Analyzing Cluster Scheduling Schemes by Probabilistic Model Checking (Addendum)*

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

Tamás Bérczes*
berczes.tamas@inf.unideb.hu

János Sztrik*
berczes.tamas@inf.unideb.hu

Ádám Tóth*
adamtoth102@gmail.com

* Faculty of Informatics
University of Debrecen, Hungary

October 6, 2015

Abstract

This short note presents an addendum to the previously published report "Analyzing Cluster Scheduling Schemes by Probabilistic Model Checking" in which we have used 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. Here we model and analyze two new policies for the "Separate Queue" scheme where each server has a separate job queue; in these policies the mean response time of new jobs is taken into account. It is shown that the policies may perform better than the original one; for a certain range of parameters, however, they also may perform worse.

*Supported by the project 90öu6 "Leistungsmodellierung von Drahtlosen Sensor-Netzwerken" of the Stiftung Aktion Österreich-Ungarn.

Addendum

In [5] we have applied the probabilistic model checker PRISM [4, 3] to model and analyze various cluster scheduling schemes that were originally introduced in [2] and analyzed by simulation there. One of these schemes was the "Separate Queue" scheme where 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. In the "High Performance" (HP) policy, ranking is based on server performance, i.e., the fastest server is preferred.

In the not yet published manuscript [1], two variations of this policy are suggested:

1. *Mean Response Time (MRT):* in this policy, a new job is placed in the queue of the server with the (depending on the length of the queue and the performance of the server) minimal expected response time.

2. *Mean Response Time with High Performance (MRTHP):* in this policy, if there are some idle servers, a new job is placed in the queue of that idle server that has highest performance; if no server is idle, the MRT policy is applied.

By numerical simulation it is shown that, while the MRT policy only exhibits moderate improvements over the basic HP policy, the MRTHP policy significantly reduces the mean service time and waiting time, provided that job arrival rates are sufficiently low; for higher rates, all policies show convergent results.

In this addendum, we re-evaluate these policies by the analysis of three PRISM models. Appendix A lists the original "Separate Queue" model with HP policy presented in [5]; Appendix B shows a new model that implements the MRT policy; Appendix C presents the new PRISM model for the MRTHP policy.

For numerical simulation, in [1] the number of sources $N = 150$ and the number of servers per class $M = 8$ are chosen. For maximum queue length $Q = 3$, this however gives a state space of size $151 \cdot 4^{24} \simeq 4 \cdot 10^{16}$ which is not practical for the analysis with PRISM due to time and in particular memory constraints. We rather apply $N = 60$ and $M = 4$ which yields the more manageable state space size $61 \cdot 4^{12} \approx 10^9$ (of which less than $2 \cdot 10^7$ states are reachable); a single run of the model checker (i.e., the computation of one data point) then uses about 1GB memory and 200s time. Thus we (have to) focus our attention not on the reproduction of the quantitative results of [1] but on the validation of their qualitative conclusions. In detail, we apply for the analysis in PRISM the "Sparse" engine with the "Gauss-Seidel" solver and relative accuracy $\epsilon = 0.05$ (which is chosen comparatively high to reduce the execution time of the numerical solver).

The first part of our analysis illustrated in Figure 1 focuses on the parameter range $\lambda \in [0.07, 0.18]$ which is also applied in [1] (but, as stated above, for larger values of N and M). In this range the rejection probability (due to queues becoming full) is for all models less than 0.004; it thus can be ignored in the subsequent analysis. Different from [1], in this range it is mostly the original HP policy which minimizes the response time (rtime); the MRTHP policy behaves essentially identical to HP (since for small arrival rate usually an empty queue is detected). The MRT policy behaves worst, although it minimizes the service time (stime), because

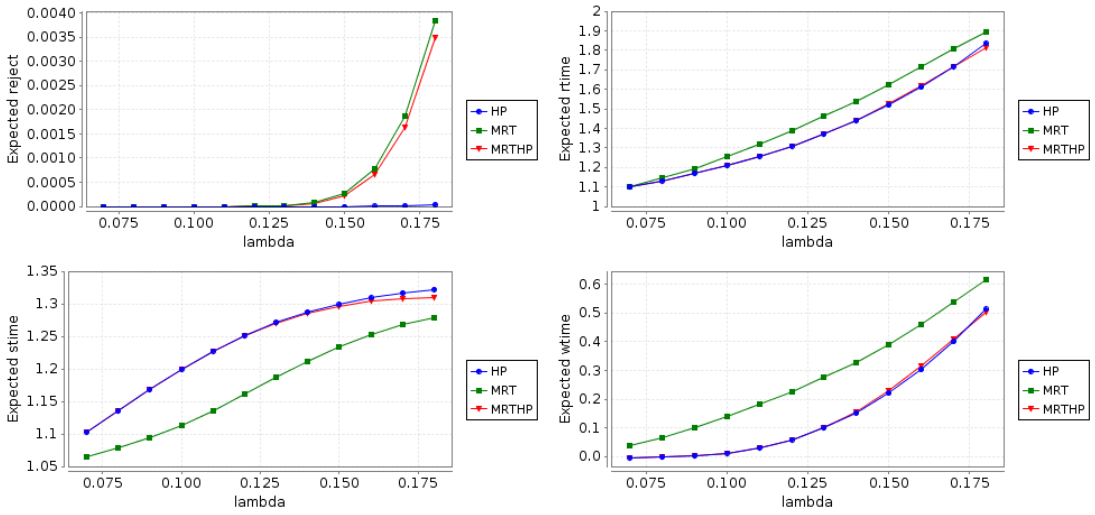

Figure 1: PRISM Performance Evaluation (Part 1)

this effect is counterbalanced by the largest waiting time (wtime). So from these figures, the new policies do not really pay off.

However, looking at the growth of the curves towards the largest value $\lambda = 0.18$, we realize that the behavior might change for larger arrival rates. In Figure 2 we thus extend the parameter range to $\lambda \in [0.19, 0.22]$ where the rejection rate is still at an acceptable level of less than 0.03. In this range both MRT and MRTHP significantly improve the response time with MRTHP performing best (however the difference between MRT and MRTHP vanishes for larger arrival rates). This is due to the fact that not only the service time but also the waiting time falls in these policies below that of the HP policy.

All in all, these results present a slightly more differentiated picture than the figures in [1]; we see that the new policies MRT and MRTHP only pay off, if the arrival rate exceeds a certain minimal threshold. Moreover, [1] shows that, once a certain maximal threshold is reached, all policies show similar results; this range, however, cannot be easily investigated by the PRISM model checker, because the high arrival rates yield high rejection probabilities (due to the limited queue sizes) which distort the results.

Summarizing, the additional analysis presented in this small note shows that the optimization effect of the new policies MRT and MRTHP are quite sensitive to the arrival rate; there is a certain "sweet spot" in which they are truly beneficial; if this sweet spot is missed on one side (arrival rate too small), the policies even behave worse than the original policy HP; if they get too large, all policies behave the same.

References

[1] Tamás Bérczes and Ádám Tóth. *New Scheduling Algorithms for Finite-Source Cluster Networks*. To appear. 2015.

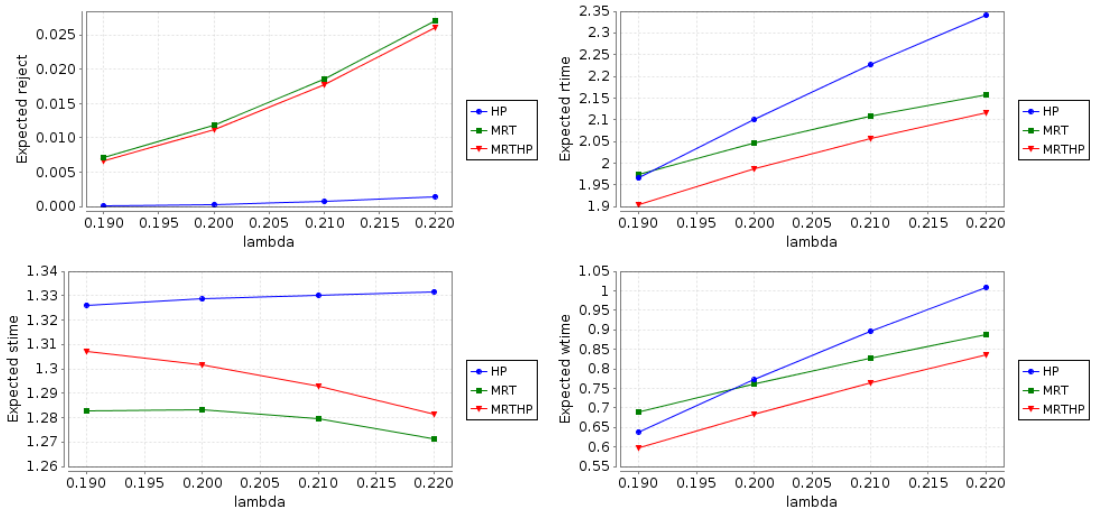

Figure 2: PRISM Performance Evaluation (Part 2)

[2] 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.

[3] 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 (C AV'11)*. Ed. by G. Gopalakrishnan and S. Qadeer. Vol. 6806. Lecture Notes in Computer Science. Springer, 2011, pp. 585–591.

[4] David A. Parker, ed. *PRISM — Probabilistic Symbolic Model Checker*. http://www.prismmodelchecker.org. Department of Computer Science, University of Oxford, UK. 2015.

[5] Wolfgang Schreiner, Tamás Bérczes, and Ádám Tóth. *Analyzing Cluster Scheduling Schemes by Probabilistic Model Checking*. Technical Report. Johannes Kepler University, Linz, Austria: Research Institute for Symbolic Computation (RISC), Sept. 2014.

A High Performance Priority (HP) Model

```
// ------------------------------------------------------------------
// 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 Mean Response Time Priority (MRT) Model

```
// -----------------------------------------------------------------
// FiniteSeparateMRT.prism
// A scheduling model for computational clusters.
//
// The model is a finite source version of the "separate queue" model with
// "mean response time priority" described in
//
//   Tamas Berczes et al.
//   "New Scheduling Algorithms for Finite-Source Cluster Networks"
//   (to appear)
//
// Authors: Berczes Tamas <berczes.tamas@inf.unideb.hu> and
// Wolfgang Schreiner <Wolfgang.Schreiner@risc.jku.at>
//
// Copyright (C) 2015 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 (3GB), "Gauss-Seidel", epsilon=0.05

// continuous time markov chain (ctmc) model
ctmc

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

// arrival rates
const double lambda;

// queue sizes for different server classes
const int Q1 = 3;
const int Q2 = 3;
```

```
const int Q3 = 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
// ------------------------------------------------------------------

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

formula t11 = (q11+1)/mu1;
formula t12 = (q12+1)/mu1;
formula t13 = (q13+1)/mu1;
formula t14 = (q14+1)/mu1;
formula t21 = (q21+1)/mu2;
formula t22 = (q22+1)/mu2;
formula t23 = (q23+1)/mu2;
formula t24 = (q24+1)/mu2;
formula t31 = (q31+1)/mu3;
```

```
formula t32 = (q32+1)/mu3;
formula t33 = (q33+1)/mu3;
formula t34 = (q34+1)/mu3;

formula minNode =
  t11 <= min(t12,t13,t14,t21,t22,t23,t24,t31,t32,t33,t34) ? 1 :
  t12 <= min(t13,t14,t21,t22,t23,t24,t31,t32,t33,t34) ? 2 :
  t13 <= min(t14,t21,t22,t23,t24,t31,t32,t33,t34) ? 3 :
  t14 <= min(t21,t22,t23,t24,t31,t32,t33,t34) ? 4 :
  t21 <= min(t22,t23,t24,t31,t32,t33,t34) ? 5 :
  t22 <= min(t23,t24,t31,t32,t33,t34) ? 6 :
  t23 <= min(t24,t31,t32,t33,t34) ? 7 :
  t24 <= min(t31,t32,t33,t34) ? 8 :
  t31 <= min(t32,t33,t34) ? 9 :
  t32 <= min(t33,t34) ? 10 :
  t33 <= t34 ? 11 : 12;

formula accept =
  (q11 < Q1 & minNode = 1) |
  (q12 < Q1 & minNode = 2) |
  (q13 < Q1 & minNode = 3) |
  (q14 < Q1 & minNode = 4) |
  (q21 < Q2 & minNode = 5) |
  (q22 < Q2 & minNode = 6) |
  (q23 < Q2 & minNode = 7) |
  (q24 < Q2 & minNode = 8) |
  (q31 < Q3 & minNode = 9) |
  (q32 < Q3 & minNode = 10) |
  (q33 < Q3 & minNode = 11) |
  (q34 < Q3 & minNode = 12);

module Servers
  q11: [0..Q1] init 0;
  q12: [0..Q1] init 0;
  q13: [0..Q1] init 0;
  q14: [0..Q1] init 0;

  q21: [0..Q2] init 0;
  q22: [0..Q2] init 0;
  q23: [0..Q2] init 0;
  q24: [0..Q2] init 0;

  q31: [0..Q3] init 0;
  q32: [0..Q3] init 0;
  q33: [0..Q3] init 0;
  q34: [0..Q3] 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] accept ->
    (q11' = q11 < Q1 & minNode =  1 ? q11+1 : q11) &
    (q12' = q12 < Q1 & minNode =  2 ? q12+1 : q12) &
    (q13' = q13 < Q1 & minNode =  3 ? q13+1 : q13) &
    (q14' = q14 < Q1 & minNode =  4 ? q14+1 : q14) &
    (q21' = q21 < Q2 & minNode =  5 ? q21+1 : q21) &
    (q22' = q22 < Q2 & minNode =  6 ? q22+1 : q22) &
    (q23' = q23 < Q2 & minNode =  7 ? q23+1 : q23) &
    (q24' = q24 < Q2 & minNode =  8 ? q24+1 : q24) &
    (q31' = q31 < Q3 & minNode =  9 ? q31+1 : q31) &
    (q32' = q32 < Q3 & minNode = 10 ? q32+1 : q32) &
    (q33' = q33 < Q3 & minNode = 11 ? q33+1 : q33) &
    (q34' = q34 < Q3 & minNode = 12 ? q34+1 : q34) ;
endmodule

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

rewards "reject"
  !accept : 1;
endrewards

rewards "stime"
  accept : minNode <= 4 ? (1/mu1) : minNode <= 8 ? (1/mu2) : (1/mu3);
endrewards

rewards "sources"
  true : sources;
endrewards


// ------------------------------------------------------------------
// FiniteSeparate.props
// ------------------------------------------------------------------

// rejection probability
"reject": R{"reject"}=? [ S ] ;

// 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");
```

C Mean Response Time with High Performance Priority (MRTHP) Model

```
// ------------------------------------------------------------------
// FiniteSeparateMRTHP.prism
// A scheduling model for computational clusters.
//
// The model is a finite source version of the "separate queue" model with
// "mean response time with high-performance priority" described in
//
//   Tamas Berczes et al.
//   "New Scheduling Algorithms for Finite-Source Cluster Networks"
//   (to appear)
//
// Authors: Berczes Tamas <berczes.tamas@inf.unideb.hu> and
// Wolfgang Schreiner <Wolfgang.Schreiner@risc.jku.at>
//
// Copyright (C) 2015 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 (3GB), "Gauss-Seidel", epsilon=0.05

// continuous time markov chain (ctmc) model
ctmc

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

// arrival rates
const double lambda;

// queue sizes for different server classes
const int Q1 = 3;
```

```
const int Q2 = 3;
const int Q3 = 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
// ------------------------------------------------------------------

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

formula t11 = (q11+1)/mu1;
formula t12 = (q12+1)/mu1;
formula t13 = (q13+1)/mu1;
formula t14 = (q14+1)/mu1;
formula t21 = (q21+1)/mu2;
formula t22 = (q22+1)/mu2;
formula t23 = (q23+1)/mu2;
formula t24 = (q24+1)/mu2;
```

```
formula t31 = (q31+1)/mu3;
formula t32 = (q32+1)/mu3;
formula t33 = (q33+1)/mu3;
formula t34 = (q34+1)/mu3;

formula minNode =
  q11 = 0 | q12 = 0 | q13 = 0 | q14 = 0 |
  q21 = 0 | q22 = 0 | q23 = 0 | q24 = 0 |
  q31 = 0 | q32 = 0 | q33 = 0 | q34 = 0 ?
  (
   q11 = 0 ? 1  : q12 = 0 ? 2  : q13 = 0 ? 3  : q14 = 0 ? 4 :
   q21 = 0 ? 5  : q22 = 0 ? 6  : q23 = 0 ? 7  : q24 = 0 ? 8 :
   q31 = 0 ? 9  : q32 = 0 ? 10 : q33 = 0 ? 11 : 12
  ) :
  t11 <= min(t12,t13,t14,t21,t22,t23,t24,t31,t32,t33,t34) ? 1 :
  t12 <= min(t13,t14,t21,t22,t23,t24,t31,t32,t33,t34) ? 2 :
  t13 <= min(t14,t21,t22,t23,t24,t31,t32,t33,t34) ? 3 :
  t14 <= min(t21,t22,t23,t24,t31,t32,t33,t34) ? 4 :
  t21 <= min(t22,t23,t24,t31,t32,t33,t34) ? 5 :
  t22 <= min(t23,t24,t31,t32,t33,t34) ? 6 :
  t23 <= min(t24,t31,t32,t33,t34) ? 7 :
  t24 <= min(t31,t32,t33,t34) ? 8 :
  t31 <= min(t32,t33,t34) ? 9 :
  t32 <= min(t33,t34) ? 10 :
  t33 <= t34 ? 11 : 12;

formula accept =
  (q11 < Q1 & minNode = 1) |
  (q12 < Q1 & minNode = 2) |
  (q13 < Q1 & minNode = 3) |
  (q14 < Q1 & minNode = 4) |
  (q21 < Q2 & minNode = 5) |
  (q22 < Q2 & minNode = 6) |
  (q23 < Q2 & minNode = 7) |
  (q24 < Q2 & minNode = 8) |
  (q31 < Q3 & minNode = 9) |
  (q32 < Q3 & minNode = 10) |
  (q33 < Q3 & minNode = 11) |
  (q34 < Q3 & minNode = 12);

module Servers
  q11: [0..Q1] init 0;
  q12: [0..Q1] init 0;
  q13: [0..Q1] init 0;
  q14: [0..Q1] init 0;

  q21: [0..Q2] init 0;
  q22: [0..Q2] init 0;
  q23: [0..Q2] init 0;
  q24: [0..Q2] init 0;

  q31: [0..Q3] init 0;
  q32: [0..Q3] init 0;
  q33: [0..Q3] init 0;
```

```
  q34: [0..Q3] 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] accept ->
    (q11' = q11 < Q1 & minNode =  1 ? q11+1 : q11) &
    (q12' = q12 < Q1 & minNode =  2 ? q12+1 : q12) &
    (q13' = q13 < Q1 & minNode =  3 ? q13+1 : q13) &
    (q14' = q14 < Q1 & minNode =  4 ? q14+1 : q14) &
    (q21' = q21 < Q2 & minNode =  5 ? q21+1 : q21) &
    (q22' = q22 < Q2 & minNode =  6 ? q22+1 : q22) &
    (q23' = q23 < Q2 & minNode =  7 ? q23+1 : q23) &
    (q24' = q24 < Q2 & minNode =  8 ? q24+1 : q24) &
    (q31' = q31 < Q3 & minNode =  9 ? q31+1 : q31) &
    (q32' = q32 < Q3 & minNode = 10 ? q32+1 : q32) &
    (q33' = q33 < Q3 & minNode = 11 ? q33+1 : q33) &
    (q34' = q34 < Q3 & minNode = 12 ? q34+1 : q34) ;
endmodule

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

rewards "reject"
  !accept : 1;
endrewards

rewards "stime"
  accept : minNode <= 4 ? (1/mu1) : minNode <= 8 ? (1/mu2) : (1/mu3);
endrewards

rewards "sources"
  true : sources;
endrewards


// ------------------------------------------------------------------
// FiniteSeparate.props
// ------------------------------------------------------------------

// rejection probability
```

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

// 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");
```