

Markov Model Solver (MMS)

Andrew H. Nguyen

**June 11, 1997
CSE 228B**

Markov Model Solver (MMS)

1 Introduction

Disk array architectures have been proposed and implemented to provide fault-tolerant high performance while maintaining high data reliability and availability as conventional disks. Numerous studies have been done in the past to analyze disk array reliability. This lead to the computation of probability of disk failures based on several factors including byte error rate and the repair rate of a disk. One traditional form of measurement that has been used and remains unchanged is MTTF (mean time to failure). As an example, consider a disk array of 2 disks and single-fault-tolerant, the MTTF is $1/\lambda$. For a group of n drives, using the Markov reliability model we can obtain the exact MTTF equation in a formal and direct manner. Unfortunately, when the value of n is greater than 5, the manual process of calculating MTTF is often long and tedious. This is the primary motivation of MMS (Markov Model Solver), which takes a direct textual representation of Markov model as input and automatically generates MTTF equation in *Mathematica* notations. Since it is relevant to question how much additional information is given by the variance if MTTF is already known, MMS also outputs corresponding variance equation. The output equations are in the form of symbolic matrix multiplication, which can be computed quickly and efficiently by *Mathematica* application.

2 Organization

This report is presented as follows. In section 3, the choice of system environment and the programming language will be discussed. The format of Markov model textual representation which is used as input to the MMS program, is specified in section 4. Section 5 will contain explanation of the algorithm and the output of MMS program. In section 6, a few sample output of the program to demonstrate its correctness. Conclusion is in the last section.

3 Environment

Besides automating the MTTF and variance calculation process, other important goals are to achieve high performance with the currently available resources. MMS will be written in standard *awk*, a pattern scanning and processing language. *Awk* was chosen for its specialized pattern matching capabilities and provides powerful, yet simple and efficient data structures. (i.e, multidimensional array indexed by strings, which will be useful for implementing MMS). Since in the Computer Science Department of UCSD, only *bonzo* machine has *Mathematica* installed, the MMS will be written using the version of *awk* available on that machine for convenient reason. The most important gain in performance is the use of *Mathematica* application, which have the capability of performing symbolic matrix multiplication in extremely fast rate.

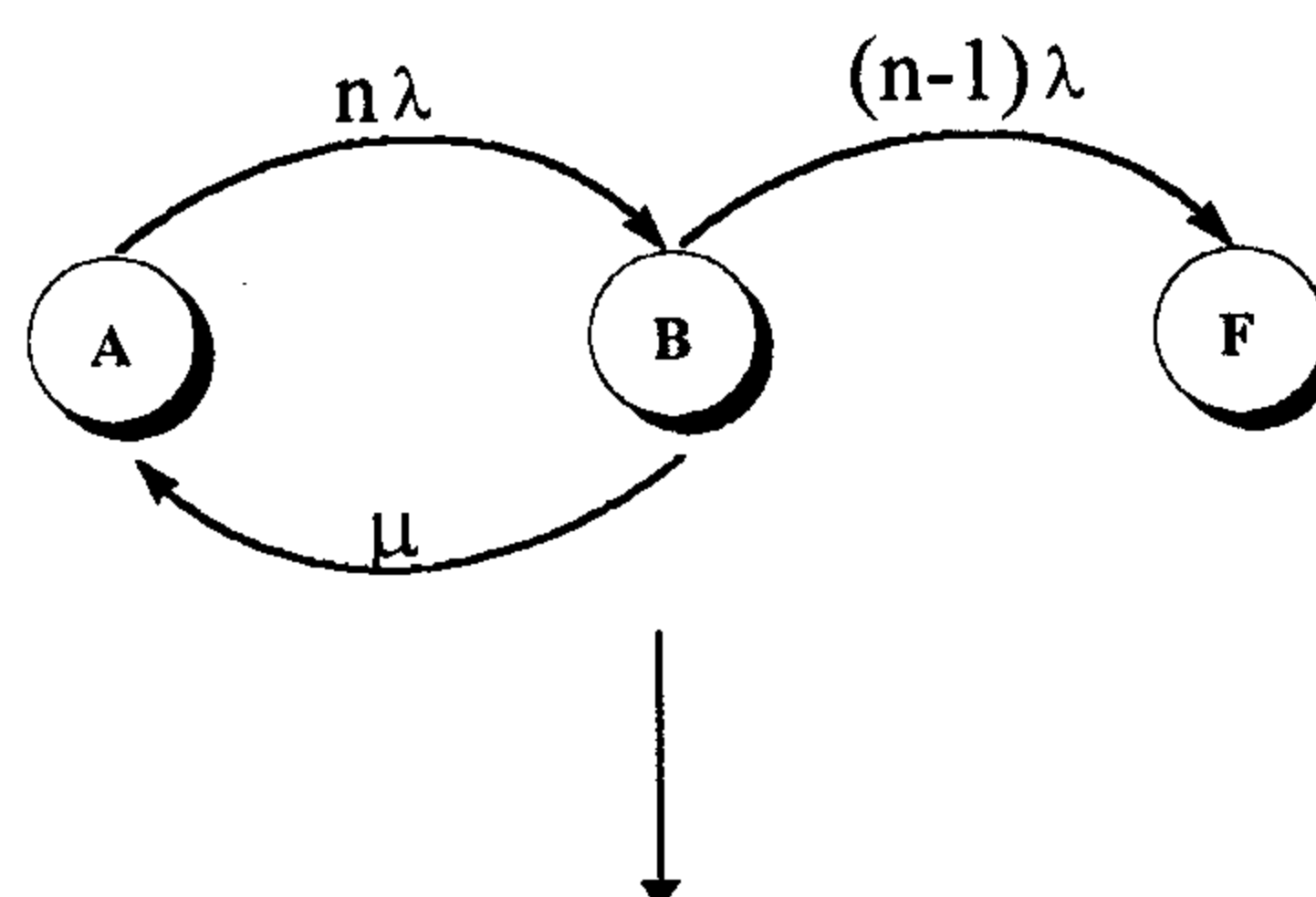
4 Input Format

Since Markov Model Diagram cannot be directly read by MMS, a textual representation of the model is required. To provide user friendly interface, the format of the input file is designed to be simple while maintaining one-to-one mapping with the Markov model. The input file format is specified below:

<i>Input File Format</i>
BEGIN TRANSITION <name1 name2 expression> END TRANSITION BEGIN PROBABILITY <name expression> END PROBABILITY

Each line in the transition section specifies a state transition from *name1* to *name2* having its transition rate of *expression*. State name can be a string of characters, digits, or any combination of both. No spaces is allowed in state name. Since the output of MMS is in Mathematica notation, all expressions are required to be in *Mathematica* expression format. MMS only allows one and only one absorbing(failure) state. Another important requirement is that the variables: "S", "M", "varM", "OneMatrix", "Prob", "VarProb", "VarOneMatrix", "SecondMomentum", "Variance", and "MTTDL", can not be used in any expression because they are reserved to be used for setting up Variance and MTTDL equations. The state transition and its associated rate together can be given in any order. Each line in the probability section assigns a reward rate (*expression*) to a non-failure state (*name*). No assumption of the reward rate for an particular state is made, hence all non-absorbing states that appear in the transition section are required to have an entry in the probability section. Transition and probability section can be given in any order. As a simple example, configuration 1 shows a Markov diagram with three states: A, B and F, where A is the initial state and F is the absorbing state. The symbol λ represents a single disk failure rate and μ is a single disk repair rate. The associated text representation is shown below.

Configuration 1



Config 1 Text Format
BEGIN TRANSITION
A B n λ
B A μ
B F (n-1) λ
END TRANSITION
BEGIN PROBABILITY
A 1
B 0
END PROBABILITY

4.1 Error Handling Capabilities

Adhering to one of the goals of providing friendly user interface, MMS has syntactic error checking capability. It can detect probability and state transition duplication (i.e *name1* and *name2* appear together more than once). MMS can also detect the existence of an absorbing state, and prompt for one when necessary. All available error messages and its corresponding explanation are listed in the table below.

Message Displayed By MMS	Explanation
"Error: Duplication of probability entry." Line No: <line content>	A state is not allowed to assigned reward value more than once.
"Error: Required at least three fields per line entry." Line No: <line content>	SSM requires to have at least 3 fields per line in the transition section. Two fields for state names and one or more for transition rate. If input file contain less than 3 fields, this error message will be displayed and line number where the error occurs.

"Error: Duplicate transition entry." Line No: <line content>	Entries in transition section are not allowed to be given more than once.
"Error: No transition section found."	This error message indicates that input file has not transition section.
"Error: There exist no failure state."	MMS attempted to find a failure state, but could not find one, hence this message.
"Error: Invalid probability entry." Line No: <line content>	Indicates that only one field of the probability entry is given when at least two fields is required.
"Error: Source and destination states are the same." Line No: <line content>	This message is displayed when the state <i>name1</i> and <i>name2</i> are the same.
"Error: There exist multiple failure states."	Only one absorbing state is allowed in MMS.
"Error: No probability section found."	This indicates that probability section has no entry.
"Error: No transtion entry found."	This message indicates that the transition section is empty.
"Error: No probability value for state X."	This indicate that state X which is a non-absorbing state, has no reward value. MMS require every state except absorbing state to have one entry in the probability section.

The two tables below show what the user actually see when MMS reads in an erroneous input files. In the *ErrorInputFile* on line 2, the transition rate is missing. If running MMS using this input file, MMS will detect and display an informative error message. Notice that MMS will does not continue processing the input file once an error is detected.

<i>ErrorInputFile</i>
1. BEGIN TRANSITION 2. A F 3. END TRANSITION 4. BEGIN PROBABILITY 5. A 1 6. END PROBABILITY

Error Message Generated By MMS
bonzo % awk -f MMS ErrorInputFile Error: Required at least three fields per line entry. Line 2: A F bonzo %

5 MMS Algorithm & Output

MMS program has one parameter input, text file which is a special format specified in section 4. The content of input file is processed line by line. Since there are two different sections in the input file: probability and transition, MMS need performs a pattern matching for reserved key phrases in order to determine which section that it is processing. The four key phrases are:

1. BEGIN TRANSITION
2. END TRANSITION
3. BEGIN PROBABILITY
4. END PROBABILITY

The key phrases can be both upper and lowercase. Once a section type is detected, MMS process each entry by doing syntactic checking and then store the values into multidimensional arrays. If a syntactic error occurs during processing, MMS displays an informative error message and then aborts. After

making one complete pass through the input file, MMS prints out MTTF and Variance equations in *Mathematica* notation. Particularly, the formula that is being used to calculate MTDDL is described below.

$$\text{MTDDL} = -1 \bullet \text{Inverse}[M] \bullet P$$

where 1 is a matrix of one's, P is the probability matrix obtained in the probability section. M is the matrix obtained after processing the transition section. Similarly, the equation for calculating variance is

$$\text{Variance} = E(t^2) - [E(t)]^2$$

$$E(t) = \text{MTDDL}$$

$$E(t^2) = \lim_{s \rightarrow 0} \{-2[P(0)] (d/ds)[(S + 1)[I] - [T]]^{-1}\}$$

where P(0) is the probability of being in a state at time zero, T is the transition matrix. A proof of the MTDDL and Variance can be found in [Shooman, p. 241-243].

The table below shows a sample output of MMS program given a small input file.

<i>Input File</i>	<i>MMS Output</i>
<pre> BEGIN TRANSITION A B n 1 B A u B F (n-1) 1 END TRANSITION BEGIN PROBABILITY A 1 B 0 END PROBABILITY </pre>	<pre> M = { { -(n 1), u } { n 1, -(u + (n-1) 1) } } VarM = { { S + (n 1), u } { n 1, S + (u + (n-1) 1) } } OneMatrix = {1,1} VarOneMatrix = {{1},{1}} Prob = {{1},{0}} VarProb = {1,0} SecondMomentum = (Limit[{-2 VarProb.(D[(Inverse[VarM]),S]),S->0}].VarOneMatrix MTDDL = Simplify[- OneMatrix . Inverse[M] . Prob] Variance = Simplify[SecondMomentum - MTDDL^2] </pre>

Once the output of MMS is generated, the equations can be directly feed into *Mathematica* program. By typing MTDDL, Mathematica will perform symbolic or numeric matrix computation and then display the simplified MTDDL expression. Variance calculated in similar fashion. Continue with the example above, the resulting MTDDL and Variance expression output by Mathematica are:

<i>Mathematica Output</i>
$\text{MTDDL} = \left\{ \frac{-1 + 2 \, 1 \, n + u}{1 \, (-1 + n) \, n} \right\}$
$\text{Variance} = \left\{ \frac{1^2 - 2 \, 1 \, n^2 - 2 \, 1 \, n \, u - u^2}{1 \, (-1 + n) \, n} \right\}$

6 Sample Output

To serve as a form of correctness verification of MMS program, a few small classical Markov diagrams is used. In addition, Markov diagrams of RAID Level 5 with 4 disks and DATUM with 6 disks is also presented.

Table S.1

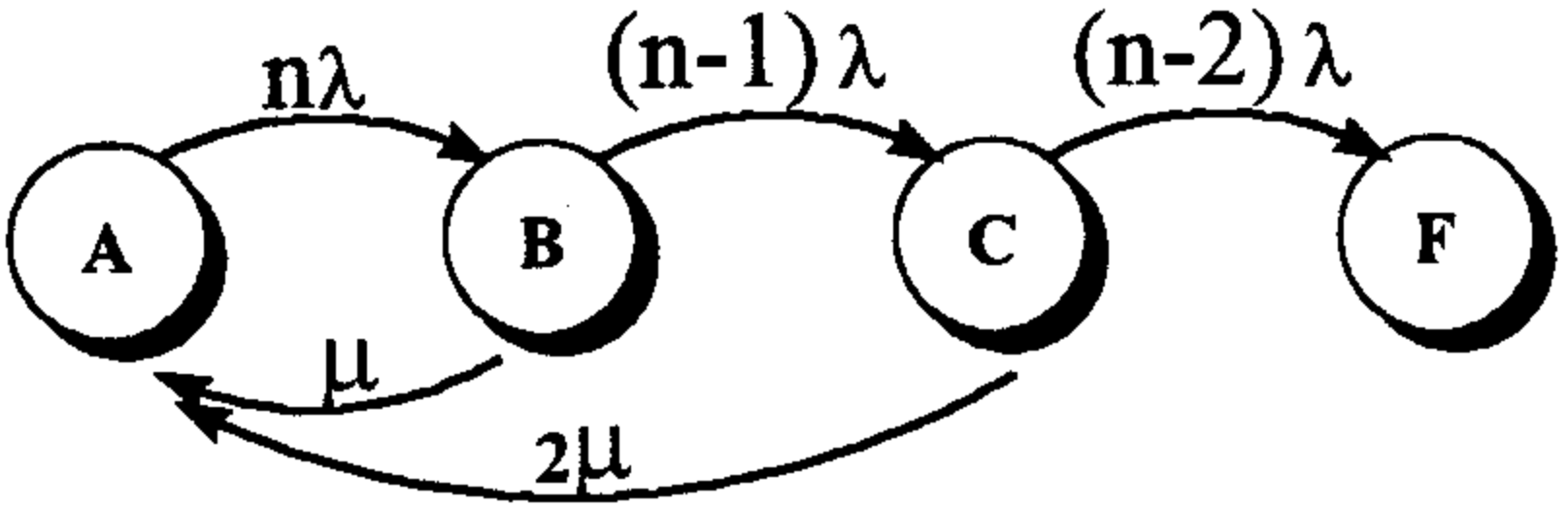
Text representation	Raid Level 6 λ single disk failure rate u sing disk repair rate
<pre> BEGIN TRANSITION A B n 1 B A u B C (n-1) 1 C F (n-2) 1 C A 2 u END TRANSITION BEGIN PROBABILITY A 1 B 0 C 0 END PROBABILITY </pre>	 <pre> graph LR A((A)) -- "nλ" --> B((B)) B -- "μ" --> A B -- "(n-1)λ" --> C((C)) C -- "2μ" --> B C -- "(n-2)λ" --> F((F)) </pre>
<pre> M = { { -(n 1), u, 2 u } { n 1, -(u + (n-1) 1), 0 } { 0, (n-1) 1, -((n-2) 1 + 2 u) } } VarM = { { S + (n 1), u, 2 u } { n 1, S + (u + (n-1) 1), 0 } { 0, (n-1) 1, S + ((n-2) 1 + 2 u) } } OneMatrix = {1,1,1} VarOneMatrix = {{1},{1},{1}} Prob = {{1},{0},{0}} VarProb = {1,0,0} SecondMomment = (Limit[{-2 VarProb.(D[(Inverse[VarM]),S]),S->0}).VarOneMatrix MTTDL = Simplify[- OneMatrix . Inverse[M] . Prob] Variance = Simplify[SecondMomment - MTTDL^2] </pre>	
$MTTDL = \frac{2^2 1^2 - 6^2 1^2 n + 3^2 1^2 n^2 - 4^2 1^2 u + 5^2 1^2 n u + 2^2 u^2}{1^3 n (2^2 - 3^2 n + n^2)}$	
$Variance = \frac{\begin{aligned} & (16^6 1^6 - 16^6 1^6 n - 84^6 1^6 n^2 + 184^6 1^6 n^3 - 146^6 1^6 n^4 + 52^6 1^6 n^5 - \\ & 7^6 1^6 n^6 + 32^5 1^5 u - 448^5 1^5 n u + 1216^5 1^5 n^2 u - 1272^5 1^5 n^3 u + \\ & 578^5 1^5 n^4 u - 96^5 1^5 n^5 u - 352^4 1^4 u^2 + 1792^4 1^4 n u^2 - 2924^4 1^4 n^2 u^2 + \\ & 1868^4 1^4 n^3 u^2 - 409^4 1^4 n^4 u^2 + 736^3 1^3 u^3 - 2528^3 1^3 n u^3 + \\ & 2600^3 1^3 n^2 u^3 - 808^3 1^3 n^3 u^3 - 688^2 1^2 u^4 + 1520^2 1^2 n u^4 - \\ & 764^2 1^2 n^2 u^4 + 320^5 1^5 u^5 - 352^5 1^5 n u^5 - 64^6 u^6) / \\ & (1^6 (-2 + n)^2 (-1 + n)^2 n^2 (-2^2 1 + 1^2 n + 4^2 u)^2) \end{aligned}}{1}$	

Table S.2

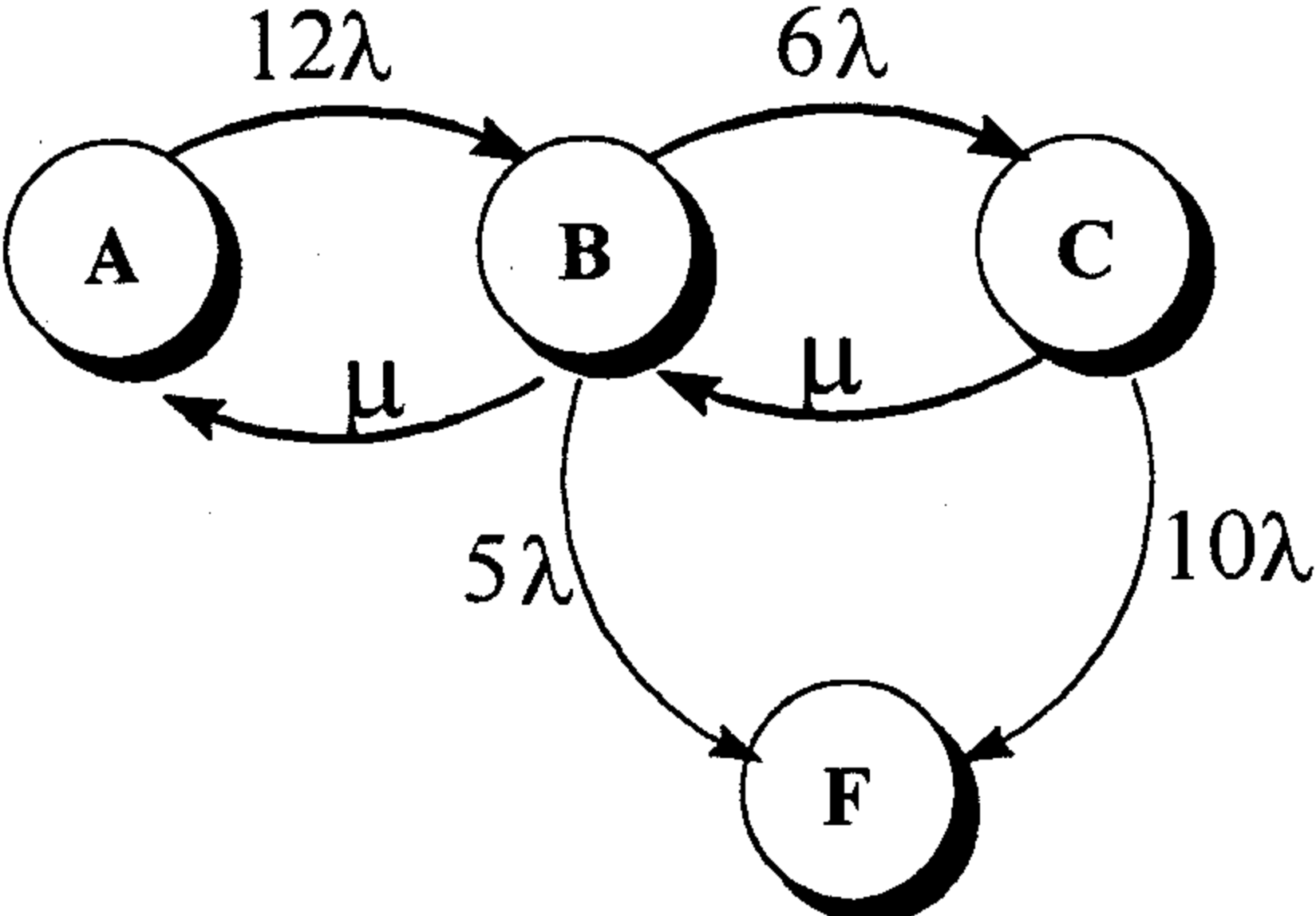
Text representation	2 DATUM stripe 4 with 6 disk each
<pre> BEGIN TRANSITION A B 12 1 B A u B C 6 1 C B u C F 10 1 B F 5 1 END TRANSITION BEGIN PROBABILITY A 1 B 0 C 0 END PROBABILITY </pre>	 <pre> graph LR A((A)) -- "12λ" --> B((B)) B -- "μ" --> A B -- "6λ" --> C((C)) C -- "μ" --> B C -- "10λ" --> F((F)) B -- "5λ" --> F </pre>
<pre> M = { { -(12 1), u, 0 } , { 12 1, -(u + 6 1 + 5 1), u } , { 0, 6 1, -(u + 10 1) } } VarM = { { S + (12 1), u, 0 } , { 12 1, S + (u + 6 1 + 5 1), u } , { 0, 6 1, S + (u + 10 1) } } OneMatrix = {1,1,1} VarOneMatrix = {{1},{1},{1}} Prob = {{1},{0},{0}} VarProb = {1,0,0} SecondMoment = (Limit[{-2 VarProb . (D[(Inverse[VarM]),S])},S->0]) . VarOneMatrix MTTDL = Simplify[- OneMatrix . Inverse[M] . Prob] Variance = Simplify[SecondMoment - MTTDL^2] </pre>	
$\text{MTTDL} = \frac{302 \lambda^2 + 27 \lambda u + u^2}{60 \lambda^2 (22 \lambda + u)}$	
$\text{Variance} = \frac{-67004 \lambda^4 - 11908 \lambda^3 u - 239 \lambda^2 u^2 + 10 \lambda u^3 + u^4}{3600 \lambda^4 (22 \lambda + u)^2}$	

Table S.3

Text representation	3 RAID Level 5 four disks each
<pre>BEGIN TRANSITION A B 12 1 B A u B C 8 1 B F 3 1 C B u C D 4 1 C F 6 1 D C u D F 9 1 END TRANSITION BEGIN PROBABILITY A 1 B 0 C 0 D 0 END PROBABILITY</pre>	
<pre>M = { { -(12 1), u, 0, 0 } , { 12 1, -(u + 8 1 + 3 1), u, 0 } , { 0, 8 1, -(u + 4 1 + 6 1), u } , { 0, 0, 4 1, -(u + 9 1) } } VarM = { { S + (12 1), u, 0, 0 } , { 12 1, S + (u + 8 1 + 3 1), u, 0 } , { 0, 8 1, S + (u + 4 1 + 6 1), u } , { 0, 0, 4 1, S + (u + 9 1) } } OneMatrix = {1,1,1,1} VarOneMatrix = {{1},{1},{1},{1}} Prob = {{1},{0},{0},{0}} VarProb = {1,0,0,0} SecondMoment = (Limit[(-2 VarProb . (D[(Inverse[VarM]),S])),S->0]) . VarOneMatrix MTTDL = Simplify[- OneMatrix . Inverse[M] . Prob] Variance = Simplify[SecondMoment - MTTDL^2]</pre>	
$MTTDL = \frac{3318 \lambda^3 + 459 \lambda^2 u + 30 \lambda u^2 + u^3}{36 \lambda^2 + (330 \lambda + 31 \lambda u + u^2)}$	
$Variance = \frac{-(9048924 \lambda^6 + 2499444 \lambda^5 u + 275499 \lambda^4 u^2 + 20424 \lambda^3 u^3 + 1056 \lambda^2 u^4 + 36 \lambda u^5 + u^6)}{(1296 \lambda^4 + (330 \lambda + 31 \lambda u + u^2)^2)}$	

tc4

7 Conclusion

Using MMS and *Mathematica* together will not only spare us from long and tedious task of solving MTTDL and Variance directly from Markov reliability model, but also allows us to obtain results in a fast and efficient manner. MMS provides a friendly user interface by providing error checking and having simple input file format that directly maps to Markov diagram. More importantly, efficient matrix operations and symbolic calculation capability of *Mathematica* are employed.

8 MMS Source Code

```

#-----#
# File Name: MMS.awk                                     #
# Description: This awk program converts Markov diagrams into mathematica #
# equations. Given input files with two sections: transitional and      #
# probability, this program outputs MTDL and Variance equations in      #
# mathematica notation.                                                #
# Usage:  awk -f MMS.awk <input_file_name>                          #
#                                               #
# University Of California, San Diego                                #
# Computer Science Department                                       #
# Designer:  Andrew H. Nguyen                                       #
# Course:   Storage System 228B                                     #
# Last Modified:  June 6, 1997                                       #
#-----#

BEGIN {
    error = 0
    numOfColumn = 0
    initRow = 0
    startProb = 0
    startTrans = 0
    probCounter = 0
    mIsEmpty = 1
}

1 == 1 { if (NF == 0) next }

/^[ \t]*[bB][eE][gG][iI][nN][ \t]+[pP][rR][oO][bB][aA][bB][iI][lL][iI][tT][yY]/ &&
(startProb == 0) {
    startProb = 1;next
}

/^[ \t]*[eE][nN][dD][ \t]+[pP][rR][oO][bB][aA][bB][iI][lL][iI][tT][yY]/ && (startProb ==
1) {
    startProb = 2;next
}

/^[ \t]*[bB][eE][gG][iI][nN][ \t]+[tT][rR][aA][nN][sS][iI][tT][iI][oO][nN]/ && (startTrans
== 0) {
    startTrans = 1;next
}

/^[ \t]*[eE][nN][dD][ \t]+[tT][rR][aA][nN][sS][iI][tT][iI][oO][nN]/ && (startTrans == 1) {
    startTrans = 2;next
}

startProb == 1 {
    if (NF < 2) {
        print "Error: Invalid probability entry."
        print "Line " NR ": " $0
        error = 1
        exit
    }
    if (probMatrix[$1] != "") {
        print "Error: Duplication of probability entry."
        print "Line " NR ": " $0
        error = 1
        exit
    }
    tempStr = $2
    for(i = 3;i<NF+1;i++)
        tempStr = tempStr " " $i;
    probMatrix[$1] = tempStr
    probCounter++
    next
}

startTrans == 1 {
    if (NF == 0) next
    if (NF < 3) {
        print "Error: Required at least three fields per line entry."
        print "Line " NR ": " $0
    }
}

```

```

        error = 1
        exit
    }

    if ($1 == $2) {
        print "Error: Source and destination states are the same."
        print "Line " NR ": " $0
        error = 1
        exit
    }

    if (Raw[$1$2] == 1) {
        print "Error: Duplicate transition entry."
        print "Line " NR ": " $0
        error = 1
        exit
    } else
        Raw[$1$2] = 1

    # TempStr = transition value
    TempStr = $3
    for (i = 4; i < NF+1; i++) { TempStr = TempStr " " $i }
    if (diag[$1] == "") diag[$1] = TempStr
    else diag[$1] = diag[$1] " + " TempStr

    if (diag1[$1] == "") diag1[$1] = TempStr
    else diag1[$1] = diag1[$1] " + " TempStr

    states[$1] = 1
    input[$1 $2] = TempStr
    source[$1] = 1
    dest[$2] = 1
    mIsEmpty = 0
}

END {
    if (error == 1) {exit}

    if (startTrans != 2)
        { print "Error: No transition section found.";exit}
    if (mIsEmpty == 1)
        { print "Error: No transtion entry found.";exit}

    ExistFail = 0
    for (item in dest) {
        if (source[item] != 1) { ExistFail++ }
    }

    if (ExistFail == 0)
        { print "Error: There exist no failure state.";exit }

    if (ExistFail > 1 )
        { print "Error: There exist multiple failure states.";exit }

    if (startProb == 0)
        { print "Error: No probability section found.";exit}

    for (item in diag) diag[item] = "-" (diag[item] " ")
    for (item in diag1) diag1[item] = "S + (" diag1[item] " )"

    numOfState = 0
    for (curState in states) numOfState++
    i = 0
    for (curState in states) {
        i++
        pValue = ""
        plValue = ""
        for (curColumn in states) {
            if (input[curColumn curState] == "") input[curColumn curState] = 0
            if (curState == curColumn) {
                pValue = pValue ", " diag[curState]
                plValue = plValue ", " diag1[curState]
            } else {
                pValue = pValue ", " input[curColumn curState]
                plValue = plValue ", " input[curColumn curState]
            }
        }
    }
}

```

```

    }
    }
    numOfColumn++
    M[numOfColumn] = "{ " substr(pValue,3,length(pValue)) " }"
    VarM[numOfColumn] = "{ " substr(p1Value,3,length(p1Value)) " }"
    P[numOfColumn] = curState

    if (probMatrix[P[numOfColumn]] == "") {
        print "Error: No probability value for state " P[numOfColumn]". " ;
        exit
    }
}

print "M = {"
print "\t "M[1]
for (i = 2; i<numOfColumn+1; i++) { print "\t,"M[i] }
print "    }"

print "VarM = {"
print "\t "VarM[1]
for (i = 2; i<numOfColumn+1; i++) { print "\t,"VarM[i] }
print "    }"

tempStr = "{1"
for (i=1;i<numOfState;i++) { tempStr = tempStr ",1"}
tempStr = tempStr "}"
print "OneMatrix = " tempStr

tempStr = ""
VarOneMatrix = ""
for (i =1;i<numOfColumn+1;i++) {
    if (tempStr == "") {
        tempStr = "{" probMatrix[P[i]] "}"
        VarOneMatrix = "{1}"
    } else {
        tempStr = tempStr ",{" probMatrix[P[i]] "}"
        VarOneMatrix = VarOneMatrix ",{1}"
    }
}
tempStr = "{" tempStr "}"
VarOneMatrix = "{" VarOneMatrix "}"
print "VarOneMatrix = " VarOneMatrix

print "Prob = " tempStr

tempStr = ""
tempStr = probMatrix[P[1]]
for (i =2;i<numOfColumn+1;i++) {
    tempStr = tempStr "," probMatrix[P[i]]
}
tempStr = "{" tempStr "}"
print "VarProb = " tempStr
print "SecondMomment = (Limit[(-2 VarProb . (D[(Inverse[VarM]),S])),S->0]) .
VarOneMatrix"
print "MTTDL = Simplify[- OneMatrix . Inverse[M] . Prob]"
print "Variance = Simplify[SecondMomment - MTTDL^2]"
}

```