queue

Appendix A.1 lists the PRISM version of the "separate queue" model for $K = 3$ classes of $M = 8$ servers each. The queue of server j in class i is modeled by a variable $q_{i,j}$ that denotes the number of jobs in the queue. The transitions with precondition $q_{i,j} > 0$ process the jobs with rate μ_i associated to server class i; the transition with precondition *true* describes the scheduling scheme by the changes of all variables $q_{i,j}$ under the various load situations.

To analyze this model, the reward structures "stime" and "qload" are introduced; reward structure "stime" assigns to every state as a reward the time that servicing a job that enters the system in that state will have (because it is placed in the queue of a server with corresponding performance); reward structure "qload" assigns to every state as a reward the number of jobs that are currently in the system.

The results of the analysis are presented in Figure 1. They correspond quite well to those presented in Figures 4–6 of [1] for utilizations up to 70% but are significantly lower for utilizations from 80%1; the main reason is apparently the high rejection probability depicted at the top of the figure, which is the probability that an arriving job does not find a place in any of the server

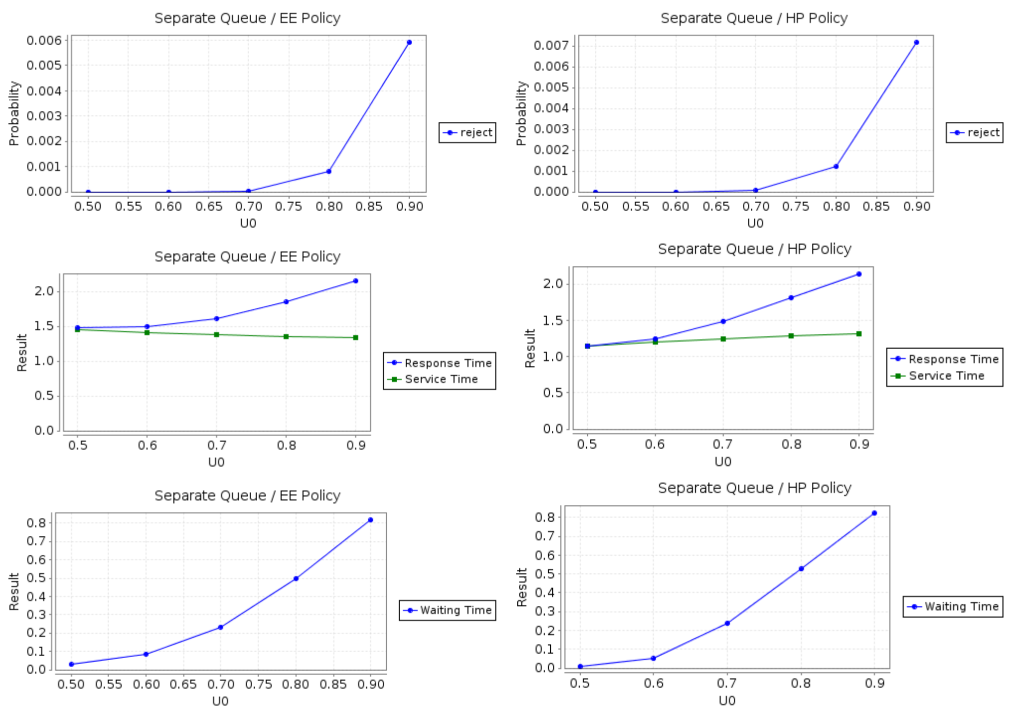

Figure 2: Separate Queue ($3 \cdot 4$ Servers, $Q = 3$)

queues.

This issue highlights the main difference of the PRISM model to the originally developed one: since PRISM is only able to analyze finite state systems, we have to impose an upper bound Q on the size of every queue. Furthermore, since the size of the state space increases exponentially with Q and the model checking time is quadratically in the number of states [3], we could only check the system for $Q = 1$ which resulted in the inaccurate results for higher utilization.

To produce more accurate results, we cut the number of servers by two (using only $M = 4$ servers by class) which allowed us to increase the queue size to $Q = 3$. The corresponding model is listed in Appendix A.2; the results of the analysis are depicted in Figure 2. We see also for higher utilization values only very moderate rejection probabilities (less than 1%). While the figures are not comparable with those in [1], the overall shapes of the curves now more closely corresponds to the results presented there.

2.2. Class Queue

Appendix A.3 lists the PRISM version of the "class queue" model for $K = 3$ classes of $M = 8$ servers each. In this model, we only need three class queues represented by variables q_i. The transitions with preconditions $q_i > 0$ process jobs from these queues; the transition with precondition *queue* describes the scheduling scheme by the changes of all variables q_i under the various load situations. The reward structures used in the properties are analogous to those of the "separate queue" case.

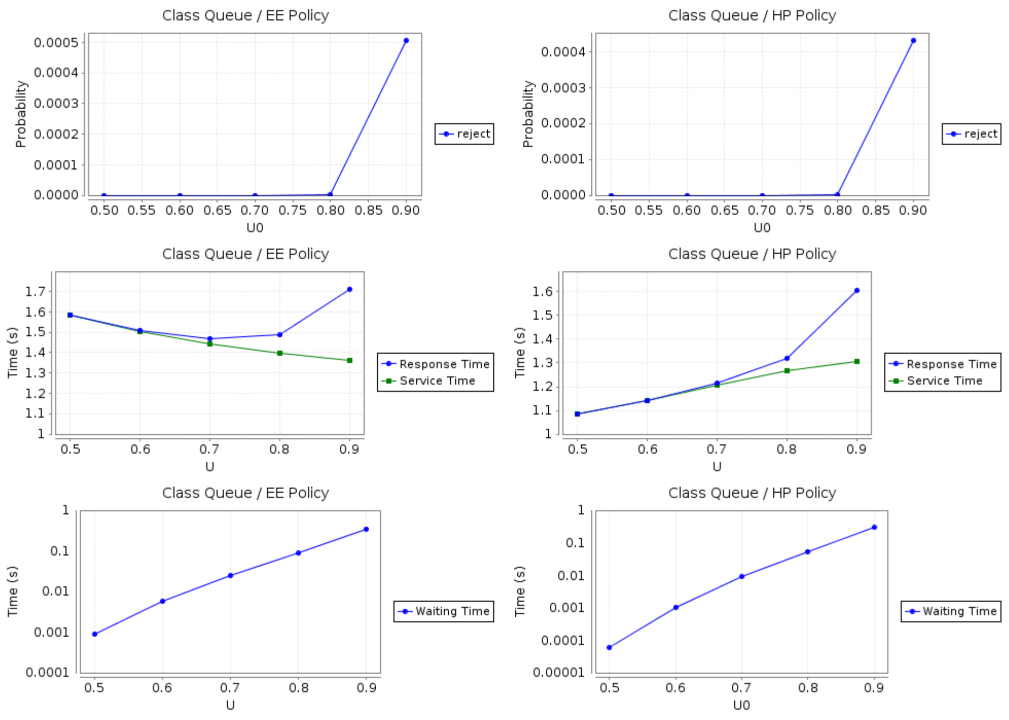

Figure 3: Class Queue ($3 \cdot 8$ Servers, $Q = 24$)

Since the state space of this model is comparatively small, we can use in our analysis $Q = 24$ with very small rejection rates. The results depicted in Figure 3 are apparently very close to those presented in Figures 4–6 of [1]; we thus consider the model as adequate.

2.3. Common Queue

Appendix A.4 lists the PRISM version of the "common queue" model for $K = 3$ classes of $M = 8$ servers each. In this model, we only need one class queues represented by variables q; the three variables q_i now represent the number of jobs that are concurrently processed by a server of type i.

The transitions with preconditions $q_i > 0$ describe the concurrent processing of jobs by q_i servers of type i with total rate $q_i \cdot \mu_i$. The transitions with preconditions $q_i < M \wedge q > 0$ describe the processing of a job from the queue; since there are $M - q_i$ free servers available for processing such jobs, the processing rate is $(M - q_i) \cdot mu_i$ (this simplifies the actual situation; it actually describes the average runtime of such a transition assuming that there are $M - q_i$ jobs ready to be grabbed by a server of type i). The transition with precondition *true* describes the scheduling scheme by the changes of q and all variables q_i under the various load situations.

The reward structures used in the properties are analogous to those of the "separate queue" case. There is however a difference: in the case that all servers are busy, we cannot accurately predict the service time for the job that is placed into the queue. We therefore perform multiple analysis runs with the lower and upper bounds of the service time (corresponding to the fastest

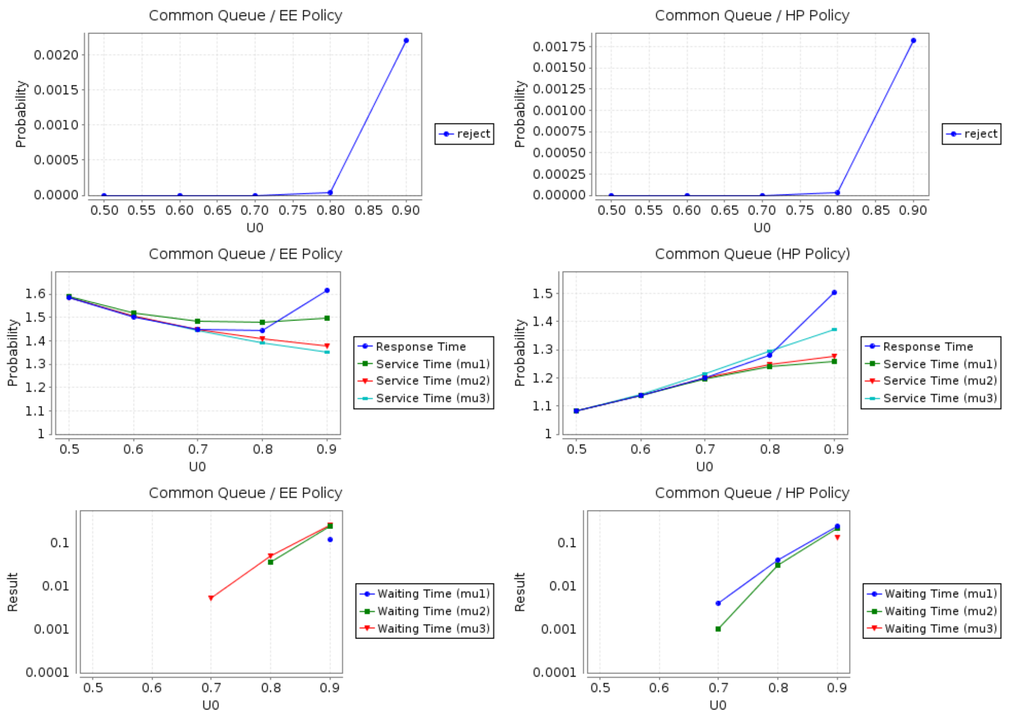

Figure 4: Common Queue ($3 \cdot 8$ Servers, $Q = 32$)

and the slowest server class) and for a medium value. The results depicted in Figure 4 thus depict a range of values for the mean service times and waiting times. The results generally agree well with those depicted in Figures 4–6 of [1] (for small system utilizations, the waiting times are approximated to 0; these values not shown in above diagrams due to the logarithmic time scale).

3. Finite Source Models

We have adapted the models presented in the previous section also to the finite source case; see the listings presented in Appendices B.1, B.2, and B.3. The main difference to the infinite source models is the introduction of a module "Sources" that governs a pool of at most N job each of which is forwarded with rate λ to the server system; if a job is finished, it is returned to the pool. The system thus maintains the invariant that the total number of jobs in the pool and in the server system equals N at any time. For this purpose, the transition "server" that accepts a job has to be extended by a guard condition that prevents the execution of the transition if the system is full.

In the following we analyze the various models under the HP policy.

Figure 5 describes the result of the analysis for the "separate queue" scheme with $K = 3$ classes of $M = 4$ servers each; we use a pool of $N = 60$ sources and as for the infinite source case a queue size $Q = 3$ (we omit the case $M = 8$ due to the larger state space of the finite source system which makes also experiments with queue size $Q = 1$ problematic). Appendix C contains an alternative formulation of the model which demonstrates that the same mathematical

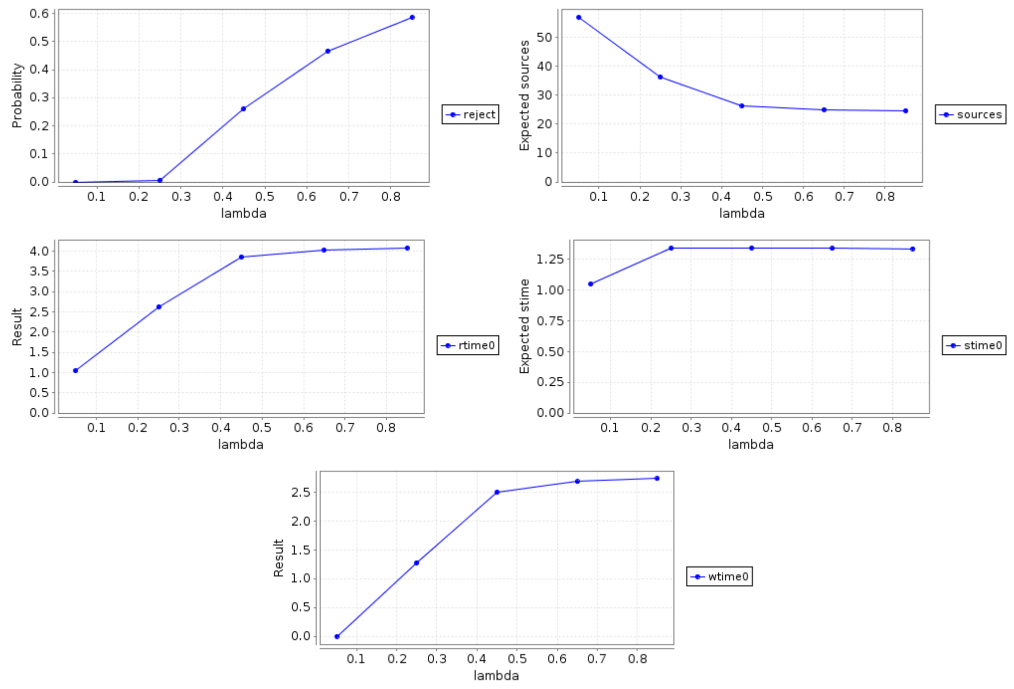

Figure 5: Separate Queue ($3 \cdot 4$ Servers, $Q = 3$)

model can be described in vastly syntactic ways; the analysis of that model however yields the same numerical results.

Although we have no way to validate the results by comparison with the result of an independent evaluation, the results look plausible: for $\lambda \geq 0.45$ the number of jobs in the pool approximates the possible minimum (24=60-3*4*3), i.e., the server system gets saturated and the rejection probability drastically increases; the response time and the waiting times here reach their maxima. However, already for $\lambda \geq 25$, there are sufficiently many jobs in the server system to also utilize the slowest servers, thus the mean service time of accepted jobs reaches its maximum already here.

Figures 6 and 7 describe the results for the “class queue” scheme for both $M = 8$ and $M = 4$ servers per class; due to the smaller state space a queue size $Q = 24$ that yields very low rejection probability can be analyzed without problems.

Analogously Figures 8 and 9 describe the results for the “common queue” scheme for both $M = 8$ and $M = 4$ servers per class; for the common queue a slightly larger size $Q = 32$ is chosen to yield low rejection probabilities.

4. Conclusions

The initial results presented in this paper justify the assumption that probabilistic model checking with PRISM can compete with simulation in analyzing the performance of cluster scheduling

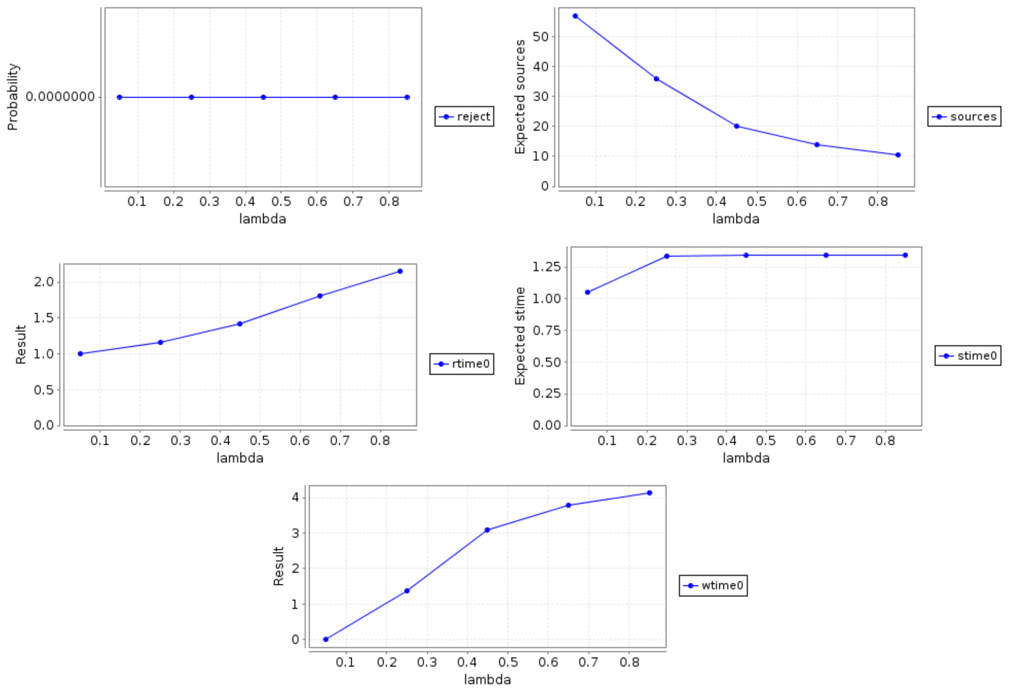

Figure 6: Class Queue (3 · 4 Servers, $Q = 24$)

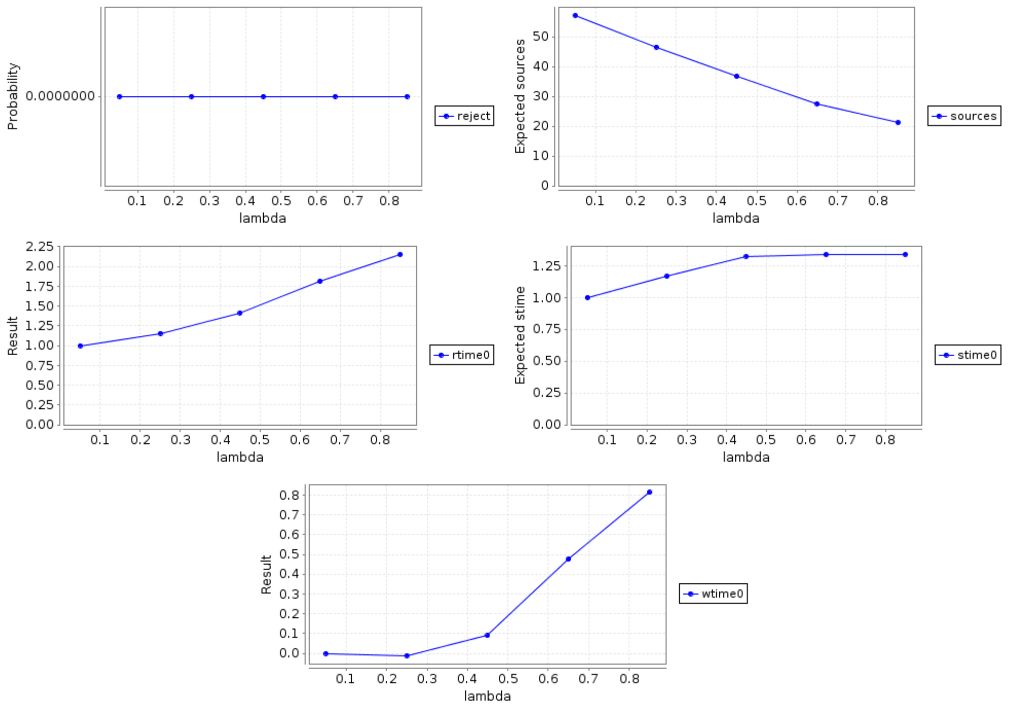

Figure 7: Class Queue (3 · 8 Servers, $Q = 24$)

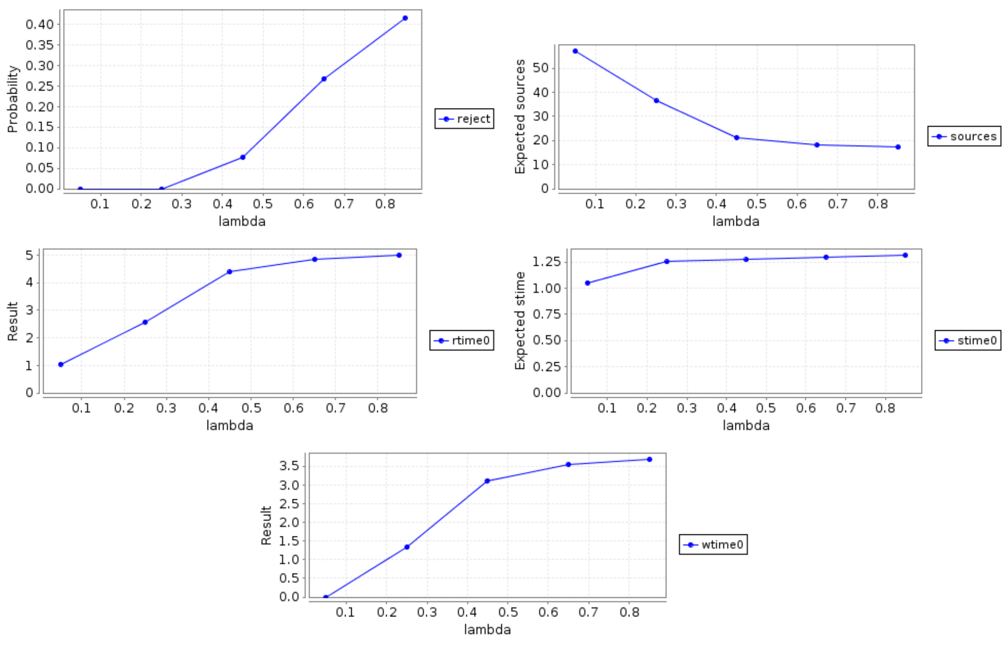

Figure 8: Common Queue (3 · 4 Servers, $Q = 32$)

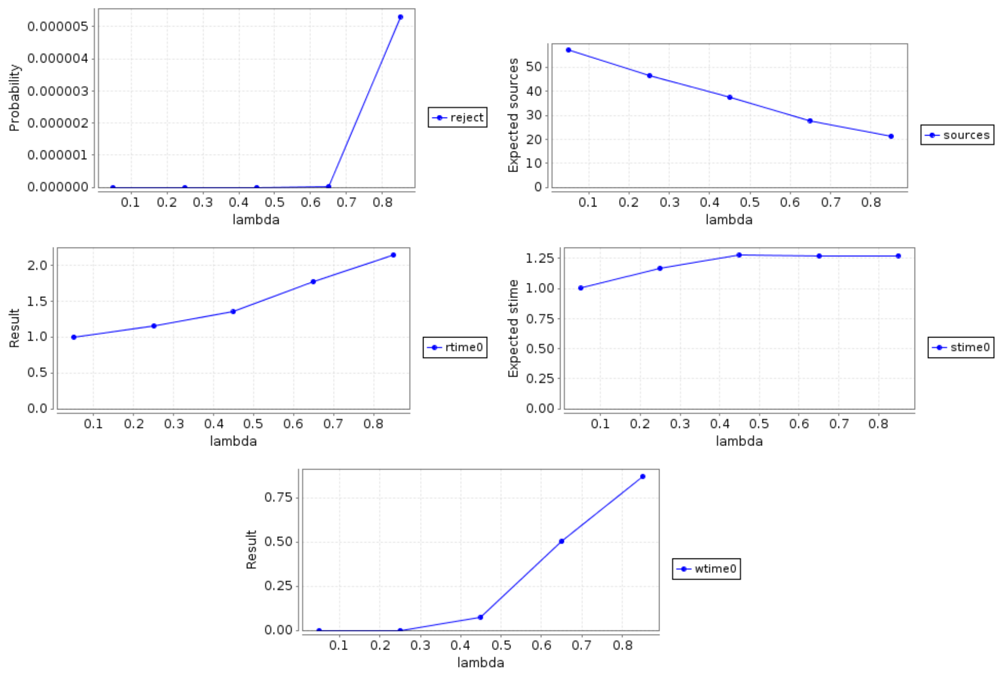

Figure 9: Common Queue (3 · 8 Servers, $Q = 32$)

schemes; we can thus analyze models accurately without resorting to the uncertainties of simulation.

However, there are also some issues that must be addressed:

- Models whose accurate description depends on state spaces that are significantly larger than 10^8 states defy model checking due to unrealistic requirements with respect to memory and time (the models presented in this paper required about 1GB of memory and about 5 mins of model checking time). In particular, we could thus not analyze the "separate queue" scheme with 24 servers.
- Some quantities may be hard to determine by the mechanisms of PRISM relying on reward structures and the PRISM property specification language. In particular, we could not determine the mean service time of jobs in the "common queue" scheme (but only lower and upper bounds).

On the other side, it was very easy to turn infinite source models into finite source models and analyze these without major changes to the structure of models and properties.

Up to now we have only analyzed a small fraction of the measures that were analyzed in [1], essentially only those related to time. In the future, we plan to extend the range of the analysis of our models in particular by also analyzing the energy-related measures; this will certainly yield new challenges and additional insights.

References

[1] Tien v. Do, Binh T. Vu, Xuan T. Tran, and Anh P. Nguyen. "A Generalized Model for Investigating Scheduling Schemes in Computational Clusters". In: *Simulation Modelling Practice and Theory* 37 (2013), pp. 30–42.

[2] M. Kwiatkowska, G. Norman, and D. Parker. "PRISM 4.0: Verification of Probabilistic Real-time Systems". In: *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*. Ed. by G. Gopalakrishnan and S. Qadeer. Vol. 6806. Lecture Notes in Computer Science. Springer, 2011, pp. 585–591.

[3] Marta Kwiatkowska, Gethin Norman, and David Parker. "Stochastic Model Checking". In: *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*. Ed. by M. Bernardo and J. Hillston. Vol. 4486. Lecture Notes in Computer Science (Tutorial Volume). Springer, 2007, pp. 220–270.

[4] David A. Parker, ed. *PRISM — Probabilistic Symbolic Model Checker*. Department of Computer Science, University of Oxford, UK. 2013. URL: http://www.prismmodelchecker.org.

[5] Wolfgang Schreiner, Tamás Bérczes, and János Sztrik. *Probabilistic Model Checking on HPC Systems for the Performance Analysis of Mobile Networks*. Technical Report. Johannes Kepler University Linz, Austria: Research Institute for Symbolic Computation (RISC), Sept. 2013. URL: http://www.risc.jku.at/publications/download/risc_4819/main.pdf.

A. The Infinite Source Models

A.1. Separate Queue ($3 \cdot 8$ Servers)

```
// ----------------------------------------------------------------------
// InfiniteSeparateHP.prism
// A scheduling model for computational clusters.
//
// The model is the "separate queue" model with
// "high-performance priority" described in
//
//   Tien v. Do, Binh t. Vu, Xuan T. Tran, Anh P. Nguyen:
//   "A generalized model for investigating scheduling schemes in
//   computational clusters", Simulation Modelling Practice and
//   Theory, 37 (2013), 30-42.
//
// Authors: Berczes Tamas <berczes.tamas@inf.unideb.hu> and
// Wolfgang Schreiner <Wolfgang.Schreiner@risc.jku.at>
//
// Copyright (C) 2014 Department of Informatics Systems and Networks,
// University of Debrecen, Debrecen, Hungary (http://irh.inf.unideb.hu)
// and Research Institute for Symbolic Computation, Johannes Kepler
// University, Linz, Austria (http://www.risc.jku.at)
// ----------------------------------------------------------------------

// checking parameters: "sparse" (1GB), "Jacobi", epsilon=0.01

// Q=1 gives rejection probability from 2E-4 (U0=0.5) to 0.08 (U0=0.9)
// so the results for high U0 are not very accurate

// continuous time markov chain (ctmc) model
ctmc

// ----------------------------------------------------------------------
// system parameters
// ----------------------------------------------------------------------

// system utilization (50%, 60%, 70%, 80%, 90%)
const double U0;

// arrival rates
const double lambda = U0*8*(rp1+rp2+rp3);

// queue size
const int Q = 1;

// number of server classes and number of servers per class
const int K = 3;
const int M = 8;

// ranking based on performance
const double rp1 = 1.0;
const double rp2 = 0.82;
const double rp3 = 0.43;
```

```
// ranking based on energy efficiency
const double re1 = 0.64;
const double re2 = 0.66;
const double re3 = 1.0;

// service rates according to chosen ranking
const double mu1 = rp1;
const double mu2 = rp2;
const double mu3 = rp3;

// ------------------------------------------------------------------
// system model
// ------------------------------------------------------------------

module System
  q11: [0..Q] init 0;
  q12: [0..Q] init 0;
  q13: [0..Q] init 0;
  q14: [0..Q] init 0;
  q15: [0..Q] init 0;
  q16: [0..Q] init 0;
  q17: [0..Q] init 0;
  q18: [0..Q] init 0;

  q21: [0..Q] init 0;
  q22: [0..Q] init 0;
  q23: [0..Q] init 0;
  q24: [0..Q] init 0;
  q25: [0..Q] init 0;
  q26: [0..Q] init 0;
  q27: [0..Q] init 0;
  q28: [0..Q] init 0;

  q31: [0..Q] init 0;
  q32: [0..Q] init 0;
  q33: [0..Q] init 0;
  q34: [0..Q] init 0;
  q35: [0..Q] init 0;
  q36: [0..Q] init 0;
  q37: [0..Q] init 0;
  q38: [0..Q] init 0;

  [] q11 > 0 -> mu1 : (q11' = q11-1);
  [] q12 > 0 -> mu1 : (q12' = q12-1);
  [] q13 > 0 -> mu1 : (q13' = q13-1);
  [] q14 > 0 -> mu1 : (q14' = q14-1);
  [] q15 > 0 -> mu1 : (q15' = q15-1);
  [] q16 > 0 -> mu1 : (q16' = q16-1);
  [] q17 > 0 -> mu1 : (q17' = q17-1);
  [] q18 > 0 -> mu1 : (q18' = q18-1);

  [] q21 > 0 -> mu2 : (q21' = q21-1);
  [] q22 > 0 -> mu2 : (q22' = q22-1);
  [] q23 > 0 -> mu2 : (q23' = q23-1);
  [] q24 > 0 -> mu2 : (q24' = q24-1);
  [] q25 > 0 -> mu2 : (q25' = q25-1);
  [] q26 > 0 -> mu2 : (q26' = q26-1);
  [] q27 > 0 -> mu2 : (q27' = q27-1);
  [] q28 > 0 -> mu2 : (q28' = q28-1);

  [] q31 > 0 -> mu3 : (q31' = q31-1);
  [] q32 > 0 -> mu3 : (q32' = q32-1);
  [] q33 > 0 -> mu3 : (q33' = q33-1);
  [] q34 > 0 -> mu3 : (q34' = q34-1);
  [] q35 > 0 -> mu3 : (q35' = q35-1);
  [] q36 > 0 -> mu3 : (q36' = q36-1);
  [] q37 > 0 -> mu3 : (q37' = q37-1);
  [] q38 > 0 -> mu3 : (q38' = q38-1);

  [] true -> lambda :
    (q11' =
         q11 <= q11 & q11 <= q12 & q11 <= q13 & q11 <= q14 &
         q11 <= q15 & q11 <= q16 & q11 <= q17 & q11 <= q18 &
         q11 <= q21 & q11 <= q22 & q11 <= q23 & q11 <= q24 &
         q11 <= q25 & q11 <= q26 & q11 <= q27 & q11 <= q28 &
         q11 <= q31 & q11 <= q32 & q11 <= q33 & q11 <= q34 &
         q11 <= q35 & q11 <= q36 & q11 <= q37 & q11 <= q38 &
         q11 < Q ? q11+1 : q11 ) &
    (q12' =
         q12 <  q11 & q12 <= q12 & q12 <= q13 & q12 <= q14 &
         q12 <= q15 & q12 <= q16 & q12 <= q17 & q12 <= q18 &
         q12 <= q21 & q12 <= q22 & q12 <= q23 & q12 <= q24 &
         q12 <= q25 & q12 <= q26 & q12 <= q27 & q12 <= q28 &
         q12 <= q31 & q12 <= q32 & q12 <= q33 & q12 <= q34 &
         q12 <= q35 & q12 <= q36 & q12 <= q37 & q12 <= q38 &
         q12 < Q ? q12+1 : q12 ) &
    (q13' =
         q13 <  q11 & q13 <  q12 & q13 <= q13 & q13 <= q14 &
         q13 <= q15 & q13 <= q16 & q13 <= q17 & q13 <= q18 &
         q13 <= q21 & q13 <= q22 & q13 <= q23 & q13 <= q24 &
         q13 <= q25 & q13 <= q26 & q13 <= q27 & q13 <= q28 &
         q13 <= q31 & q13 <= q32 & q13 <= q33 & q13 <= q34 &
         q13 <= q35 & q13 <= q36 & q13 <= q37 & q13 <= q38 &
         q13 < Q ? q13+1 : q13 ) &
    (q14' =
         q14 <  q11 & q14 <  q12 & q14 <  q13 & q14 <= q14 &
         q14 <= q15 & q14 <= q16 & q14 <= q17 & q14 <= q18 &
         q14 <= q21 & q14 <= q22 & q14 <= q23 & q14 <= q24 &
         q14 <= q25 & q14 <= q26 & q14 <= q27 & q14 <= q28 &
         q14 <= q31 & q14 <= q32 & q14 <= q33 & q14 <= q34 &
         q14 <= q35 & q14 <= q36 & q14 <= q37 & q14 <= q38 &
         q14 < Q ? q14+1 : q14 ) &
    (q15' =
         q15 <  q11 & q15 <  q12 & q15 <  q13 & q15 <  q14 &
         q15 <= q15 & q15 <= q16 & q15 <= q17 & q15 <= q18 &
         q15 <= q21 & q15 <= q22 & q15 <= q23 & q15 <= q24 &
         q15 <= q25 & q15 <= q26 & q15 <= q27 & q15 <= q28 &
```

```
         q15 <= q31 & q15 <= q32 & q15 <= q33 & q15 <= q34 &
         q15 <= q35 & q15 <= q36 & q15 <= q37 & q15 <= q38 &
         q15 < Q ? q15+1 : q15 ) &
(q16' =
         q16 <  q11 & q16 <  q12 & q16 <  q13 & q16 <  q14 &
         q16 <  q15 & q16 <= q16 & q16 <= q17 & q16 <= q18 &
         q16 <= q21 & q16 <= q22 & q16 <= q23 & q16 <= q24 &
         q16 <= q25 & q16 <= q26 & q16 <= q27 & q16 <= q28 &
         q16 <= q31 & q16 <= q32 & q16 <= q33 & q16 <= q34 &
         q16 <= q35 & q16 <= q36 & q16 <= q37 & q16 <= q38 &
         q16 < Q ? q16+1 : q16 ) &
(q17' =
         q17 <  q11 & q17 <  q12 & q17 <  q13 & q17 <  q14 &
         q17 <  q15 & q17 <  q16 & q17 <= q17 & q17 <= q18 &
         q17 <= q21 & q17 <= q22 & q17 <= q23 & q17 <= q24 &
         q17 <= q25 & q17 <= q26 & q17 <= q27 & q17 <= q28 &
         q17 <= q31 & q17 <= q32 & q17 <= q33 & q17 <= q34 &
         q17 <= q35 & q17 <= q36 & q17 <= q37 & q17 <= q38 &
         q17 < Q ? q17+1 : q17 ) &
(q18' =
         q18 <  q11 & q18 <  q12 & q18 <  q13 & q18 <  q14 &
         q18 <  q15 & q18 <  q16 & q18 <  q17 & q18 <= q18 &
         q18 <= q21 & q18 <= q22 & q18 <= q23 & q18 <= q24 &
         q18 <= q25 & q18 <= q26 & q18 <= q27 & q18 <= q28 &
         q18 <= q31 & q18 <= q32 & q18 <= q33 & q18 <= q34 &
         q18 <= q35 & q18 <= q36 & q18 <= q37 & q18 <= q38 &
         q18 < Q ? q18+1 : q18 ) &
(q21' =
         q21 <  q11 & q21 <  q12 & q21 <  q13 & q21 <  q14 &
         q21 <  q15 & q21 <  q16 & q21 <  q17 & q21 <  q18 &
         q21 <= q21 & q21 <= q22 & q21 <= q23 & q21 <= q24 &
         q21 <= q25 & q21 <= q26 & q21 <= q27 & q21 <= q28 &
         q21 <= q31 & q21 <= q32 & q21 <= q33 & q21 <= q34 &
         q21 <= q35 & q21 <= q36 & q21 <= q37 & q21 <= q38 &
         q21 < Q ? q21+1 : q21 ) &

(q22' =
         q22 <  q11 & q22 <  q12 & q22 <  q13 & q22 <  q14 &
         q22 <  q15 & q22 <  q16 & q22 <  q17 & q22 <  q18 &
         q22 <  q21 & q22 <= q22 & q22 <= q23 & q22 <= q24 &
         q22 <= q25 & q22 <= q26 & q22 <= q27 & q22 <= q28 &
         q22 <= q31 & q22 <= q32 & q22 <= q33 & q22 <= q34 &
         q22 <= q35 & q22 <= q36 & q22 <= q37 & q22 <= q38 &
         q22 < Q ? q22+1 : q22 ) &
(q23' =
         q23 <  q11 & q23 <  q12 & q23 <  q13 & q23 <  q14 &
         q23 <  q15 & q23 <  q16 & q23 <  q17 & q23 <  q18 &
         q23 <  q21 & q23 <  q22 & q23 <= q23 & q23 <= q24 &
         q23 <= q25 & q23 <= q26 & q23 <= q27 & q23 <= q28 &
         q23 <= q31 & q23 <= q32 & q23 <= q33 & q23 <= q34 &
         q23 <= q35 & q23 <= q36 & q23 <= q37 & q23 <= q38 &
         q23 < Q ? q23+1 : q23 ) &
(q24' =
         q24 <  q11 & q24 <  q12 & q24 <  q13 & q24 <  q14 &
```

```
     q24 <  q15 & q24 <  q16 & q24 <  q17 & q24 <  q18 &
     q24 <  q21 & q24 <  q22 & q24 <  q23 & q24 <= q24 &
     q24 <= q25 & q24 <= q26 & q24 <= q27 & q24 <= q28 &
     q24 <= q31 & q24 <= q32 & q24 <= q33 & q24 <= q34 &
     q24 <= q35 & q24 <= q36 & q24 <= q37 & q24 <= q38 &
     q24 < Q ? q24+1 : q24 ) &
(q25' =
     q25 <  q11 & q25 <  q12 & q25 <  q13 & q25 <  q14 &
     q25 <  q15 & q25 <  q16 & q25 <  q17 & q25 <  q18 &
     q25 <  q21 & q25 <  q22 & q25 <  q23 & q25 <  q24 &
     q25 <= q25 & q25 <= q26 & q25 <= q27 & q25 <= q28 &
     q25 <= q31 & q25 <= q32 & q25 <= q33 & q25 <= q34 &
     q25 <= q35 & q25 <= q36 & q25 <= q37 & q25 <= q38 &
     q25 < Q ? q25+1 : q25 ) &
(q26' =
     q26 <  q11 & q26 <  q12 & q26 <  q13 & q26 <  q14 &
     q26 <  q15 & q26 <  q16 & q26 <  q17 & q26 <  q18 &
     q26 <  q21 & q26 <  q22 & q26 <  q23 & q26 <  q24 &
     q26 <  q25 & q26 <= q26 & q26 <= q27 & q26 <= q28 &
     q26 <= q31 & q26 <= q32 & q26 <= q33 & q26 <= q34 &
     q26 <= q35 & q26 <= q36 & q26 <= q37 & q26 <= q38 &
     q26 < Q ? q26+1 : q26 ) &
(q27' =
     q27 <  q11 & q27 <  q12 & q27 <  q13 & q27 <  q14 &
     q27 <  q15 & q27 <  q16 & q27 <  q17 & q27 <  q18 &
     q27 <  q21 & q27 <  q22 & q27 <  q23 & q27 <  q24 &
     q27 <  q25 & q27 <  q26 & q27 <= q27 & q27 <= q28 &
     q27 <= q31 & q27 <= q32 & q27 <= q33 & q27 <= q34 &
     q27 <= q35 & q27 <= q36 & q27 <= q37 & q27 <= q38 &
     q27 < Q ? q27+1 : q27 ) &
(q28' =
     q28 <  q11 & q28 <  q12 & q28 <  q13 & q28 <  q14 &
     q28 <  q15 & q28 <  q16 & q28 <  q17 & q28 <  q18 &
     q28 <  q21 & q28 <  q22 & q28 <  q23 & q28 <  q24 &
     q28 <  q25 & q28 <  q26 & q28 <  q27 & q28 <= q28 &
     q28 <= q31 & q28 <= q32 & q28 <= q33 & q28 <= q34 &
     q28 <= q35 & q28 <= q36 & q28 <= q37 & q28 <= q38 &
     q28 < Q ? q28+1 : q28 ) &

(q31' =
     q31 <  q11 & q31 <  q12 & q31 <  q13 & q31 <  q14 &
     q31 <  q15 & q31 <  q16 & q31 <  q17 & q31 <  q18 &
     q31 <  q21 & q31 <  q22 & q31 <  q23 & q31 <  q24 &
     q31 <  q25 & q31 <  q26 & q31 <  q27 & q31 <  q28 &
     q31 <= q31 & q31 <= q32 & q31 <= q33 & q31 <= q34 &
     q31 <= q35 & q31 <= q36 & q31 <= q37 & q31 <= q38 &
     q31 < Q ? q31+1 : q31 ) &
(q32' =
     q32 <  q11 & q32 <  q12 & q32 <  q13 & q32 <  q14 &
     q32 <  q15 & q32 <  q16 & q32 <  q17 & q32 <  q18 &
     q32 <  q21 & q32 <  q22 & q32 <  q23 & q32 <  q24 &
     q32 <  q25 & q32 <  q26 & q32 <  q27 & q32 <  q28 &
     q32 <  q31 & q32 <= q32 & q32 <= q33 & q32 <= q34 &
     q32 <= q35 & q32 <= q36 & q32 <= q37 & q32 <= q38 &
```

```
         q32 < Q ? q32+1 : q32 ) &
    (q33' =
         q33 <  q11 & q33 <  q12 & q33 <  q13 & q33 <  q14 &
         q33 <  q15 & q33 <  q16 & q33 <  q17 & q33 <  q18 &
         q33 <  q21 & q33 <  q22 & q33 <  q23 & q33 <  q24 &
         q33 <  q25 & q33 <  q26 & q33 <  q27 & q33 <  q28 &
         q33 <  q31 & q33 <  q32 & q33 <= q33 & q33 <= q34 &
         q33 <= q35 & q33 <= q36 & q33 <= q37 & q33 <= q38 &
         q33 < Q ? q33+1 : q33 ) &
    (q34' =
         q34 <  q11 & q34 <  q12 & q34 <  q13 & q34 <  q14 &
         q34 <  q15 & q34 <  q16 & q34 <  q17 & q34 <  q18 &
         q34 <  q21 & q34 <  q22 & q34 <  q23 & q34 <  q24 &
         q34 <  q25 & q34 <  q26 & q34 <  q27 & q34 <  q28 &
         q34 <  q31 & q34 <  q32 & q34 <  q33 & q34 <= q34 &
         q34 <= q35 & q34 <= q36 & q34 <= q37 & q34 <= q38 &
         q34 < Q ? q34+1 : q34 ) &
    (q35' =
         q35 <  q11 & q35 <  q12 & q35 <  q13 & q35 <  q14 &
         q35 <  q15 & q35 <  q16 & q35 <  q17 & q35 <  q18 &
         q35 <  q21 & q35 <  q22 & q35 <  q23 & q35 <  q24 &
         q35 <  q25 & q35 <  q26 & q35 <  q27 & q35 <  q28 &
         q35 <  q31 & q35 <  q32 & q35 <  q33 & q35 <  q34 &
         q35 <= q35 & q35 <= q36 & q35 <= q37 & q35 <= q38 &
         q35 < Q ? q35+1 : q35 ) &
    (q36' =
         q36 <  q11 & q36 <  q12 & q36 <  q13 & q36 <  q14 &
         q36 <  q15 & q36 <  q16 & q36 <  q17 & q36 <  q18 &
         q36 <  q21 & q36 <  q22 & q36 <  q23 & q36 <  q24 &
         q36 <  q25 & q36 <  q26 & q36 <  q27 & q36 <  q28 &
         q36 <  q31 & q36 <  q32 & q36 <  q33 & q36 <  q34 &
         q36 <  q35 & q36 <= q36 & q36 <= q37 & q36 <= q38 &
         q36 < Q ? q36+1 : q36 ) &
    (q37' =
         q37 <  q11 & q37 <  q12 & q37 <  q13 & q37 <  q14 &
         q37 <  q15 & q37 <  q16 & q37 <  q17 & q37 <  q18 &
         q37 <  q21 & q37 <  q22 & q37 <  q23 & q37 <  q24 &
         q37 <  q25 & q37 <  q26 & q37 <  q27 & q37 <  q28 &
         q37 <  q31 & q37 <  q32 & q37 <  q33 & q37 <  q34 &
         q37 <  q35 & q37 <  q36 & q37 <= q37 & q37 <= q38 &
         q37 < Q ? q37+1 : q37 ) &
    (q38' =
         q38 <  q11 & q38 <  q12 & q38 <  q13 & q38 <  q14 &
         q38 <  q15 & q38 <  q16 & q38 <  q17 & q38 <  q18 &
         q38 <  q21 & q38 <  q22 & q38 <  q23 & q38 <  q24 &
         q38 <  q25 & q38 <  q26 & q38 <  q27 & q38 <  q28 &
         q38 <  q31 & q38 <  q32 & q38 <  q33 & q38 <  q34 &
         q38 <  q35 & q38 <  q36 & q38 <  q37 & q38 <= q38 &
         q38 < Q ? q38+1 : q38 );
endmodule

// ------------------------------------------------------------------
// system rewards
// ------------------------------------------------------------------
```

```
rewards "stime"
    !(q11 = Q & q12 = Q & q13 = Q & q14 = Q &
      q15 = Q & q16 = Q & q17 = Q & q18 = Q &
      q21 = Q & q22 = Q & q23 = Q & q24 = Q &
      q25 = Q & q26 = Q & q27 = Q & q28 = Q &
      q31 = Q & q32 = Q & q33 = Q & q34 = Q &
      q35 = Q & q36 = Q & q37 = Q & q38 = Q) :
    min(q11, q12, q13, q14, q15, q16, q17, q18) <=
      min(q21, q22, q23, q24, q25, q26, q27, q28,
          q31, q32, q33, q34, q35, q36, q37, q38) ? (1/mu1) :
    min(q21, q22, q23, q24, q25, q26, q27, q28) <=
      min(q31, q32, q33, q34, q35, q36, q37, q38) ? (1/mu2) : (1/mu3);
endrewards

rewards "qload"
  true : q11+q12+q13+q14+q15+q16+q17+q18+
         q21+q22+q23+q24+q25+q26+q27+q28+
         q31+q32+q33+q34+q35+q36+q37+q38;
endrewards


// -----------------------------------------------------------------
// InfiniteSeparate.props
// -----------------------------------------------------------------

// rejection probability
"reject": S=? [
      q11 = Q & q12 = Q & q13 = Q & q14 = Q &
      q15 = Q & q16 = Q & q17 = Q & q18 = Q &
      q21 = Q & q22 = Q & q23 = Q & q24 = Q &
      q25 = Q & q26 = Q & q27 = Q & q28 = Q &
      q31 = Q & q32 = Q & q33 = Q & q34 = Q &
      q35 = Q & q36 = Q & q37 = Q & q38 = Q ] ;

// mean service time
"stime": R{"stime"}=? [ S ] ;

// response time (of accepted jobs)
"qload":  R{"qload"}=? [ S ];
"rtime":  "qload"/(lambda*(1-"reject"));

// waiting time (of accepted jobs)
"wtime":  "rtime"-"stime";
```

A.2. Separate Queue (3 · 4 Servers)

```
// -----------------------------------------------------------------
// InfiniteSeparateHP.prism
// A scheduling model for computational clusters.
//
// The model is the "separate queue" model with
// "high-performance priority" described in
```

```
//
//   Tien v. Do, Binh t. Vu, Xuan T. Tran, Anh P. Nguyen:
//   "A generalized model for investigating scheduling schemes in
//   computational clusters", Simulation Modelling Practice and
//   Theory, 37 (2013), 30-42.
//
// Authors: Berczes Tamas <berczes.tamas@inf.unideb.hu> and
// Wolfgang Schreiner <Wolfgang.Schreiner@risc.jku.at>
//
// Copyright (C) 2014 Department of Informatics Systems and Networks,
// University of Debrecen, Debrecen, Hungary (http://irh.inf.unideb.hu)
// and Research Institute for Symbolic Computation, Johannes Kepler
// University, Linz, Austria (http://www.risc.jku.at)
// -----------------------------------------------------------------

// checking parameters: "sparse" (1GB), "Jacobi", epsilon=0.01

// Q=1 gives rejection probability from 2E-4 (U0=0.5) to 0.08 (U0=0.9)
// so the results for high U0 are not very accurate

// continuous time markov chain (ctmc) model
ctmc

// -----------------------------------------------------------------
// system parameters
// -----------------------------------------------------------------

// system utilization (50%, 60%, 70%, 80%, 90%)
const double U0;

// arrival rates
const double lambda = U0*4*(rp1+rp2+rp3);

// queue size
const int Q = 3;

// number of server classes and number of servers per class
const int K = 3;
const int M = 4;

// ranking based on performance
const double rp1 = 1.0;
const double rp2 = 0.82;
const double rp3 = 0.43;

// ranking based on energy efficiency
const double re1 = 0.64;
const double re2 = 0.66;
const double re3 = 1.0;

// service rates according to chosen ranking
const double mu1 = rp1;
const double mu2 = rp2;
const double mu3 = rp3;
```

```
// -------------------------------------------------------------------
// system model
// -------------------------------------------------------------------

module System
  q11: [0..Q] init 0;
  q12: [0..Q] init 0;
  q13: [0..Q] init 0;
  q14: [0..Q] init 0;

  q21: [0..Q] init 0;
  q22: [0..Q] init 0;
  q23: [0..Q] init 0;
  q24: [0..Q] init 0;

  q31: [0..Q] init 0;
  q32: [0..Q] init 0;
  q33: [0..Q] init 0;
  q34: [0..Q] init 0;

  [] q11 > 0 -> mu1 : (q11' = q11-1);
  [] q12 > 0 -> mu1 : (q12' = q12-1);
  [] q13 > 0 -> mu1 : (q13' = q13-1);
  [] q14 > 0 -> mu1 : (q14' = q14-1);

  [] q21 > 0 -> mu2 : (q21' = q21-1);
  [] q22 > 0 -> mu2 : (q22' = q22-1);
  [] q23 > 0 -> mu2 : (q23' = q23-1);
  [] q24 > 0 -> mu2 : (q24' = q24-1);

  [] q31 > 0 -> mu3 : (q31' = q31-1);
  [] q32 > 0 -> mu3 : (q32' = q32-1);
  [] q33 > 0 -> mu3 : (q33' = q33-1);
  [] q34 > 0 -> mu3 : (q34' = q34-1);

  [] true -> lambda :
    (q11' =
         q11 <= q11 & q11 <= q12 & q11 <= q13 & q11 <= q14 &
         q11 <= q21 & q11 <= q22 & q11 <= q23 & q11 <= q24 &
         q11 <= q31 & q11 <= q32 & q11 <= q33 & q11 <= q34 &
         q11 < Q ? q11+1 : q11 ) &
    (q12' =
         q12 <  q11 & q12 <= q12 & q12 <= q13 & q12 <= q14 &
         q12 <= q21 & q12 <= q22 & q12 <= q23 & q12 <= q24 &
         q12 <= q31 & q12 <= q32 & q12 <= q33 & q12 <= q34 &
         q12 < Q ? q12+1 : q12 ) &
    (q13' =
         q13 <  q11 & q13 <  q12 & q13 <= q13 & q13 <= q14 &
         q13 <= q21 & q13 <= q22 & q13 <= q23 & q13 <= q24 &
         q13 <= q31 & q13 <= q32 & q13 <= q33 & q13 <= q34 &
         q13 < Q ? q13+1 : q13 ) &
    (q14' =
         q14 <  q11 & q14 <  q12 & q14 <  q13 & q14 <= q14 &
```

```
          q14 <= q21 & q14 <= q22 & q14 <= q23 & q14 <= q24 &
          q14 <= q31 & q14 <= q32 & q14 <= q33 & q14 <= q34 &
          q14 < Q ? q14+1 : q14 ) &

     (q21' =
          q21 <  q11 & q21 <  q12 & q21 <  q13 & q21 <  q14 &
          q21 <= q21 & q21 <= q22 & q21 <= q23 & q21 <= q24 &
          q21 <= q31 & q21 <= q32 & q21 <= q33 & q21 <= q34 &
          q21 < Q ? q21+1 : q21 ) &

     (q22' =
          q22 <  q11 & q22 <  q12 & q22 <  q13 & q22 <  q14 &
          q22 <  q21 & q22 <= q22 & q22 <= q23 & q22 <= q24 &
          q22 <= q31 & q22 <= q32 & q22 <= q33 & q22 <= q34 &
          q22 < Q ? q22+1 : q22 ) &
     (q23' =
          q23 <  q11 & q23 <  q12 & q23 <  q13 & q23 <  q14 &
          q23 <  q21 & q23 <  q22 & q23 <= q23 & q23 <= q24 &
          q23 <= q31 & q23 <= q32 & q23 <= q33 & q23 <= q34 &
          q23 < Q ? q23+1 : q23 ) &
     (q24' =
          q24 <  q11 & q24 <  q12 & q24 <  q13 & q24 <  q14 &
          q24 <  q21 & q24 <  q22 & q24 <  q23 & q24 <= q24 &
          q24 <= q31 & q24 <= q32 & q24 <= q33 & q24 <= q34 &
          q24 < Q ? q24+1 : q24 ) &

     (q31' =
          q31 <  q11 & q31 <  q12 & q31 <  q13 & q31 <  q14 &
          q31 <  q21 & q31 <  q22 & q31 <  q23 & q31 <  q24 &
          q31 <= q31 & q31 <= q32 & q31 <= q33 & q31 <= q34 &
          q31 < Q ? q31+1 : q31 ) &
     (q32' =
          q32 <  q11 & q32 <  q12 & q32 <  q13 & q32 <  q14 &
          q32 <  q21 & q32 <  q22 & q32 <  q23 & q32 <  q24 &
          q32 <  q31 & q32 <= q32 & q32 <= q33 & q32 <= q34 &
          q32 < Q ? q32+1 : q32 ) &
     (q33' =
          q33 <  q11 & q33 <  q12 & q33 <  q13 & q33 <  q14 &
          q33 <  q21 & q33 <  q22 & q33 <  q23 & q33 <  q24 &
          q33 <  q31 & q33 <  q32 & q33 <= q33 & q33 <= q34 &
          q33 < Q ? q33+1 : q33 ) &
     (q34' =
          q34 <  q11 & q34 <  q12 & q34 <  q13 & q34 <  q14 &
          q34 <  q21 & q34 <  q22 & q34 <  q23 & q34 <  q24 &
          q34 <  q31 & q34 <  q32 & q34 <  q33 & q34 <= q34 &
          q34 < Q ? q34+1 : q34 );
endmodule

// ------------------------------------------------------------------
// system rewards
// ------------------------------------------------------------------

rewards "stime"
     !(q11 = Q & q12 = Q & q13 = Q & q14 = Q &
      q21 = Q & q22 = Q & q23 = Q & q24 = Q &
      q31 = Q & q32 = Q & q33 = Q & q34 = Q) :
    min(q11, q12, q13, q14) <=
      min(q21, q22, q23, q24,
          q31, q32, q33, q34) ? (1/mu1) :
    min(q21, q22, q23, q24) <=
      min(q31, q32, q33, q34) ? (1/mu2) : (1/mu3);
endrewards

rewards "qload"
  true : q11+q12+q13+q14+
         q21+q22+q23+q24+
         q31+q32+q33+q34;
endrewards


// ---------------------------------------------------------------------
// InfiniteSeparate.props
// ---------------------------------------------------------------------

// rejection probability
"reject": S=? [
      q11 = Q & q12 = Q & q13 = Q & q14 = Q &
      q21 = Q & q22 = Q & q23 = Q & q24 = Q &
      q31 = Q & q32 = Q & q33 = Q & q34 = Q ] ;

// mean service time (of accepted jobs)
"stime": R{"stime"}=? [ S ] ;
"stime0" : "stime"/(1-"reject");

// response time (of accepted jobs)
"qload":  R{"qload"}=? [ S ];
"rtime":  "qload"/lambda;
"rtime0": "rtime"/(1-"reject");

// waiting time (of accepted jobs)
"wtime":  ("rtime"-"stime")/(1-"reject");
```

A.3. Class Queue

```
// ---------------------------------------------------------------------
// InfiniteClassHP.prism
// A scheduling model for computational clusters.
//
// The model is the "class queue" model with
// "high-performance priority" described in
//
//   Tien v. Do, Binh t. Vu, Xuan T. Tran, Anh P. Nguyen:
//   "A generalized model for investigating scheduling schemes in
//   computational clusters", Simulation Modelling Practice and
//   Theory, 37 (2013), 30-42.
//
// Authors: Berczes Tamas <berczes.tamas@inf.unideb.hu> and
// Wolfgang Schreiner <Wolfgang.Schreiner@risc.jku.at>
//
// Copyright (C) 2014 Department of Informatics Systems and Networks,
// University of Debrecen, Debrecen, Hungary (http://irh.inf.unideb.hu)
// and Research Institute for Symbolic Computation, Johannes Kepler
// University, Linz, Austria (http://www.risc.jku.at)
// ------------------------------------------------------------------

// checking parameters: "sparse", "Jacobi", epsilon=0.001

// continuous time markov chain (ctmc) model
ctmc

// ------------------------------------------------------------------
// system parameters
// ------------------------------------------------------------------

// system utilization (50%, 60%, 70%, 80%, 90%)
const double U0;

// arrival rates
const double lambda = U0*8*(rp1+rp2+rp3);

// queue size
const int Q = 24;

// number of server classes and number of servers per class
const int K = 3;
const int M = 8;

// ranking based on performance
const double rp1 = 1.0;
const double rp2 = 0.82;
const double rp3 = 0.43;

// ranking based on energy efficiency
const double re1 = 0.64;
const double re2 = 0.66;
const double re3 = 1.0;

// service rates according to chosen ranking
const double mu1 = rp1;
const double mu2 = rp2;
const double mu3 = rp3;

// ------------------------------------------------------------------
// system model
// ------------------------------------------------------------------

module System
  q1: [0..Q] init 0;
  q2: [0..Q] init 0;
  q3: [0..Q] init 0;
  [] q1 > 0 -> min(M,q1)*mu1 : (q1' = q1-1);
  [] q2 > 0 -> min(M,q2)*mu2 : (q2' = q2-1);
  [] q3 > 0 -> min(M,q3)*mu3 : (q3' = q3-1);

  [] true -> lambda :
    (q1' =
       (q1 < M | (q2 >= M & q3 >= M & q1 <= q2 & q1 <= q3)) & q1 < Q ?
       q1+1 : q1) &
    (q2' =
       q1 >= M & (q2 < M | (q3 >= M & q2 < q1 & q2 <= q3)) & q2 < Q ?
       q2+1 : q2) &
    (q3' =
       q1 >= M & q2 >= M & (q3 < M | (q3 < q1 & q3 < q2)) & q3 < Q ?
       q3+1 : q3);
endmodule

// -----------------------------------------------------------------
// system rewards
// -----------------------------------------------------------------

rewards "stime"
  !(q1 = Q & q2 = Q & q3 = Q) :
  q1 < M ? (1/mu1) :
  q2 < M ? (1/mu2) :
  q3 < M ? (1/mu3) :
  q1 <= q2 & q1 <= q3 ? (1/mu1) :
             q2 <= q3 ? (1/mu2) :
                        (1/mu3) ;
endrewards

rewards "qload"
  true : q1+q2+q3;
endrewards


// -----------------------------------------------------------------
// InfiniteClass.props
// -----------------------------------------------------------------

// rejection probability
"reject": S=? [ q1 = Q & q2 = Q & q3 = Q ] ;

// probability that all servers are busy
"busy": S=? [ q1 >= M ] ;

// mean service time
"stime": R{"stime"}=? [ S ] ;

// response time (of accepted jobs)
"qload":  R{"qload"}=? [ S ];
"rtime":  "qload"/(lambda*(1-"reject"));

// waiting time (of accepted jobs)
"wtime":  "rtime"-"stime";
```

A.4. Common Queue

```
// ------------------------------------------------------------------
// InfiniteCommonHP.prism
// A scheduling model for computational clusters.
//
// The model is the "common queue" model with
// "high-performance priority" described in
//
//   Tien v. Do, Binh t. Vu, Xuan T. Tran, Anh P. Nguyen:
//   "A generalized model for investigating scheduling schemes in
//   computational clusters", Simulation Modelling Practice and
//   Theory, 37 (2013), 30-42.
//
// Authors: Berczes Tamas <berczes.tamas@inf.unideb.hu> and
// Wolfgang Schreiner <Wolfgang.Schreiner@risc.jku.at>
//
// Copyright (C) 2014 Department of Informatics Systems and Networks,
// University of Debrecen, Debrecen, Hungary (http://irh.inf.unideb.hu)
// and Research Institute for Symbolic Computation, Johannes Kepler
// University, Linz, Austria (http://www.risc.jku.at)
// ------------------------------------------------------------------

// checking parameters: "sparse", "Jacobi", epsilon=0.001

// continuous time markov chain (ctmc) model
ctmc

// ------------------------------------------------------------------
// system parameters
// ------------------------------------------------------------------

// system utilization (50%, 60%, 70%, 80%, 90%)
const double U0;

// arrival rates
const double lambda = U0*8*(rp1+rp2+rp3);

// queue size
const int Q = 32;

// number of server classes and number of servers per class
const int K = 3;
const int M = 8;

// ranking based on performance
const double rp1 = 1.0;
const double rp2 = 0.82;
const double rp3 = 0.43;

// ranking based on energy efficiency
```

```
const double re1 = 0.64;
const double re2 = 0.66;
const double re3 = 1.0;

// service rates according to chosen ranking
const double mu1 = rp1;
const double mu2 = rp2;
const double mu3 = rp3;

// ------------------------------------------------------------------
// system model
// ------------------------------------------------------------------

module System
  q: [0..Q] init 0;

  q1: [0..M] init 0;
  q2: [0..M] init 0;
  q3: [0..M] init 0;

  [] q1 > 0 -> q1*mu1 : (q1' = q1-1);
  [] q1 < M & q > 0 -> (M-q1)*mu1 : (q' = q-1);

  [] q2 > 0 -> q2*mu2 : (q2' = q2-1);
  [] q2 < M & q > 0 -> (M-q2)*mu2 : (q' = q-1);

  [] q3 > 0 -> q3*mu3 : (q3' = q3-1);
  [] q3 < M & q > 0 -> (M-q3)*mu3 : (q' = q-1);

  [] true -> lambda :
    (q' =
      q1 = M & q2 = M & q3 = M & q < Q ?
        q+1 : q) &
    (q1' =
      q1 < M ?
        q1+1 : q1) &
    (q2' =
      q1 = M & q2 < M ?
        q2+1 : q2) &
    (q3' =
      q1 = M & q2 = M & q3 < M ?
        q3+1 : q3);
endmodule

// ------------------------------------------------------------------
// system rewards
// ------------------------------------------------------------------

rewards "stime"
  !(q = Q & q1 = M & q2 = M & q3 = M) :
  q1 < M ? (1/mu1) :
  q2 < M ? (1/mu2) :
  q3 < M ? (1/mu3) :
    (1/mu1); // higher value
    // (1/mu2); // medium value
    // (1/mu3); // lower value
endrewards

rewards "qload"
  true : q+q1+q2+q3;
endrewards


// ------------------------------------------------------------------
// InfiniteCommon.props
// ------------------------------------------------------------------

// rejection probability
"reject": S=? [ q = Q & q1 = M & q2 = M & q3 = M ] ;

// mean service time
"stime": (R{"stime"}=? [ S ])/(1-"reject") ;

// mean response time
"rtime": (R{"qload"}=? [ S ])/(lambda*(1-"reject"));

// waiting time
"wtime":  max(0, "rtime"-"stime");
```

B. The Finite Source Models

B.1. Separate Queue ($3 \cdot 4$ Servers)

```
// ------------------------------------------------------------------
// FiniteSeparateHP.prism
// A scheduling model for computational clusters.
//
// The model is a finite source version of the "separate queue" model with
// "high-performance priority" described in
//
//   Tien v. Do, Binh t. Vu, Xuan T. Tran, Anh P. Nguyen:
//   "A generalized model for investigating scheduling schemes in
//   computational clusters", Simulation Modelling Practice and
//   Theory, 37 (2013), 30-42.
//
// Authors: Berczes Tamas <berczes.tamas@inf.unideb.hu> and
// Wolfgang Schreiner <Wolfgang.Schreiner@risc.jku.at>
//
// Copyright (C) 2014 Department of Informatics Systems and Networks,
// University of Debrecen, Debrecen, Hungary (http://irh.inf.unideb.hu)
// and Research Institute for Symbolic Computation, Johannes Kepler
// University, Linz, Austria (http://www.risc.jku.at)
// ------------------------------------------------------------------

// check with sparse (1GB), "Jacobi", epsilon=0.05
// continuous time markov chain (ctmc) model
ctmc

// ---------------------------------------------------------------------
// system parameters
// ---------------------------------------------------------------------

// arrival rates
const double lambda;

// queue size
const int Q = 3;

// population size
const int N = 60;

// number of server classes and number of servers per class
const int K = 3;
const int M = 4;

// ranking based on performance
const double rp1 = 1.0;
const double rp2 = 0.82;
const double rp3 = 0.43;

// ranking based on energy efficiency
const double re1 = 0.64;
const double re2 = 0.66;
const double re3 = 1.0;

// service rates according to chosen ranking
const double mu1 = rp1;
const double mu2 = rp2;
const double mu3 = rp3;

// ---------------------------------------------------------------------
// system model
// ---------------------------------------------------------------------

module Sources
  sources: [0..N] init N;
  [server]   sources > 0 -> lambda*sources : (sources' = sources-1);
  [source11] sources < N -> (sources' = sources+1);
  [source12] sources < N -> (sources' = sources+1);
  [source13] sources < N -> (sources' = sources+1);
  [source14] sources < N -> (sources' = sources+1);
  [source21] sources < N -> (sources' = sources+1);
  [source22] sources < N -> (sources' = sources+1);
  [source23] sources < N -> (sources' = sources+1);
  [source24] sources < N -> (sources' = sources+1);
  [source31] sources < N -> (sources' = sources+1);
  [source32] sources < N -> (sources' = sources+1);
  [source33] sources < N -> (sources' = sources+1);
  [source34] sources < N -> (sources' = sources+1);
```

```
endmodule

module Servers
  q11: [0..Q] init 0;
  q12: [0..Q] init 0;
  q13: [0..Q] init 0;
  q14: [0..Q] init 0;

  q21: [0..Q] init 0;
  q22: [0..Q] init 0;
  q23: [0..Q] init 0;
  q24: [0..Q] init 0;

  q31: [0..Q] init 0;
  q32: [0..Q] init 0;
  q33: [0..Q] init 0;
  q34: [0..Q] init 0;

  [source11] q11 > 0 -> mu1 : (q11' = q11-1);
  [source12] q12 > 0 -> mu1 : (q12' = q12-1);
  [source13] q13 > 0 -> mu1 : (q13' = q13-1);
  [source14] q14 > 0 -> mu1 : (q14' = q14-1);

  [source21] q21 > 0 -> mu2 : (q21' = q21-1);
  [source22] q22 > 0 -> mu2 : (q22' = q22-1);
  [source23] q23 > 0 -> mu2 : (q23' = q23-1);
  [source24] q24 > 0 -> mu2 : (q24' = q24-1);

  [source31] q31 > 0 -> mu3 : (q31' = q31-1);
  [source32] q32 > 0 -> mu3 : (q32' = q32-1);
  [source33] q33 > 0 -> mu3 : (q33' = q33-1);
  [source34] q34 > 0 -> mu3 : (q34' = q34-1);

  [server] !(q11 = Q & q12 = Q & q13 = Q & q14 = Q &
             q21 = Q & q22 = Q & q23 = Q & q24 = Q &
             q31 = Q & q32 = Q & q33 = Q & q34 = Q ) ->
    (q11' =
         q11 <= q11 & q11 <= q12 & q11 <= q13 & q11 <= q14 &
         q11 <= q21 & q11 <= q22 & q11 <= q23 & q11 <= q24 &
         q11 <= q31 & q11 <= q32 & q11 <= q33 & q11 <= q34 &
         q11 < Q ? q11+1 : q11 ) &
    (q12' =
         q12 <  q11 & q12 <= q12 & q12 <= q13 & q12 <= q14 &
         q12 <= q21 & q12 <= q22 & q12 <= q23 & q12 <= q24 &
         q12 <= q31 & q12 <= q32 & q12 <= q33 & q12 <= q34 &
         q12 < Q ? q12+1 : q12 ) &
    (q13' =
         q13 <  q11 & q13 <  q12 & q13 <= q13 & q13 <= q14 &
         q13 <= q21 & q13 <= q22 & q13 <= q23 & q13 <= q24 &
         q13 <= q31 & q13 <= q32 & q13 <= q33 & q13 <= q34 &
         q13 < Q ? q13+1 : q13 ) &
    (q14' =
         q14 <  q11 & q14 <  q12 & q14 <  q13 & q14 <= q14 &
         q14 <= q21 & q14 <= q22 & q14 <= q23 & q14 <= q24 &
```

```
          q14 <= q31 & q14 <= q32 & q14 <= q33 & q14 <= q34 &
          q14 < Q ? q14+1 : q14 ) &

     (q21' =
          q21 <  q11 & q21 <  q12 & q21 <  q13 & q21 <  q14 &
          q21 <= q21 & q21 <= q22 & q21 <= q23 & q21 <= q24 &
          q21 <= q31 & q21 <= q32 & q21 <= q33 & q21 <= q34 &
          q21 < Q ? q21+1 : q21 ) &

     (q22' =
          q22 <  q11 & q22 <  q12 & q22 <  q13 & q22 <  q14 &
          q22 <  q21 & q22 <= q22 & q22 <= q23 & q22 <= q24 &
          q22 <= q31 & q22 <= q32 & q22 <= q33 & q22 <= q34 &
          q22 < Q ? q22+1 : q22 ) &
     (q23' =
          q23 <  q11 & q23 <  q12 & q23 <  q13 & q23 <  q14 &
          q23 <  q21 & q23 <  q22 & q23 <= q23 & q23 <= q24 &
          q23 <= q31 & q23 <= q32 & q23 <= q33 & q23 <= q34 &
          q23 < Q ? q23+1 : q23 ) &
     (q24' =
          q24 <  q11 & q24 <  q12 & q24 <  q13 & q24 <  q14 &
          q24 <  q21 & q24 <  q22 & q24 <  q23 & q24 <= q24 &
          q24 <= q31 & q24 <= q32 & q24 <= q33 & q24 <= q34 &
          q24 < Q ? q24+1 : q24 ) &

     (q31' =
          q31 <  q11 & q31 <  q12 & q31 <  q13 & q31 <  q14 &
          q31 <  q21 & q31 <  q22 & q31 <  q23 & q31 <  q24 &
          q31 <= q31 & q31 <= q32 & q31 <= q33 & q31 <= q34 &
          q31 < Q ? q31+1 : q31 ) &
     (q32' =
          q32 <  q11 & q32 <  q12 & q32 <  q13 & q32 <  q14 &
          q32 <  q21 & q32 <  q22 & q32 <  q23 & q32 <  q24 &
          q32 <  q31 & q32 <= q32 & q32 <= q33 & q32 <= q34 &
          q32 < Q ? q32+1 : q32 ) &
     (q33' =
          q33 <  q11 & q33 <  q12 & q33 <  q13 & q33 <  q14 &
          q33 <  q21 & q33 <  q22 & q33 <  q23 & q33 <  q24 &
          q33 <  q31 & q33 <  q32 & q33 <= q33 & q33 <= q34 &
          q33 < Q ? q33+1 : q33 ) &
     (q34' =
          q34 <  q11 & q34 <  q12 & q34 <  q13 & q34 <  q14 &
          q34 <  q21 & q34 <  q22 & q34 <  q23 & q34 <  q24 &
          q34 <  q31 & q34 <  q32 & q34 <  q33 & q34 <= q34 &
          q34 < Q ? q34+1 : q34 );
endmodule

// ------------------------------------------------------------------
// system rewards
// ------------------------------------------------------------------

rewards "stime"
    !(q11 = Q & q12 = Q & q13 = Q & q14 = Q &
      q21 = Q & q22 = Q & q23 = Q & q24 = Q &
      q31 = Q & q32 = Q & q33 = Q & q34 = Q) :
    min(q11, q12, q13, q14) <=
      min(q21, q22, q23, q24,
          q31, q32, q33, q34) ? (1/mu1) :
    min(q21, q22, q23, q24) <=
      min(q31, q32, q33, q34) ? (1/mu2) : (1/mu3);
endrewards

rewards "sources"
  true : sources;
endrewards


// ------------------------------------------------------------------
// FiniteSeparate12.props
// ------------------------------------------------------------------

// rejection probability
"reject": S=? [
      q11 = Q & q12 = Q & q13 = Q & q14 = Q &
      q21 = Q & q22 = Q & q23 = Q & q24 = Q &
      q31 = Q & q32 = Q & q33 = Q & q34 = Q ] ;

// mean service time (not considering rejection)
"stime": R{"stime"}=? [ S ] ;

// mean service time
"stime0": "stime"/(1-"reject") ;

// number of currently not serviced sources
"sources": R{"sources"}=? [ S ];

// response time (not considering rejection)
"rtime":  (N/"sources"-1)/lambda;

// response time
"rtime0": "rtime"/(1-"reject");

// waiting time (not considering rejection)
"wtime":  "rtime"-"stime";

// waiting time
"wtime0":  "wtime"/(1-"reject");
```

B.2. Class Queue

```
// ------------------------------------------------------------------
// FiniteClassHP.prism
// A scheduling model for computational clusters.
//
// The model is a finite source version of the "class queue" model with
// "high-performance priority" described in
//
```

```
//   Tien v. Do, Binh t. Vu, Xuan T. Tran, Anh P. Nguyen:
//   "A generalized model for investigating scheduling schemes in
//   computational clusters", Simulation Modelling Practice and
//   Theory, 37 (2013), 30-42.
//
// Authors: Berczes Tamas <berczes.tamas@inf.unideb.hu> and
// Wolfgang Schreiner <Wolfgang.Schreiner@risc.jku.at>
//
// Copyright (C) 2014 Department of Informatics Systems and Networks,
// University of Debrecen, Debrecen, Hungary (http://irh.inf.unideb.hu)
// and Research Institute for Symbolic Computation, Johannes Kepler
// University, Linz, Austria (http://www.risc.jku.at)
// ----------------------------------------------------------------

// checking parameters: "sparse", "Jacobi", epsilon=0.001

// continuous time markov chain (ctmc) model
ctmc

// ----------------------------------------------------------------
// system parameters
// ----------------------------------------------------------------

// arrival rates
const double lambda;

// queue size
const int Q = 24;

// number of server classes and number of servers per class
const int K = 3;
const int M = 8;

// population size
const int N = 60;

// ranking based on performance
const double rp1 = 1.0;
const double rp2 = 0.82;
const double rp3 = 0.43;

// ranking based on energy efficiency
const double re1 = 0.64;
const double re2 = 0.66;
const double re3 = 1.0;

// service rates according to chosen ranking
const double mu1 = rp1;
const double mu2 = rp2;
const double mu3 = rp3;

// ----------------------------------------------------------------
// system model
// ----------------------------------------------------------------
```

```
module Sources
  sources: [0..N] init N;
  [server]  sources > 0 -> lambda*sources : (sources' = sources-1);
  [source1] sources < N -> (sources' = sources+1);
  [source2] sources < N -> (sources' = sources+1);
  [source3] sources < N -> (sources' = sources+1);
endmodule

module System
  q1: [0..Q] init 0;
  q2: [0..Q] init 0;
  q3: [0..Q] init 0;

  [source1] q1 > 0 -> min(M,q1)*mu1 : (q1' = q1-1);
  [source2] q2 > 0 -> min(M,q2)*mu2 : (q2' = q2-1);
  [source3] q3 > 0 -> min(M,q3)*mu3 : (q3' = q3-1);

  [server] !(q1 = Q & q2 = Q & q3 = Q) ->
    (q1' =
       (q1 < M | (q2 >= M & q3 >= M & q1 <= q2 & q1 <= q3)) & q1 < Q ?
       q1+1 : q1) &
    (q2' =
       q1 >= M & (q2 < M | (q3 >= M & q2 < q1 & q2 <= q3)) & q2 < Q ?
       q2+1 : q2) &
    (q3' =
       q1 >= M & q2 >= M & (q3 < M | (q3 < q1 & q3 < q2)) & q3 < Q ?
       q3+1 : q3);
endmodule

// ---------------------------------------------------------------------
// system rewards
// ---------------------------------------------------------------------

rewards "stime"
  !(q1 = Q & q2 = Q & q3 = Q) :
  q1 < M ? (1/mu1) :
  q2 < M ? (1/mu2) :
  q3 < M ? (1/mu3) :
  q1 <= q2 & q1 <= q3 ? (1/mu1) :
             q2 <= q3 ? (1/mu2) :
                        (1/mu3) ;
endrewards

rewards "sources"
  true : sources;
endrewards


// ---------------------------------------------------------------------
// FiniteClass.props
// ---------------------------------------------------------------------

// rejection probability
```

```
"reject": S=? [ q1 = Q & q2 = Q & q3 = Q ] ;

// mean service time (not considering rejection)
"stime": R{"stime"}=? [ S ] ;

// mean service time
"stime0": "stime"/(1-"reject") ;

// number of currently not serviced sources
"sources": R{"sources"}=? [ S ];

// response time (not considering rejection)
"rtime":  (N/"sources"-1)/lambda;

// response time
"rtime0": "rtime"/(1-"reject");

// waiting time (not considering rejection)
"wtime":  "rtime"-"stime";

// waiting time
"wtime0":  "wtime"/(1-"reject");
```

B.3. Common Queue

```
// ----------------------------------------------------------------------
// FiniteCommonHP.prism
// A scheduling model for computational clusters.
//
// The model is a finite source version of the "common queue" model with
// "high-performance priority" described in
//
//   Tien v. Do, Binh t. Vu, Xuan T. Tran, Anh P. Nguyen:
//   "A generalized model for investigating scheduling schemes in
//   computational clusters", Simulation Modelling Practice and
//   Theory, 37 (2013), 30-42.
//
// Authors: Berczes Tamas <berczes.tamas@inf.unideb.hu> and
// Wolfgang Schreiner <Wolfgang.Schreiner@risc.jku.at>
//
// Copyright (C) 2014 Department of Informatics Systems and Networks,
// University of Debrecen, Debrecen, Hungary (http://irh.inf.unideb.hu)
// and Research Institute for Symbolic Computation, Johannes Kepler
// University, Linz, Austria (http://www.risc.jku.at)
// ----------------------------------------------------------------------

// checking parameters: "sparse", "Jacobi", epsilon=0.001

// continuous time markov chain (ctmc) model
ctmc

// ----------------------------------------------------------------------
// system parameters
// -----------------------------------------------------------------

// arrival rates
const double lambda;

// queue size
const int Q = 32;

// number of server classes and number of servers per class
const int K = 3;
const int M = 8;

// population size
const int N = 60;

// ranking based on performance
const double rp1 = 1.0;
const double rp2 = 0.82;
const double rp3 = 0.43;

// ranking based on energy efficiency
const double re1 = 0.64;
const double re2 = 0.66;
const double re3 = 1.0;

// service rates according to chosen ranking
const double mu1 = rp1;
const double mu2 = rp2;
const double mu3 = rp3;

// -----------------------------------------------------------------
// system model
// -----------------------------------------------------------------

module Sources
  sources: [0..N] init N;
  [server]  sources > 0 -> lambda*sources : (sources' = sources-1);
  [source1a] sources < N -> (sources' = sources+1);
  [source1b] sources < N -> (sources' = sources+1);
  [source2a] sources < N -> (sources' = sources+1);
  [source2b] sources < N -> (sources' = sources+1);
  [source3a] sources < N -> (sources' = sources+1);
  [source3b] sources < N -> (sources' = sources+1);
endmodule

module System
  q: [0..Q] init 0;

  q1: [0..M] init 0;
  q2: [0..M] init 0;
  q3: [0..M] init 0;

  [source1a] q1 > 0 -> q1*mu1 : (q1' = q1-1);
  [source1b] q1 < M & q > 0 -> (M-q1)*mu1 : (q' = q-1);
  [source2a] q2 > 0 -> q2*mu2 : (q2' = q2-1);
  [source2b] q2 < M & q > 0 -> (M-q2)*mu2 : (q' = q-1);

  [source3a] q3 > 0 -> q3*mu3 : (q3' = q3-1);
  [source3b] q3 < M & q > 0 -> (M-q3)*mu3 : (q' = q-1);

  [server] !(q = Q & q1 = M & q2 = M & q3 = M) ->
    (q' =
      q1 = M & q2 = M & q3 = M & q < Q ?
        q+1 : q) &
    (q1' =
      q1 < M ?
        q1+1 : q1) &
    (q2' =
      q1 = M & q2 < M ?
        q2+1 : q2) &
    (q3' =
      q1 = M & q2 = M & q3 < M ?
        q3+1 : q3);
endmodule

// ------------------------------------------------------------------
// system rewards
// ------------------------------------------------------------------

rewards "stime"
  !(q = Q & q1 = M & q2 = M & q3 = M) :
  q1 < M ? (1/mu1) :
  q2 < M ? (1/mu2) :
  q3 < M ? (1/mu3) :
    (1/mu1); // lower bound
    // (1/mu3); // upper bound
endrewards

rewards "sources"
  true : sources;
endrewards


// ------------------------------------------------------------------
// FiniteCommon.props
// ------------------------------------------------------------------

// rejection probability
"reject": S=? [ q = Q & q1 = M & q2 = M & q3 = M ] ;

// mean service time (not considering rejection)
"stime": R{"stime"}=? [ S ] ;

// mean service time
"stime0": "stime"/(1-"reject") ;

// number of currently not serviced sources
```

```
"sources": R{"sources"}=? [ S ];

// response time (not considering rejection)
"rtime":  (N/"sources"-1)/lambda;

// response time
"rtime0": "rtime"/(1-"reject");

// waiting time (not considering rejection)
"wtime":  max(0, "rtime"-"stime");

// waiting time
"wtime0":  "wtime"/(1-"reject");
```

C. Alternative Formulations

C.1. Finite Source, Separate Queue

```
// -------------------------------------------------------------------
// FiniteSeparateHP.prism
// A scheduling model for computational clusters.
//
// The model is a finite version of the "separate queue" model with
// "high-performance priority" described in
//
//   Tien v. Do, Binh t. Vu, Xuan T. Tran, Anh P. Nguyen:
//   "A generalized model for investigating scheduling schemes in
//   computational clusters", Simulation Modelling Practice and
//   Theory, 37 (2013), 30-42.
//
// Authors: Adam Toth <adamtoth102@gmail.com> and
// Berczes Tamas <berczes.tamas@inf.unideb.hu> and
// Wolfgang Schreiner <Wolfgang.Schreiner@risc.jku.at>
//
// Copyright (C) 2014 Department of Informatics Systems and Networks,
// University of Debrecen, Debrecen, Hungary (http://irh.inf.unideb.hu)
// and Research Institute for Symbolic Computation, Johannes Kepler
// University, Linz, Austria (http://www.risc.jku.at)
// -------------------------------------------------------------------

// checking parameters: "sparse" (1GB), "Jacobi", epsilon=0.01

// Q=1 gives rejection probability from 2E-4 (U0=0.5) to 0.08 (U0=0.9)
// so the results for high U0 are not very accurate

// continuous time markov chain (ctmc) model
ctmc

// -------------------------------------------------------------------
// system parameters
// -------------------------------------------------------------------
// arrival rates
const double lambda;

// queue size
const int Q = 3;

// number of server classes and number of servers per class
const int K = 3;
const int M = 8;

// population size
const int N = 60;

// ranking based on performance
const double rp1 = 1.0;
const double rp2 = 0.82;
const double rp3 = 0.43;

// ranking based on energy efficiency
const double re1 = 0.64;
const double re2 = 0.66;
const double re3 = 1.0;

// service rates according to chosen ranking
const double mu1 = rp1;
const double mu2 = rp2;
const double mu3 = rp3;

// -----------------------------------------------------------------
// system model
// -----------------------------------------------------------------

formula MinNode = min(q11,q12,q13,q14);
formula MinNode2 = min(q21,q22,q23,q24);
formula MinNode3 = min(q31,q32,q33,q34);

module Sources
  sources: [0..N] init N;

   [sServers1] sources > 0 & MinNode = q11  & MinNode <= 3 ->
    sources*lambda : (sources' = sources-1);
  [sServers2] sources > 0 & MinNode = q12 & q11 > q12 & MinNode <= 3 ->
    sources*lambda : (sources' = sources-1);
  [sServers3] sources > 0 &
             MinNode = q13 & q11 > q13 & q12 > q13 &
             MinNode <= 3 ->
    sources*lambda : (sources' = sources-1);
  [sServers4] sources > 0 &
             MinNode = q14 & q11 > q14 & q12 > q14 & q13 > q14 &
             MinNode <= 3 ->
    sources*lambda : (sources' = sources-1);
  [sServers5] sources > 0  & MinNode = 3 & MinNode2 <= 3 & MinNode2 = q21 ->
    sources*lambda : (sources' = sources-1);
  [sServers6] sources > 0 &
```

```
              MinNode = 3 & MinNode2 <= 3 & MinNode2 = q22 & q21 > q22 ->
    sources*lambda : (sources' = sources-1);
  [sServers7] sources > 0 &
              MinNode = 3 & MinNode2 <= 3 & MinNode2 = q23 &
              q21 > q23 & q22 > q23 ->
    sources*lambda : (sources' = sources-1);
  [sServers8] sources > 0 & MinNode = 3 & MinNode2 <= 3 & MinNode2 = q24 &
              q21 > q24 & q22 > q24 & q23 > q24 ->
    sources*lambda : (sources' = sources-1);
  [sServers9] sources > 0  &
              MinNode = 3 & MinNode2 = 3 & MinNode3 <= 3 & MinNode3 = q31 ->
    sources*lambda : (sources' = sources-1);
  [sServers10] sources > 0 &
               MinNode = 3 & MinNode2 = 3 & MinNode3 <= 3 & MinNode3 = q32 &
               q31 > q32 ->
    sources*lambda : (sources' = sources-1);
  [sServers11] sources > 0 &
               MinNode = 3 & MinNode2 = 3 & MinNode3 <= 3 & MinNode3 = q33 &
               q31 > q33 & q32 > q33->
    sources*lambda : (sources' = sources-1);
  [sServers12] sources > 0 &
               MinNode = 3 & MinNode2 = 3 & MinNode3 <= 3 & MinNode3 = q34 &
               q31 > q34 & q32 > q34 & q33 > q34->
    sources*lambda : (sources' = sources-1);

  [ssources1] sources < N -> (sources' = sources+1);
  [ssources2] sources < N -> (sources' = sources+1);
  [ssources3] sources < N -> (sources' = sources+1);
  [ssources4] sources < N -> (sources' = sources+1);
  [ssources5] sources < N -> (sources' = sources+1);
  [ssources6] sources < N -> (sources' = sources+1);
  [ssources7] sources < N -> (sources' = sources+1);
  [ssources8] sources < N -> (sources' = sources+1);
  [ssources9] sources < N -> (sources' = sources+1);
  [ssources10] sources < N -> (sources' = sources+1);
  [ssources11] sources < N -> (sources' = sources+1);
  [ssources12] sources < N -> (sources' = sources+1);

endmodule

module server1
  q11: [0..Q] init 0;
  [sServers1] q11 < Q -> (q11' = q11+1);
  [ssources1] q11 > 0 -> mu1 : (q11' = q11-1);
endmodule

module server2
  q12: [0..Q] init 0;
  [sServers2] q12 < Q -> (q12' = q12+1);
  [ssources2] q12 > 0 -> mu1 : (q12' = q12-1);
endmodule

module server3
  q13: [0..Q] init 0;
  [sServers3] q13 < Q -> (q13' = q13+1);
  [ssources3] q13 > 0 -> mu1 : (q13' = q13-1);
endmodule

module server4
  q14: [0..Q] init 0;
  [sServers4] q14 < Q -> (q14' = q14+1);
  [ssources4] q14 > 0 -> mu1 : (q14' = q14-1);
endmodule

module server5
  q21: [0..Q] init 0;
  [sServers5] q21 < Q -> (q21' = q21+1);
  [ssources5] q21 > 0 -> mu2 : (q21' = q21-1);
endmodule

module server6
  q22: [0..Q] init 0;
  [sServers6] q22 < Q -> (q22' = q22+1);
  [ssources6] q22 > 0 -> mu2 : (q22' = q22-1);
endmodule

module server7
  q23: [0..Q] init 0;
  [sServers7] q23 < Q -> (q23' = q23+1);
  [ssources7] q23 > 0 -> mu2 : (q23' = q23-1);
endmodule

module server8
  q24: [0..Q] init 0;
  [sServers8] q24 < Q -> (q24' = q24+1);
  [ssources8] q24 > 0 -> mu2 : (q24' = q24-1);
endmodule

module server9
  q31: [0..Q] init 0;
  [sServers9] q31 < Q -> (q31' = q31+1);
  [ssources9] q31 > 0 -> mu3 : (q31' = q31-1);
endmodule

module server10
  q32: [0..Q] init 0;
  [sServers10] q32 < Q -> (q32' = q32+1);
  [ssources10] q32 > 0 -> mu3 : (q32' = q32-1);
endmodule

module server11
  q33: [0..Q] init 0;
  [sServers11] q33 < Q -> (q33' = q33+1);
  [ssources11] q33 > 0 -> mu3 : (q33' = q33-1);
endmodule

module server12
  q34: [0..Q] init 0;
  [sServers12] q34 < Q -> (q34' = q34+1);
  [ssources12] q34 > 0 -> mu3 : (q34' = q34-1);
endmodule

// ------------------------------------------------------------------
// system rewards
// ------------------------------------------------------------------

rewards "stime"
    !(q11 = Q & q12 = Q & q13 = Q & q14 = Q &

      q21 = Q & q22 = Q & q23 = Q & q24 = Q &

      q31 = Q & q32 = Q & q33 = Q & q34 = Q  ) :
    min(q11, q12, q13, q14) <=
      min(q21, q22, q23, q24,
          q31, q32, q33, q34) ? (1/mu1) :
    min(q21, q22, q23, q24) <=
      min(q31, q32, q33, q34) ? (1/mu2) : (1/mu3);
endrewards

rewards "qload"
  true : q11+q12+q13+q14+
         q21+q22+q23+q24+
         q31+q32+q33+q34;
endrewards

rewards "qsources"
  true : sources;
endrewards
```