

# Process algebras for quantitative analysis: Lecture 7 — Scalable Analysis of Scalable Systems

Stephen Gilmore

School of Informatics  
The University of Edinburgh  
Scotland

Joint work with Allan Clark and Mirco Tribastone

# Scalable analysis of scalable systems



**The Register**  
*Biting the hand that feeds IT*

Hardware Software Music & Media **Networks** Security Public Sector Business Science O

VoIP Wireless Mobile **Telecoms**

## Google blames Gmail outage on data centre collapse Domino effect crashes through the cloud

By [Kelly Fiveash](#) • [Get more from this author](#)

Posted in [Telecoms](#), 25th February 2009 11:44 GMT

Google has apologised for yesterday's major Gmail meltdown after some of its data centres in Europe failed to cope with a routine maintenance event.

The company's web-based email service was [flat as a pancake](#) for about three hours yesterday morning as Mountain View engineers attempted to fix the problem, which affected vast swathes of the firm's 113m strong global user base.

# Scalable analysis of scalable systems

**The Register**  
*Biting the hand that feeds IT*

Hardware Software Music & Media **Networks** Security Public Sector Business Science O

VoIP Wireless Mobile **Telecoms**

Goog  
Domir  
By **Kell**  
Posted i  
Google  
in Euro  
The cor  
yesterd  
vast sw

**The Register**  
*Biting the hand that feeds IT*

Hardware Software Music & Media Networks Security Public Sector Business Science O

Operating Systems **Applications** Developer Microbite

## Google's cloud bursts yet again Scattered outages for Gmail users

By **Kelly Fiveash** • [Get more from this author](#)  
Posted in [Applications](#), 10th March 2009 11:41 GMT

Google has admitted that some Gmail users cannot currently access their email just weeks after the company suffered a major outage following a technical cockup at one of its European data centres.

It confirmed that a "small subset of users" have been affected by the latest downtime, but didn't provide a definitive number and could not confirm when the service would be back up and running.

It's also unclear if only individuals who use Gmail for free are affected, or if business customers who pay for the service are also being hit by Google's latest embarrassing

£ 1 0 0

↻ 🔍 ↺

# Scalable analysis of scalable systems

The screenshot shows a web browser displaying a PCWorld article. The page has a red header with the PCWorld logo and a search bar. Below the header is a navigation menu with links for Home, News, Reviews, How-To, Blogs, Videos, Downloads, and Shop & Co. The main content area features the article title "Microsoft Blames Azure Outage on OS Upgrade" in large black text. The author is Elizabeth Montalbano, IDG News Service, and the article is dated Wednesday, March 18, 2009 1:40 PM PDT. The article text begins with "Microsoft is blaming a routine OS upgrade for an outage that hit its Windows Azure cloud-computing infrastructure over the weekend." Below the main text, there is a section titled "PEOPLE WHO READ THIS ALSO READ:" with two related article thumbnails. At the bottom of the page, there is a blue banner with four red circular icons containing the symbols £, 1, 0, and 0. The page also includes a sidebar on the left with various navigation links and a right sidebar with a thumbs-up icon and a comment count of 0.

**The PCWorld** Search PC World Search Browse

Hardware Software  
VoIP Wireless

Home News Reviews How-To Blogs Videos Downloads Shop & Co

**PCW Business Center** Smart Technology for Smart Companies

TOPICS: [Software / Services](#) [Office Hardware](#) [Security](#) [Servers / Storage](#) [Cell Phones / VoIP](#) [Operating Sy](#)

PC World » Business Center » Software / Services » News

## Microsoft Blames Azure Outage on OS Upgrade

By **Kelly F**  
Posted in A

Wednesday, March 18, 2009 1:40 PM PDT

Microsoft is blaming a routine OS upgrade for [an outage](#) that hit its Windows Azure cloud-computing infrastructure over the weekend.

In a post on its [Windows Azure blog](#), Microsoft said after the upgrade "the deployment service within Windows Azure began to slow down due to networking issues" on Friday. "This caused a large number of servers to time out and fail," which brought some applications down, according to the post.

0 Comments

PEOPLE WHO READ THIS ALSO READ:

- [Fujitsu Launches Color E-Paper Terminal: Bad News for Kindle?](#)
- [File Integrity: Windows Still Fails at the Most Basic Task](#)

£ 1 0 0

# Scalable analysis of scalable systems

**The Register**

**PCWorld** Search PC World Search Brow

Hardware Software  
VoIP Wireless

Goog  
Domir  
By **Kell**  
Posted i  
Google  
in Euro  
The cor  
yesterd  
vast sw

**The**

Hardware  
Operat

Google  
Scattere  
By **Kelly F**  
Posted in A  
Google ha  
after the c  
European  
It confirme  
didn't prov  
and runni  
It's also unclear if  
Gmail for free are  
customers who pa  
being hit by Googl

**PCWorld**

Home

OFFICE OF  
**News & Communications**  
DUKE UNIVERSITY

Subscribe to News: [RSS](#) |  
[email newsletters](#)

TOPIC

- Resources for media
- Find a Duke expert
- Contact us
- Quick facts about Duke
- For Duke community
- Multimedia services
- Subscribe to news

PC  
M  
O  
Eliza  
Wed  
Micr  
Azur  
In a  
depl  
netw  
out

**A detailed look at last week's system outage**

Thursday, March 12, 2009

print | email | digg | del.icio.us

Much of the university experienced a major system outage between March 4 and 8, causing the Office of Information Technology to suspend some less-critical services, like streaming media, until systems were fully restored. Klara Jelinkova, Duke's senior director of shared services and infrastructure, answers questions about what happened during the outage.

**What caused the outage?**

Duke generates large amounts of data -- data collected in research and captured in lectures, photographs, web archives, personal files and so on -- all of which must be instantly accessible. Increasingly, institutions handle data like this through shared storage. Instead of giving each service a disk of its own (for instance, a server for Duke Today, another server for one department's shared files), we use high-capacity storage arrays to hold all the data and recall it for users on demand. These disk arrays are designed to allow for flexibility in providing the right amount for storage for each service as needed, and providing redundancy by making copies of the data to multiple disks.

br

# Scalable analysis of scalable systems

- Computer systems are currently deployed in active use with no real assurance of graceful degradation of service as the user population grows.

# Scalable analysis of scalable systems

- Computer systems are currently deployed in active use with no real assurance of graceful degradation of service as the user population grows.
- Because we cannot rely on them to provide service in times of greatest need, such systems are as unreliable in practice as ones which contain programming errors.

# Outline

- 1 Quantitative Analysis
- 2 Scalability and Large-Scale Systems
- 3 Case Study

# Outline

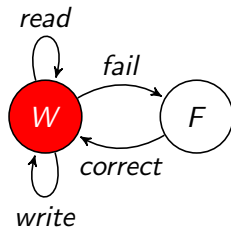
- 1 Quantitative Analysis
- 2 Scalability and Large-Scale Systems
- 3 Case Study

## Disk model in SRMC

```
DiskA::{  
  Working = (read, r).Working  
           + (write, w).Working  
           + (fail, f).Failed;  
  Failed  = (correct, c).Working;  
};
```

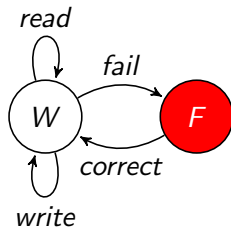
# Disk model in SRMC

```
DiskA::{\n  Working = (read, r).Working\n          + (write, w).Working\n          + (fail, f).Failed;\n  Failed  = (correct, c).Working;\n};
```



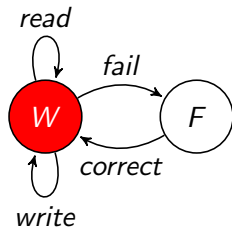
# Disk model in SRMC

```
DiskA::{\n  Working = (read, r).Working\n          + (write, w).Working\n          + (fail, f).Failed;\n  Failed  = (correct, c).Working;\n};
```



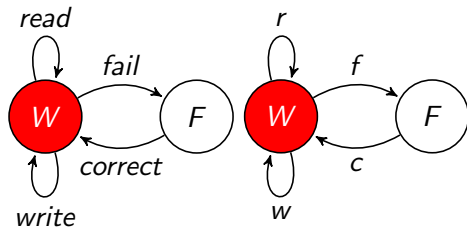
# Disk model in SRMC

```
DiskA::{\n  Working = (read, r).Working\n          + (write, w).Working\n          + (fail, f).Failed;\n  Failed  = (correct, c).Working;\n};
```



# Disk model in SRMC

```
DiskA::{\n  Working = (read, r).Working\n          + (write, w).Working\n          + (fail, f).Failed;\n  Failed  = (correct, c).Working;\n};
```

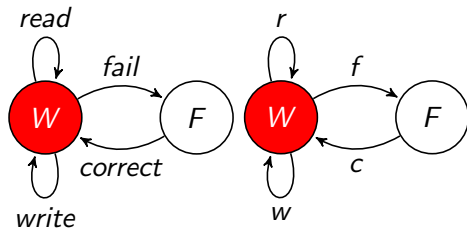


# Disk model in SRMC

```

DiskA::{
  Working = (read, r).Working
          + (write, w).Working
          + (fail, f).Failed;
  Failed  = (correct, c).Working;
};

```



Rate matrix

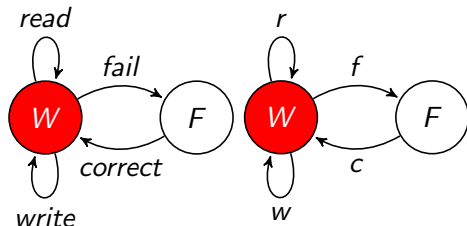
$$R = \begin{pmatrix} & f \\ c & \end{pmatrix}$$

# Disk model in SRMC

```

DiskA::{
  Working = (read, r).Working
          + (write, w).Working
          + (fail, f).Failed;
  Failed  = (correct, c).Working;
};

```



Generator matrix

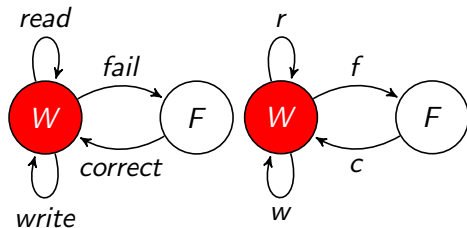
$$Q = \begin{pmatrix} -f & f \\ c & -c \end{pmatrix}$$

# Disk model in SRMC

```

DiskA::{
  Working = (read, r).Working
          + (write, w).Working
          + (fail, f).Failed;
  Failed  = (correct, c).Working;
};

```



Global balance equation

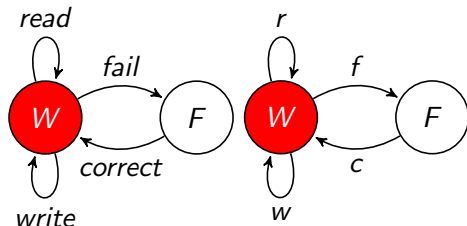
$$\begin{pmatrix} \pi_W \\ \pi_F \end{pmatrix} \begin{pmatrix} -f & f \\ c & -c \end{pmatrix} = 0$$

# Disk model in SRMC

```

DiskA::{
  Working = (read, r).Working
          + (write, w).Working
          + (fail, f).Failed;
  Failed  = (correct, c).Working;
};

```



Normalisation

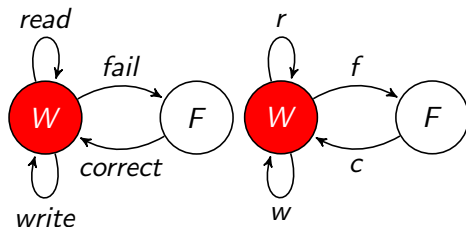
$$\pi_W + \pi_F = 1$$

# Disk model in SRMC

```

DiskA::{
  Working = (read, r).Working
          + (write, w).Working
          + (fail, f).Failed;
  Failed  = (correct, c).Working;
};

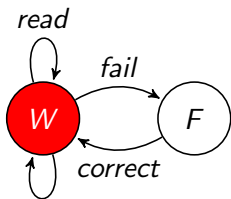
```



Global balance solution

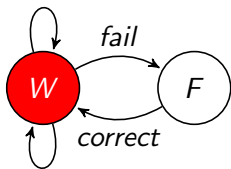
$$\begin{pmatrix} \pi_W \\ \pi_F \end{pmatrix} = \begin{pmatrix} \frac{c}{f+c} \\ \frac{f}{f+c} \end{pmatrix}$$

# Six disks

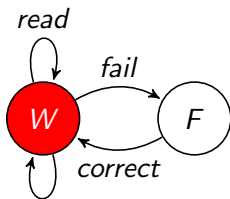


write

read

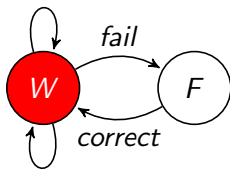


write

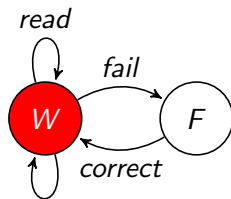


write

read

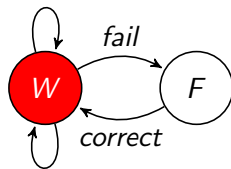


write



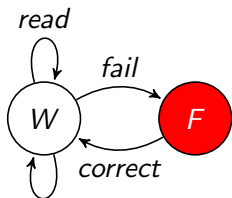
write

read



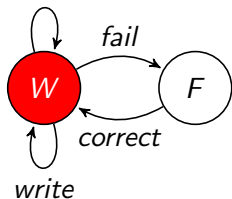
write

## Six disks

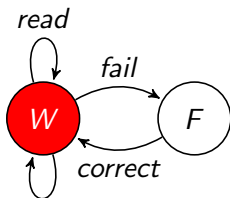


write

read

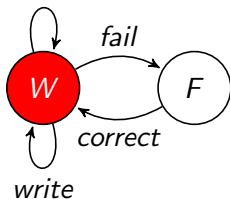


write

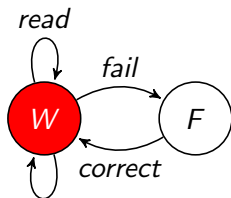


write

read

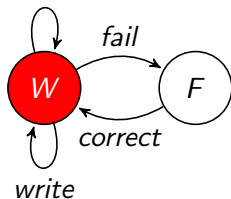


write



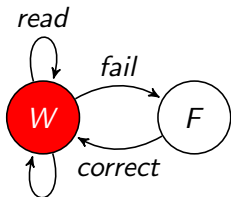
write

read



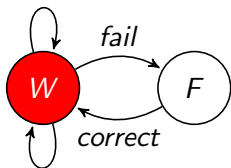
write

# Six disks

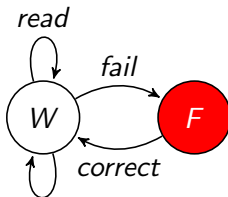


write

read

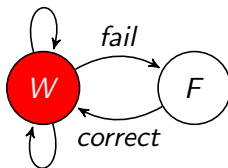


write

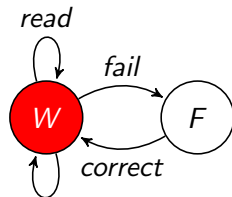


write

read

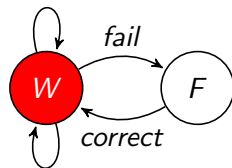


write



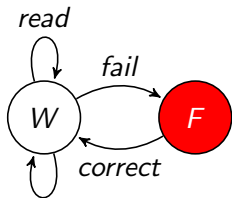
write

read



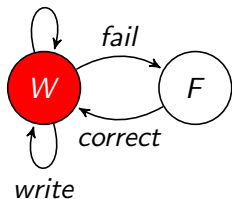
write

# Six disks

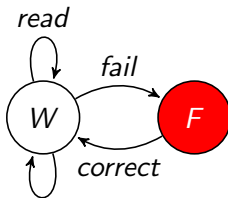


write

read

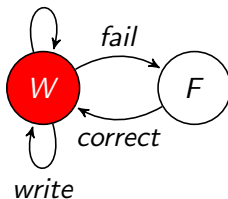


write

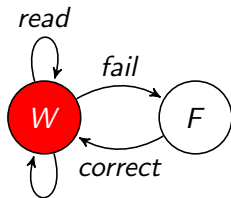


write

read

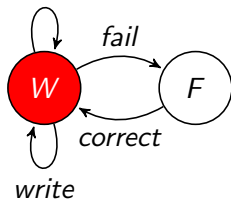


write



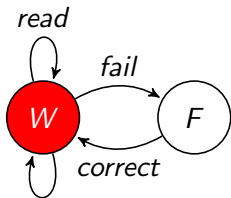
write

read



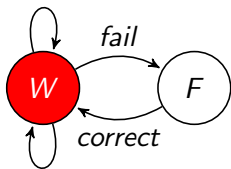
write

## Six disks

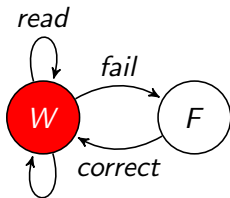


write

read

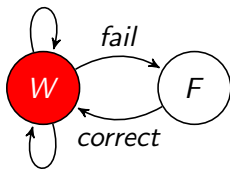


write

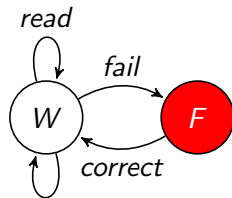


write

read

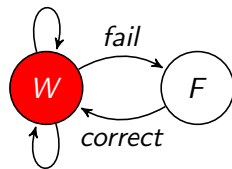


write



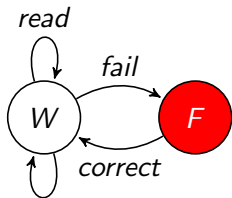
write

read



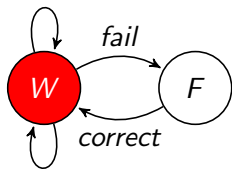
write

## Six disks

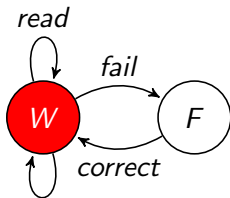


write

read

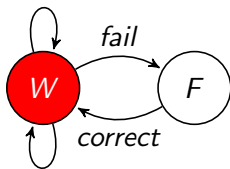


write

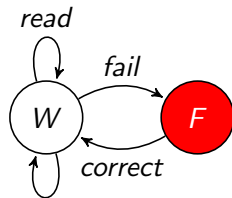


write

read

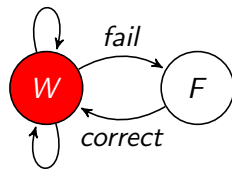


write



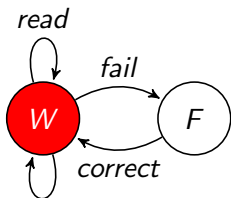
write

read



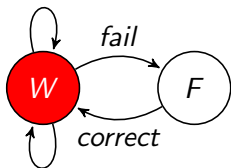
write

# Six disks

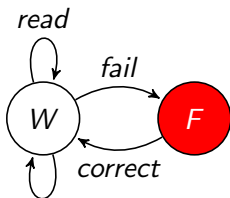


write

read

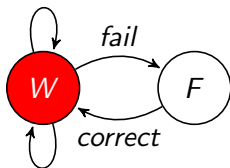


write

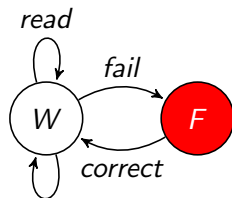


write

read

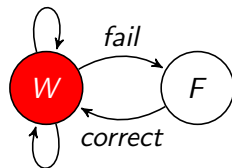


write



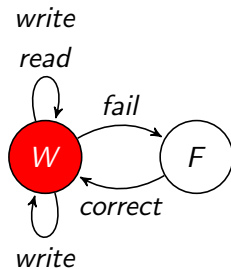
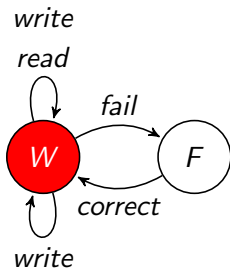
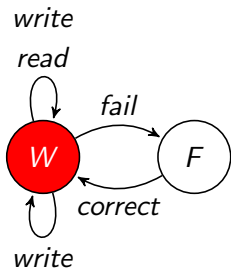
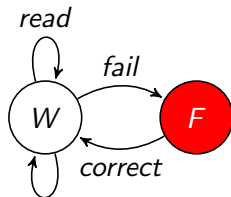
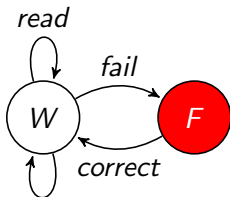
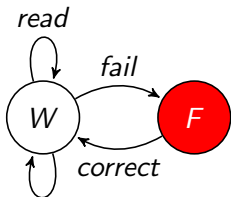
write

read

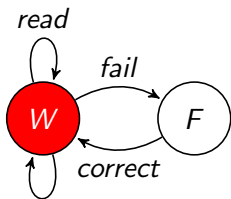


write

# Six disks

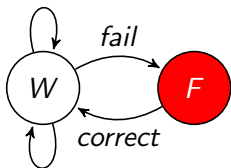


## Six disks

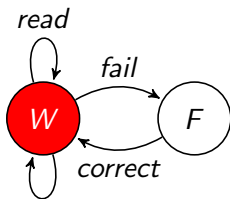


write

read

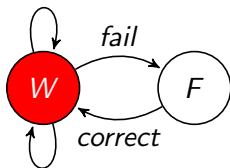


write

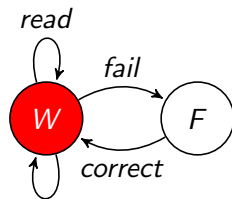


write

read

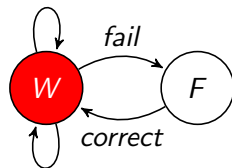


write



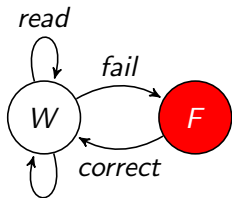
write

read



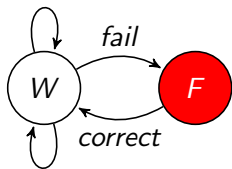
write

# Six disks

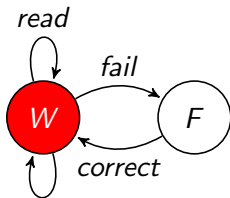


write

read

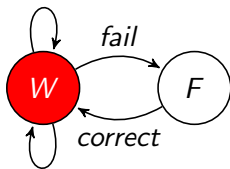


write

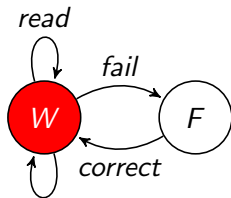


write

read

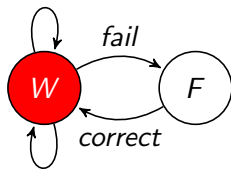


write



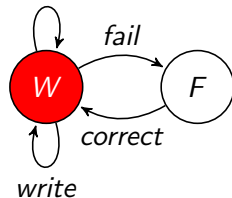
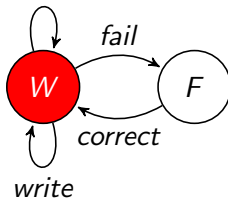
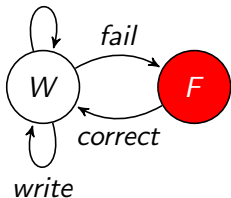
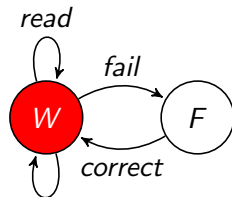
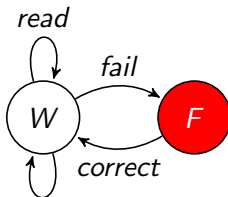
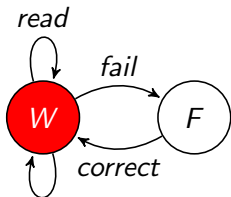
write

read

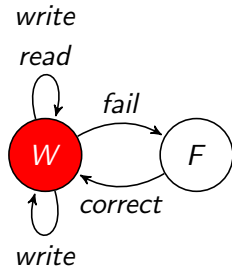
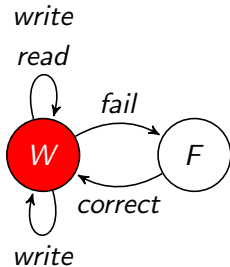
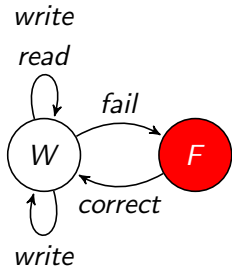
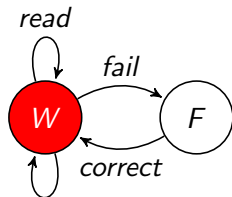
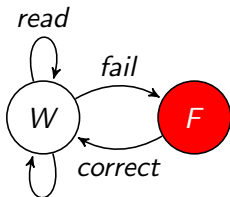
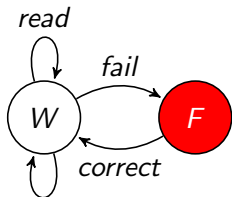


write

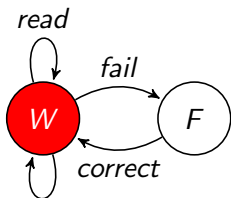
# Six disks



# Six disks

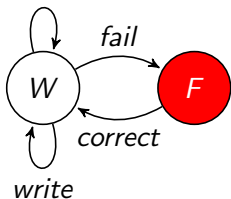


# Six disks

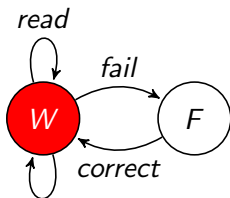


write

read

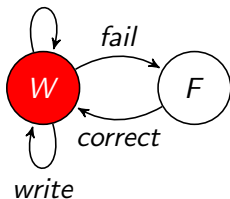


write

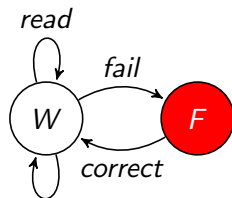


write

read

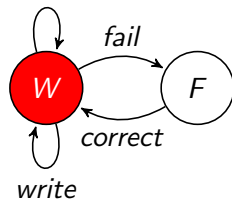


write



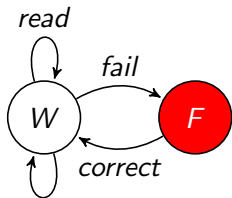
write

read



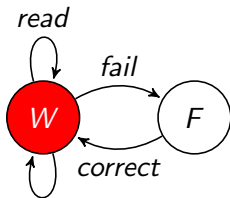
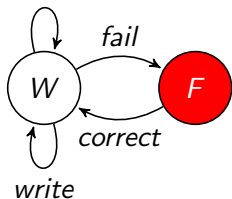
write

# Six disks



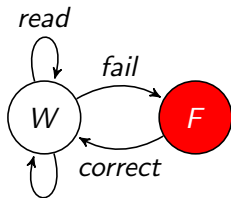
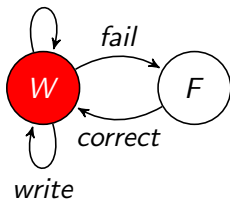
write

read



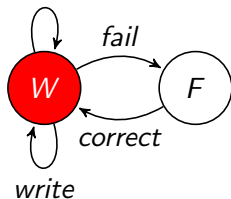
write

read

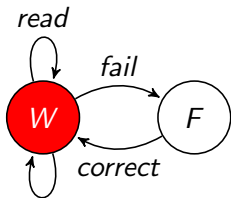


write

read

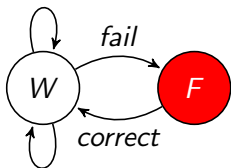


# Six disks

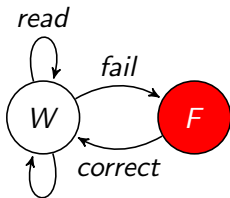


write

read

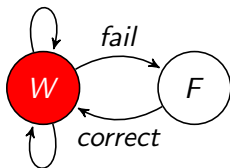


write

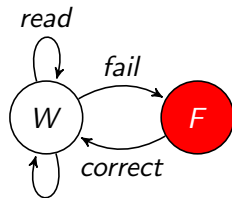


write

read

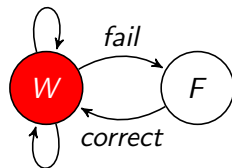


write



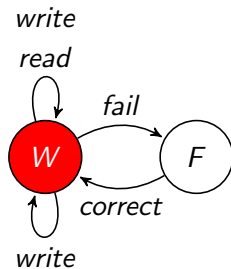
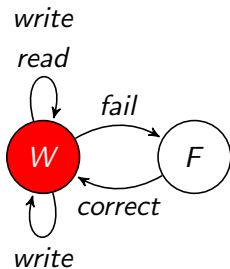
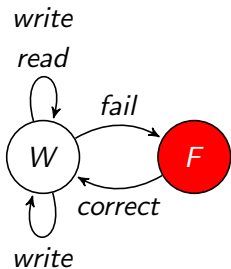
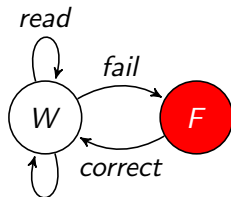
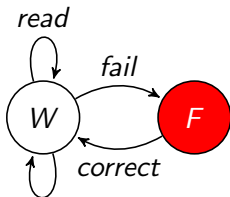
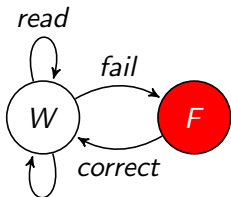
write

read

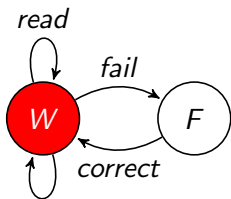


write

## Six disks

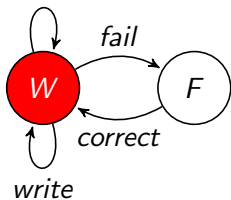


## Six disks

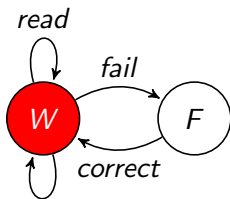


write

read

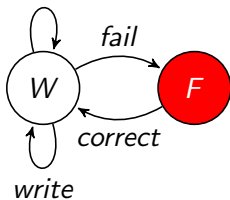


write

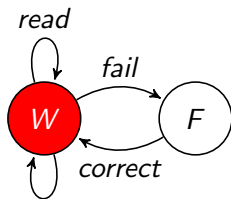


write

read

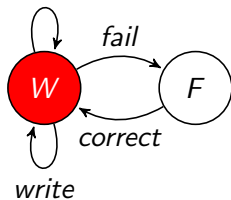


write



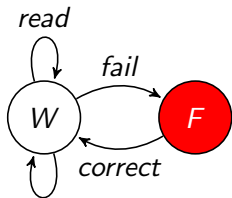
write

read



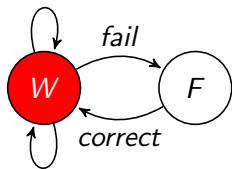
write

# Six disks

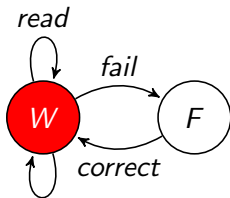


write

read

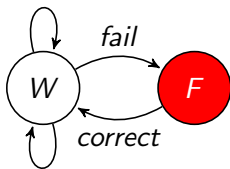


write

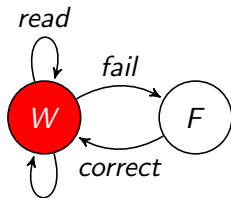


write

read

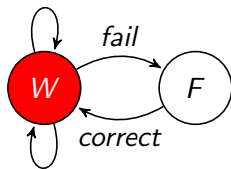


write



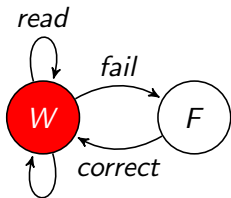
write

read



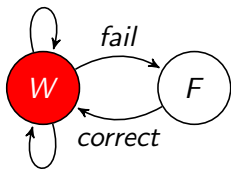
write

## Six disks

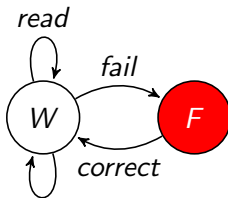


write

read

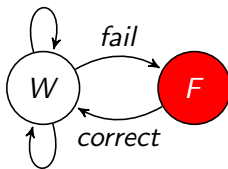


write

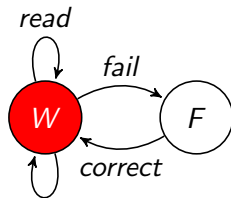


write

read

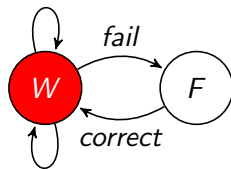


write



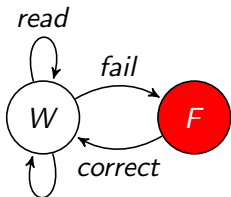
write

read



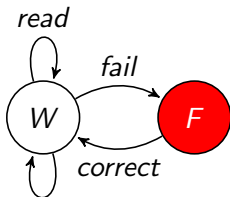
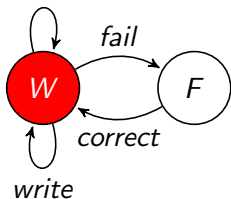
write

## Six disks



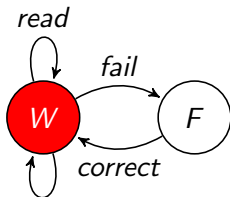
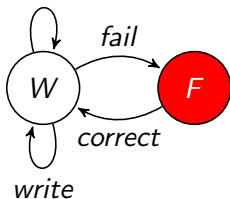
write

read



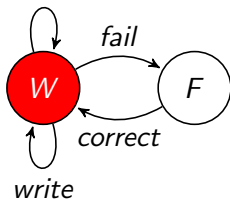
write

read

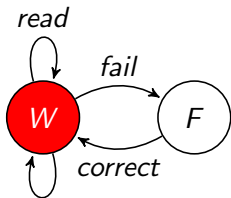


write

read

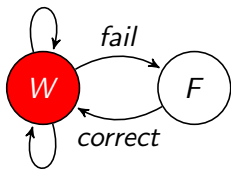


## Six disks

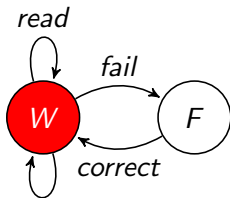


write

read

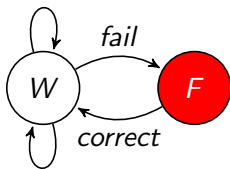


write

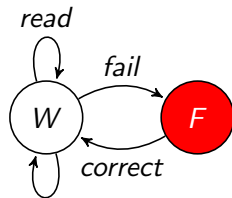


write

read

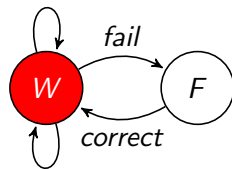


write



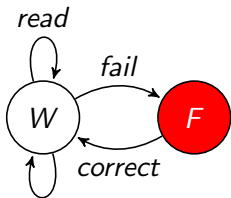
write

read



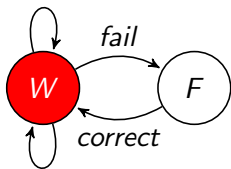
write

## Six disks

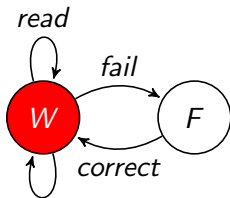


write

read

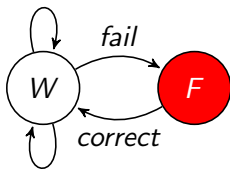


write

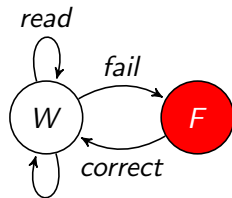


write

read

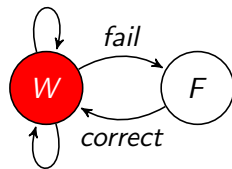


write



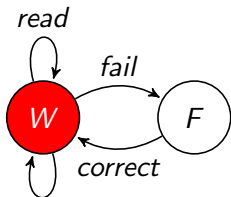
write

read



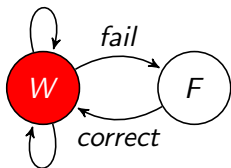
write

## Six disks

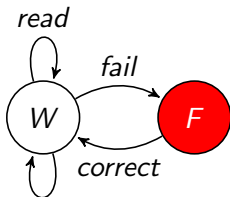


write

read

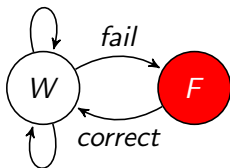


write

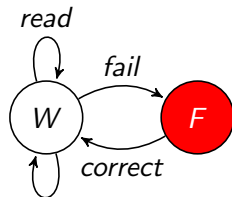


write

read

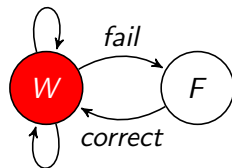


write



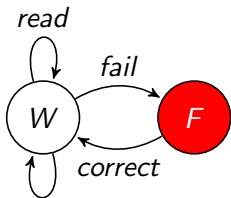
write

read



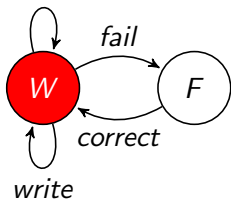
write

# Six disks

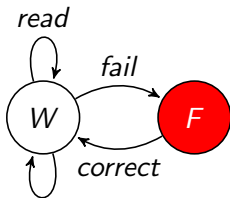


write

read

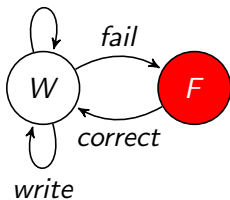


write

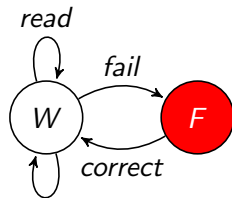


write

read

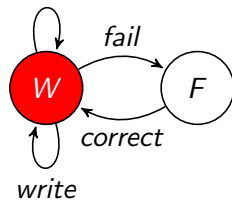


write



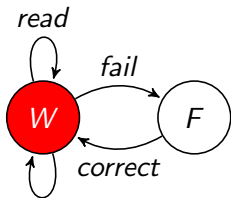
write

read



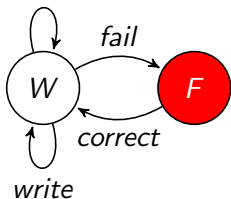
write

# Six disks

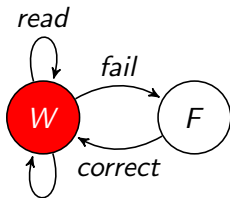


write

read

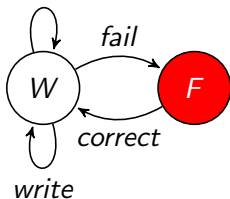


write

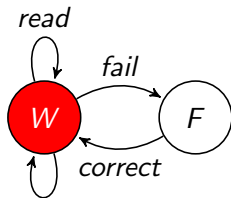


write

read

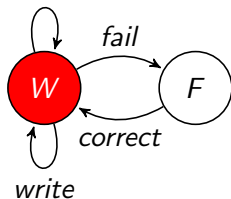


write



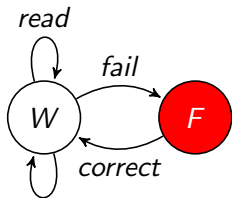
write

read



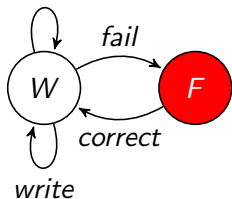
write

## Six disks

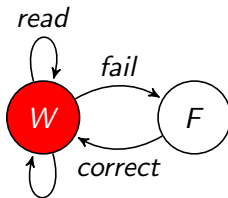


write

read

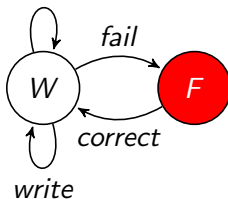


write

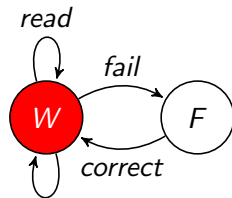


write

read

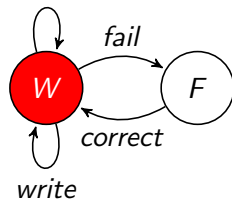


write



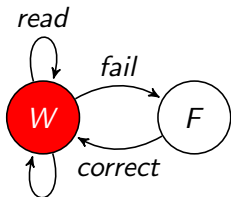
write

read



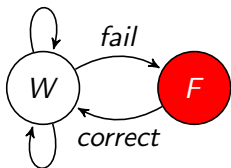
write

# Six disks

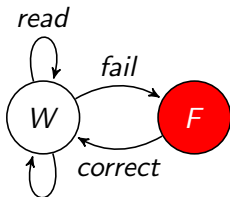


write

read

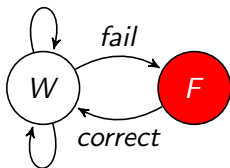


write

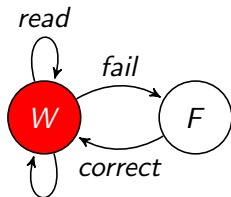


write

read

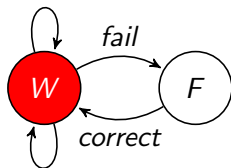


write



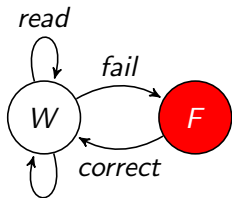
write

read



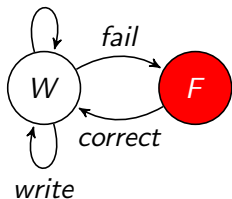
write

# Six disks

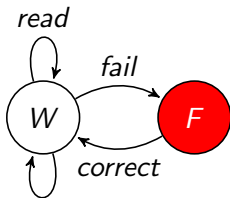


write

read

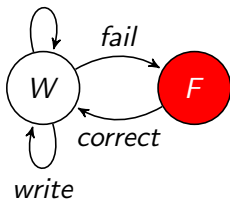


write

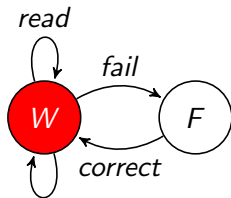


write

read

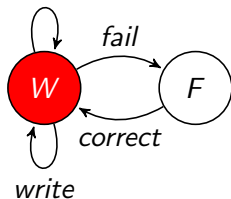


write



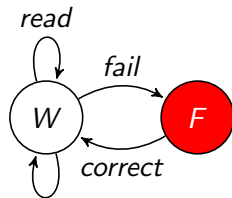
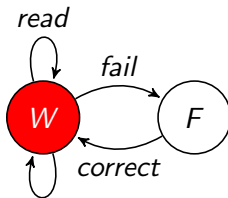
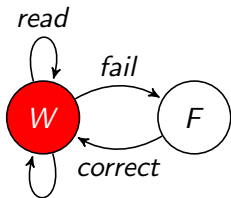
write

read



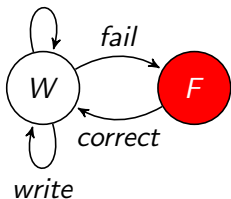
write

## Six disks



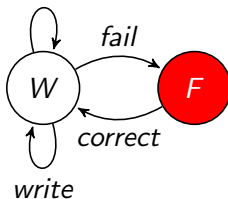
write

read



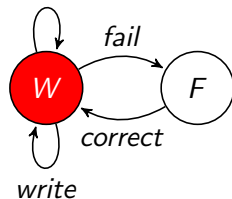
write

read



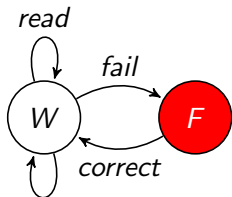
write

read



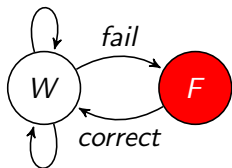
write

## Six disks

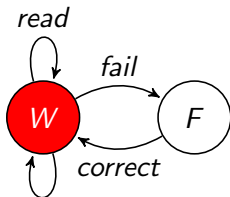


write

read

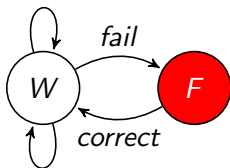


write

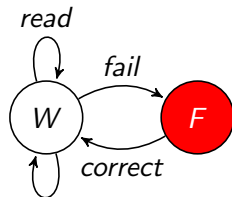


write

read

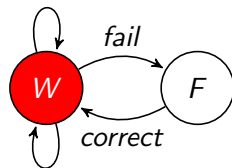


write



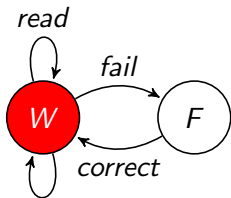
write

read



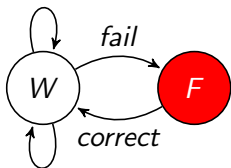
write

# Six disks

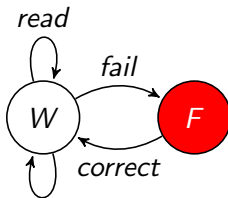


write

read

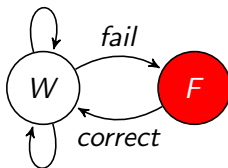


write

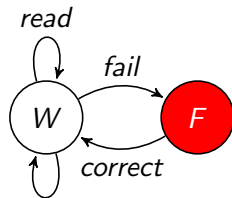


write

read

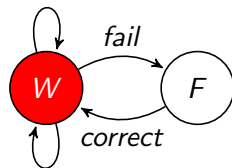


write



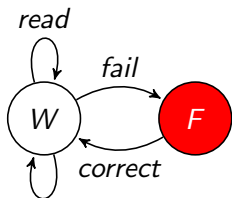
write

read



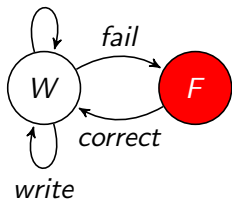
write

## Six disks

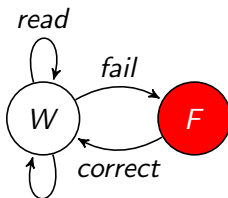


write

read

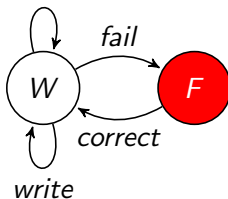


write

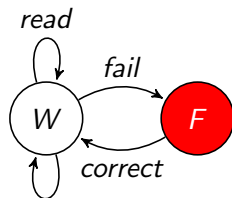


write

read

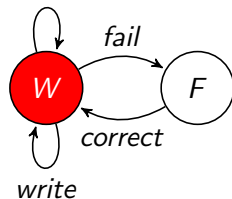


write



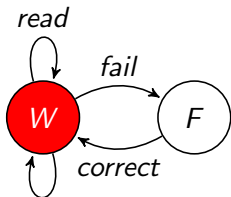
write

read



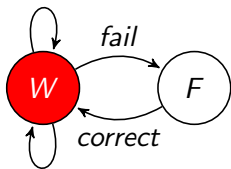
write

## Six disks

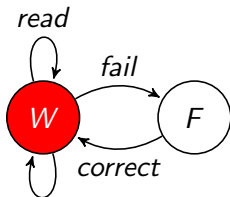


write

read

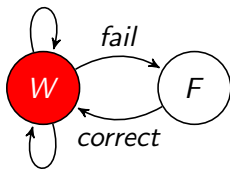


write

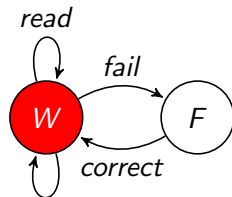


write

read

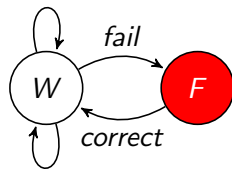


write



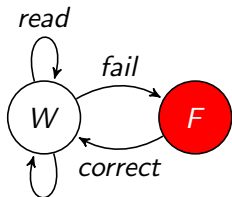
write

read



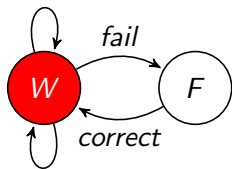
write

## Six disks

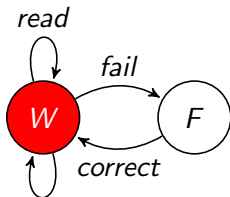


write

read

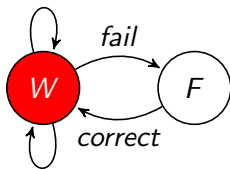


write

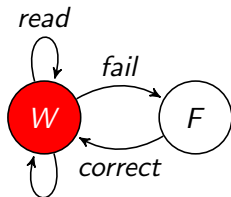


write

read

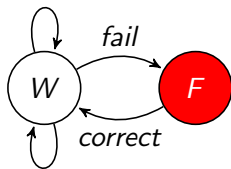


write



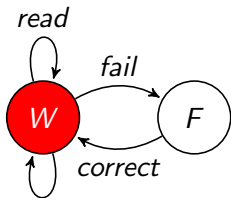
write

read



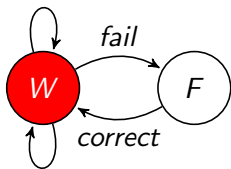
write

## Six disks

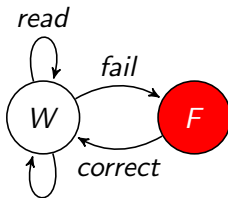


write

read

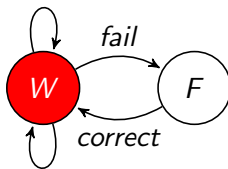


write

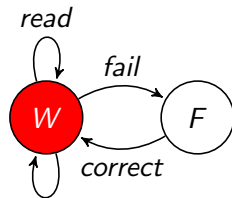


write

read

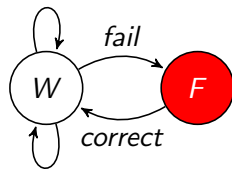


write



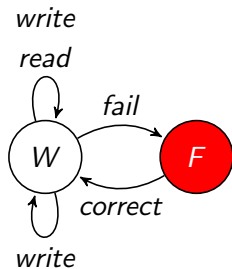
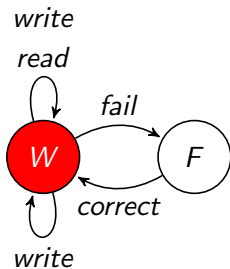
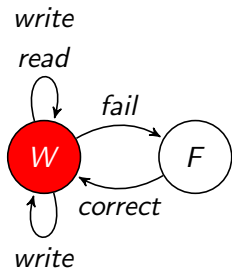
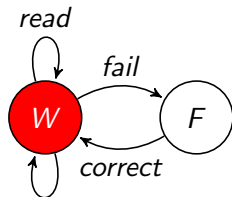
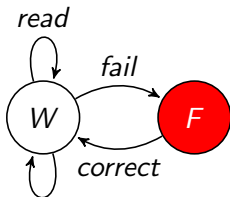
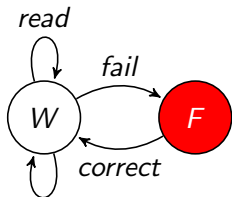
write

read

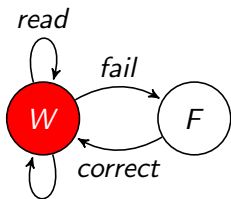


write

## Six disks

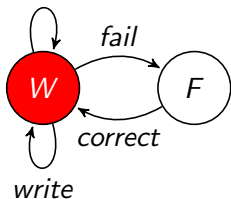


## Six disks

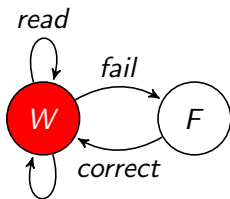


write

read

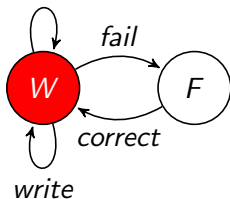


write

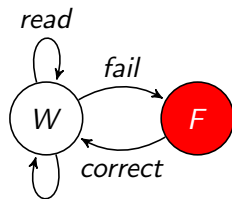


write

read

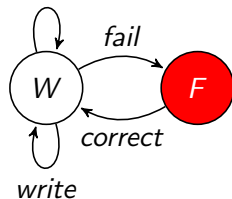


write



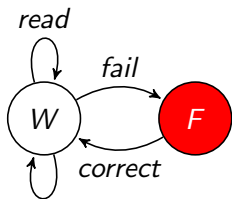
write

read



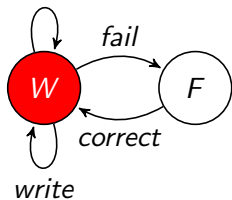
write

## Six disks

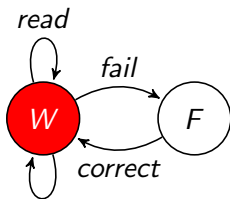


write

read

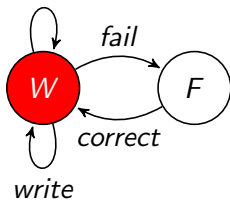


write

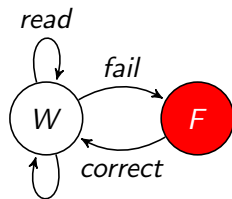


write

read

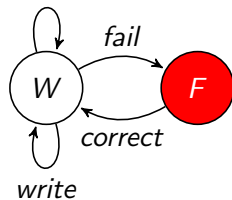


write



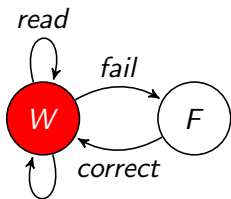
write

read



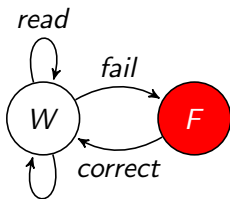
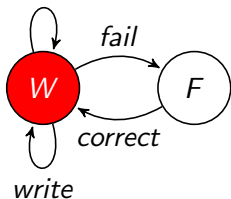
write

## Six disks



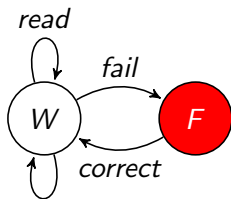
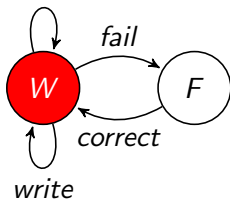
write

read



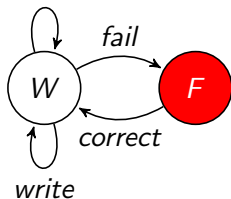
write

read

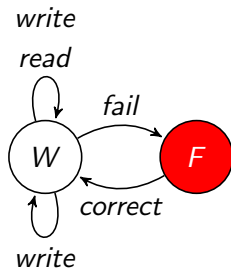
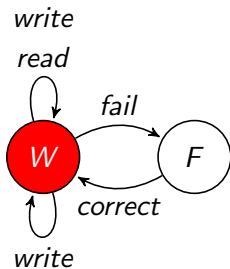
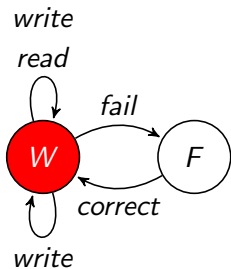
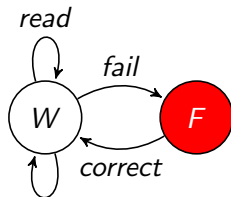
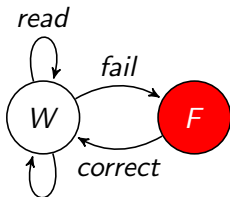
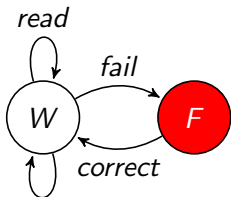


write

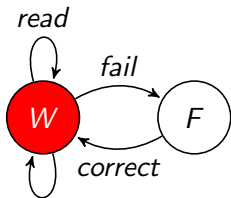
read



# Six disks

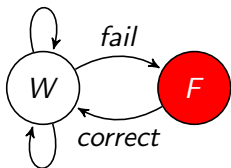


# Six disks

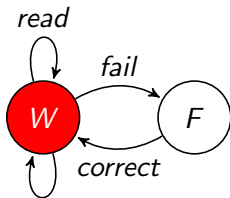


write

read

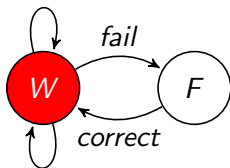


write

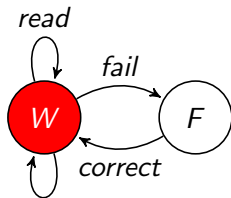


write

read

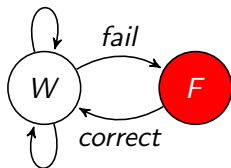


write



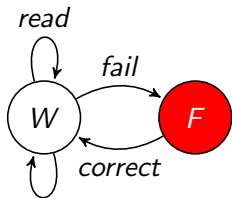
write

read



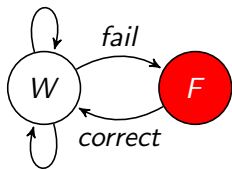
write

## Six disks

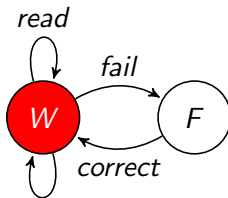


write

read

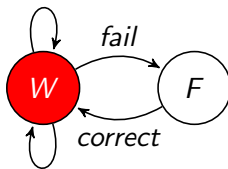


write

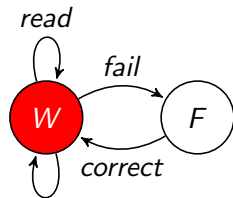


write

read

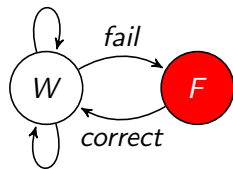


write



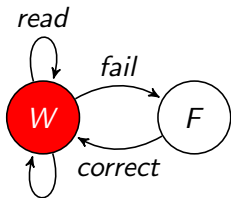
write

read



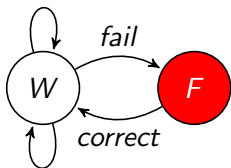
write

## Six disks

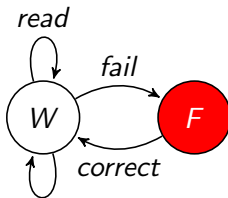


write

read

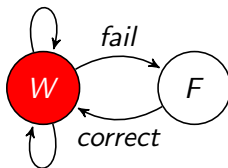


write

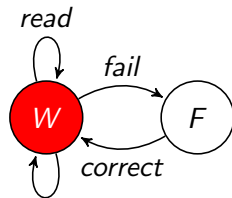


write

read

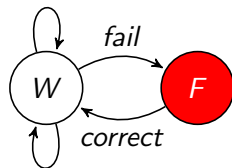


write



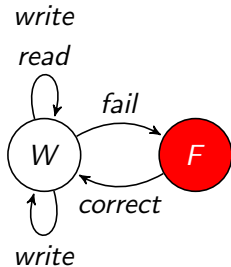
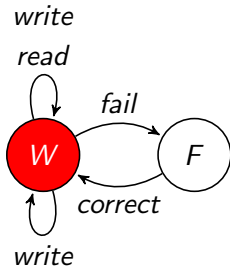
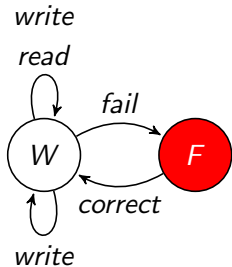
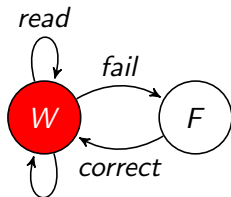
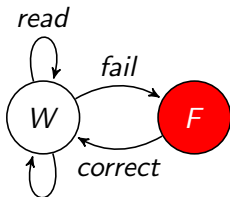
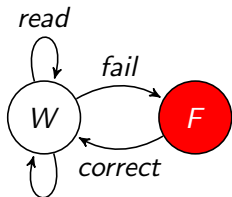
write

read

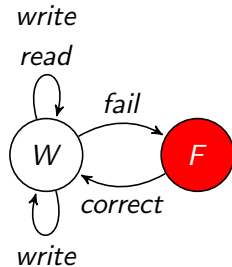
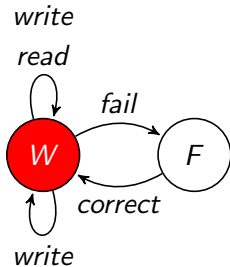
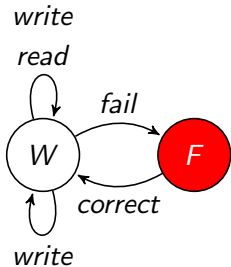
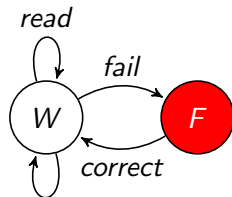
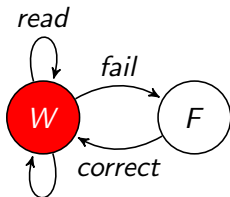
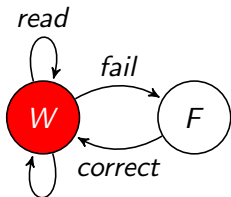


write

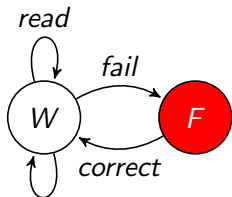
## Six disks



# Six disks

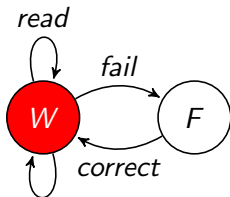
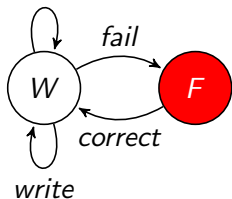


## Six disks



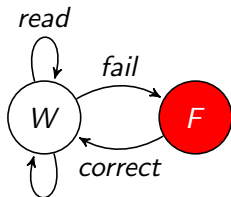
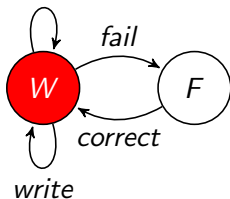
write

read



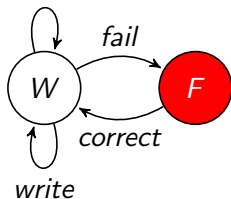
write

read

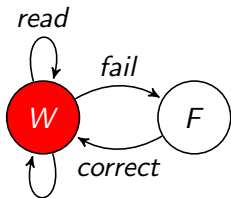


write

read

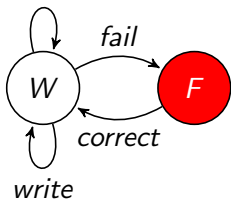


# Six disks

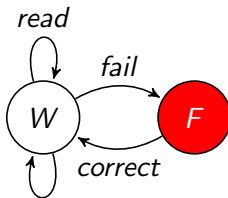


write

read

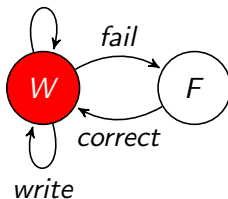


write

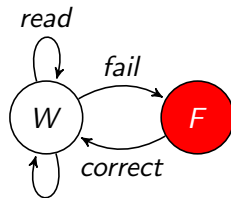


write

read

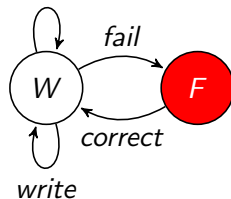


write



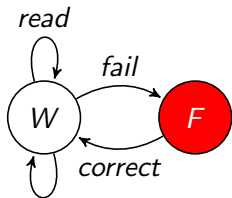
write

read



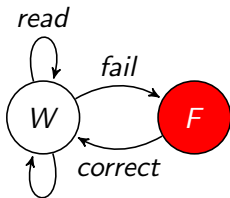
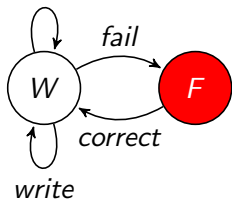
write

## Six disks



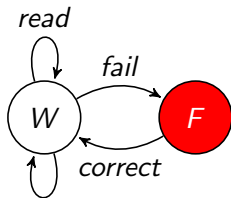
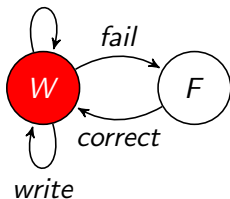
write

read



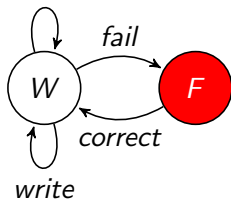
write

read

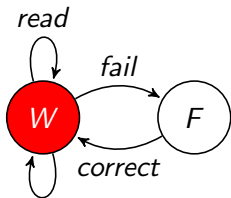


write

read

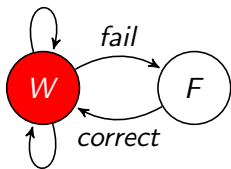


# Six disks

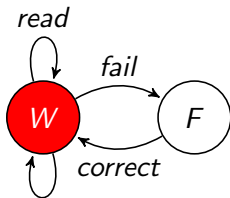


write

read

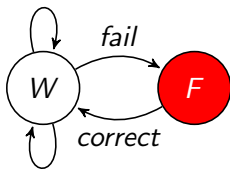


write

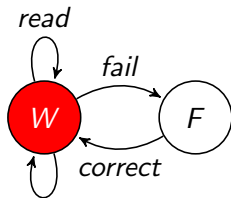


write

read

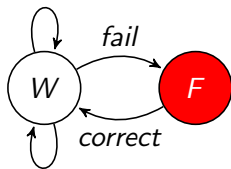


write



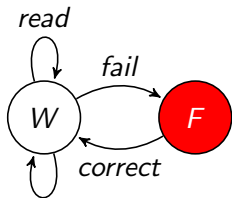
write

read



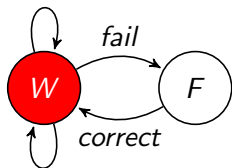
write

## Six disks

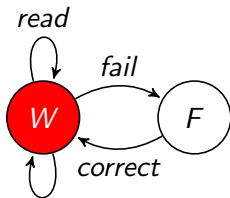


write

read

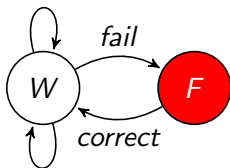


write

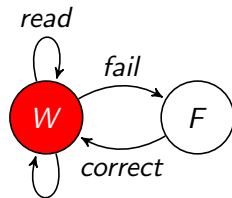


write

read

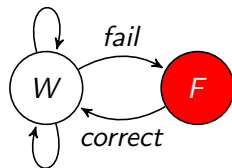


write



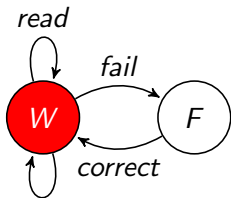
write

read



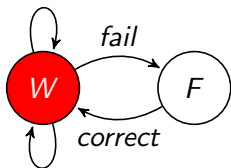
write

# Six disks

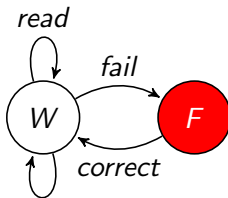


write

read

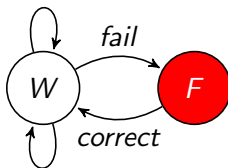


write

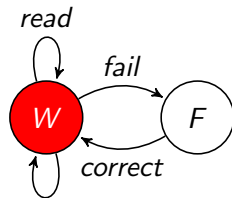


write

read

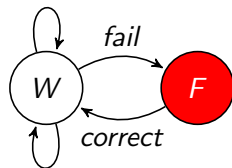


write



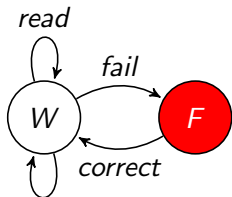
write

read



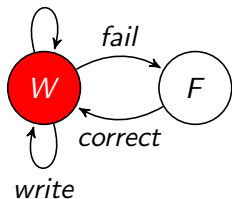
write

## Six disks

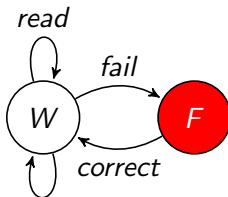


write

read

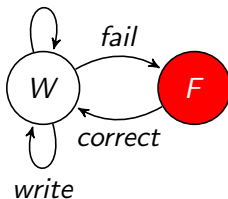


write

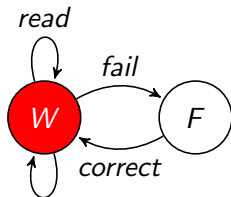


write

read

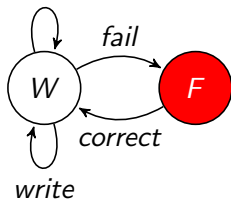


write



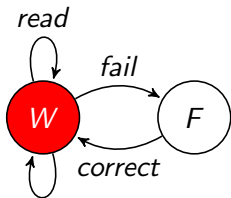
write

read



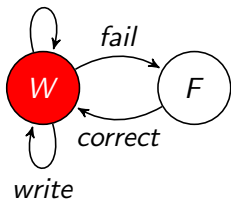
write

## Six disks

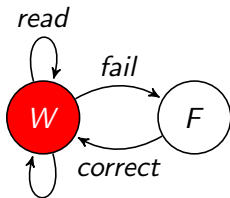


write

read

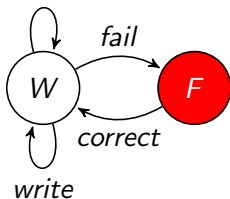


write

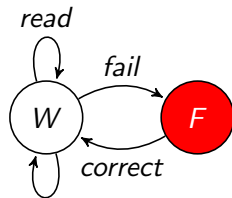


write

read

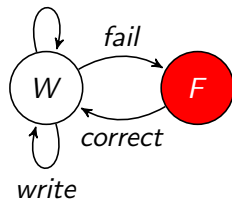


write



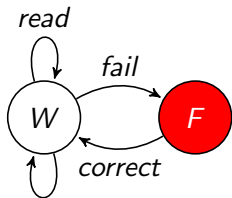
write

read



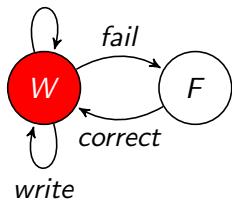
write

## Six disks

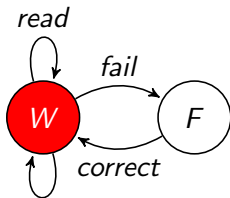


write

read

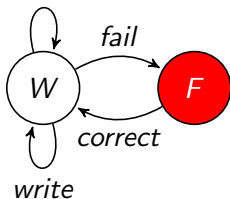


write

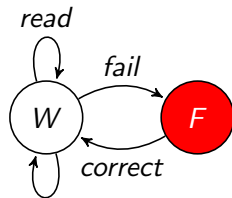


write

read

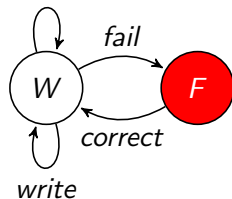


write



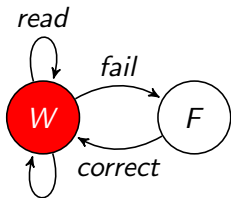
write

read



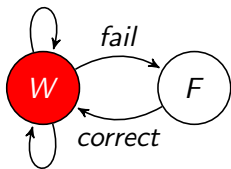
write

## Six disks

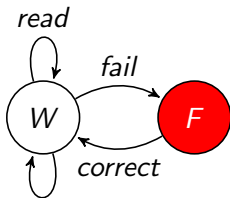


write

read

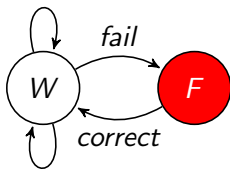


write

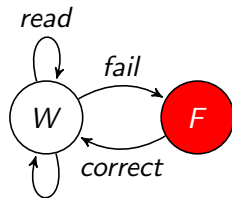


write

read

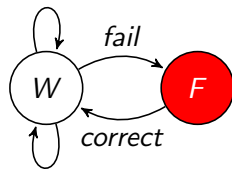


write



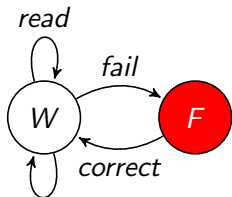
write

read



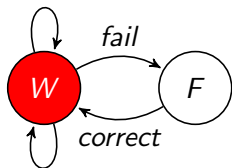
write

## Six disks

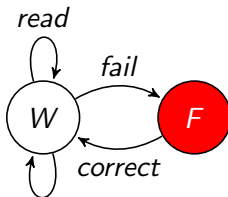


write

read

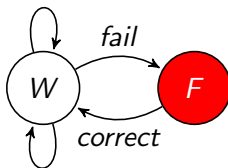


write

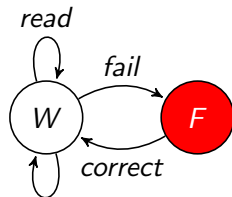


write

read

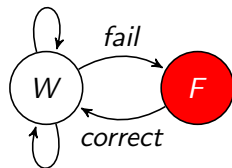


write



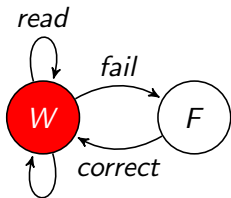
write

read



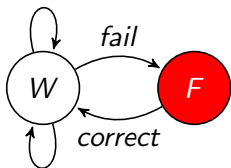
write

## Six disks

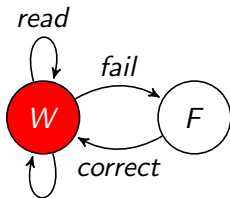


write

read

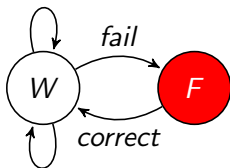


write

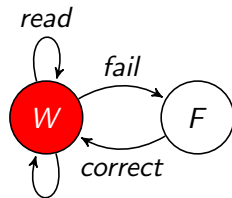


write

read

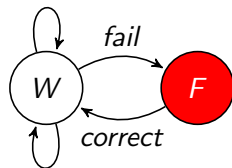


write



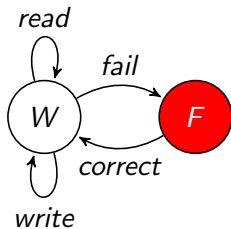
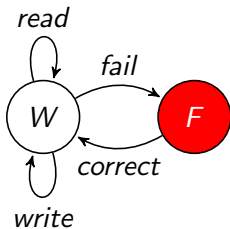
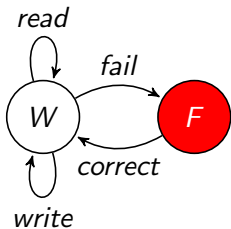
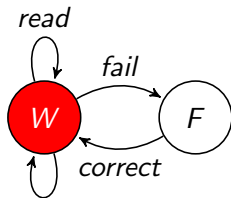
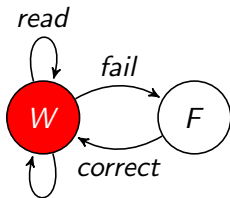
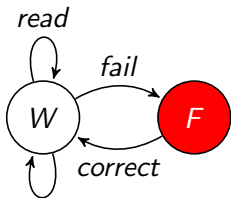
write

read

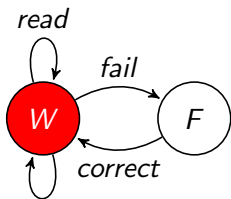


write

## Six disks

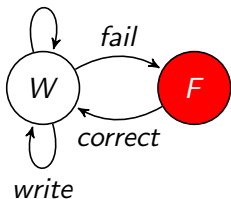


## Six disks

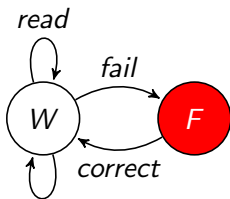


write

read

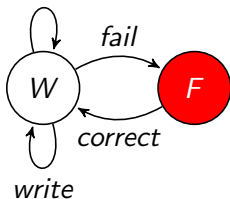


write

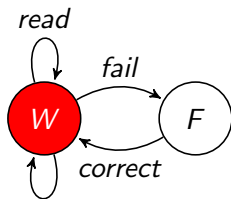


write

read

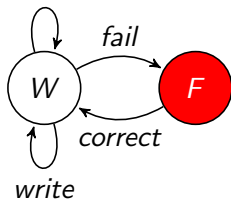


write



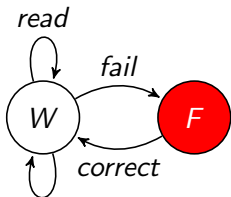
write

read



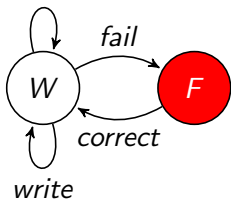
write

# Six disks

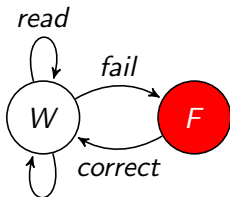


write

read

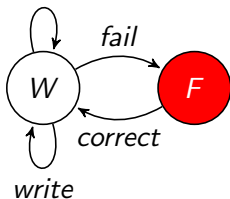


write

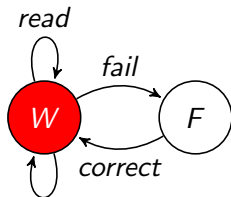


write

read

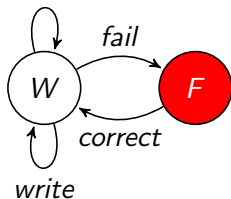


write



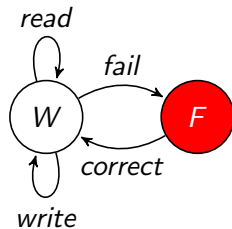
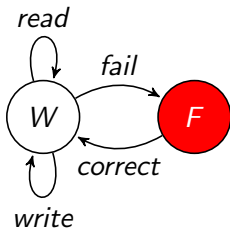
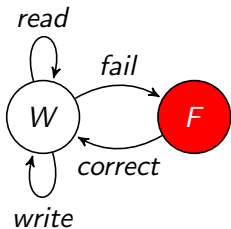
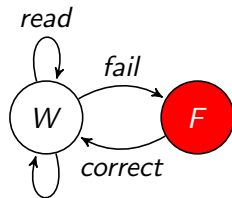
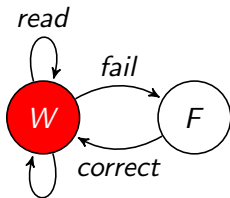
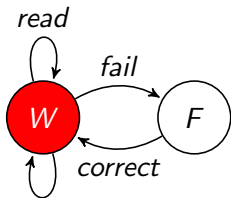
write

read

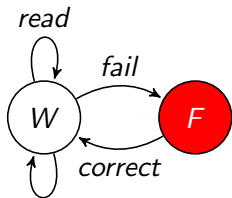


write

# Six disks

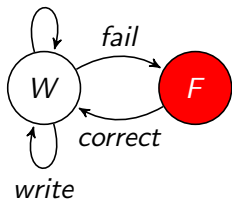


## Six disks

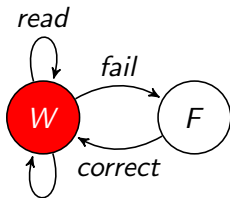


write

read

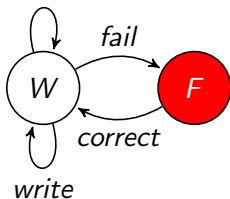


write

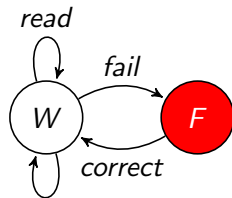


write

read

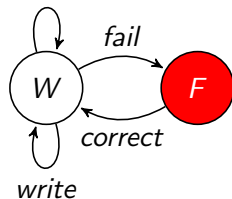


write



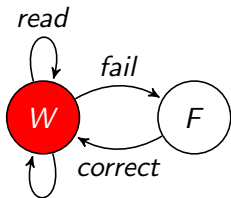
write

read



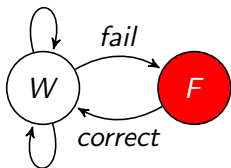
write

# Six disks

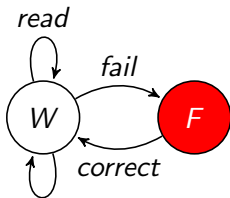


write

read

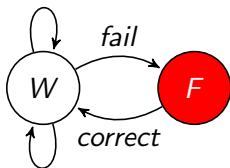


write

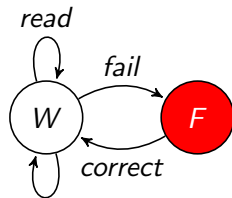


write

read

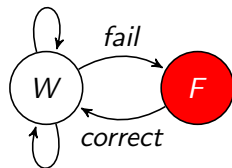


write



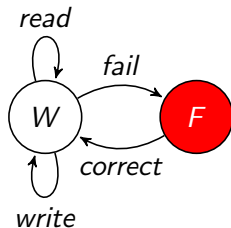
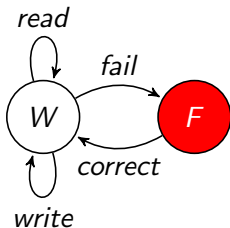
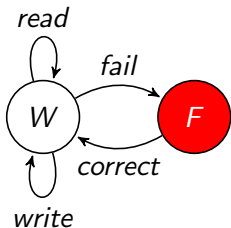
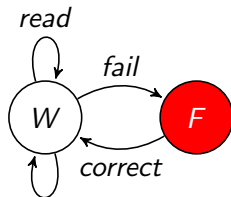
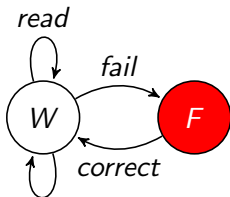
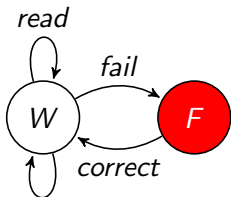
write

read

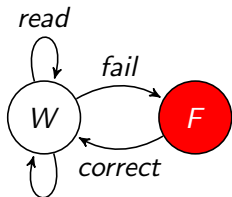


write

# Six disks

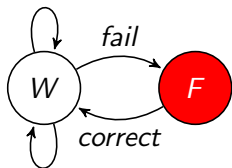


# Six disks

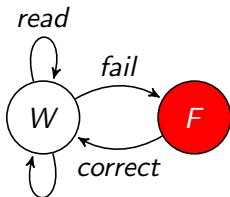


write

read

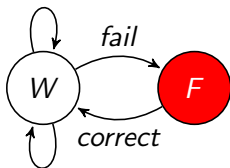


write

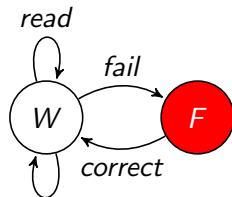


write

read

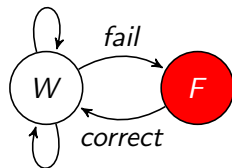


write



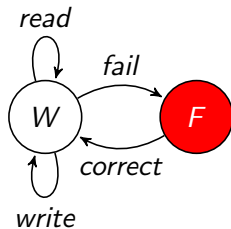
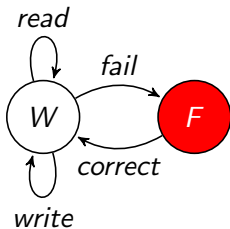
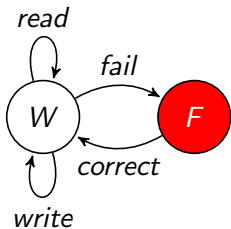
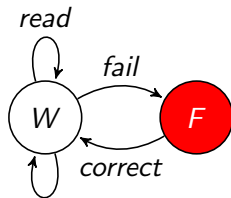
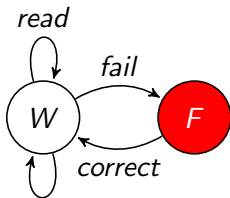
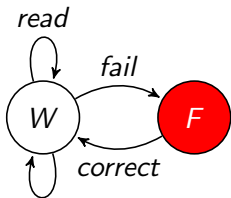
write

read



write

# Six disks



# State-space explosion

Disks	States
1	2

# State-space explosion

Disks	States
1	2
2	4

# State-space explosion

Disks	States
1	2
2	4
6	64

# State-space explosion

Disks	States
1	2
2	4
6	64
10	1024

# State-space explosion

Disks	States
1	2
2	4
6	64
10	1024
20	1048576

# State-space explosion

Disks	States
1	2
2	4
6	64
10	1024
20	1048576
50	1125899906842624

# State-space explosion

Disks	States
1	2
2	4
6	64
10	1024
20	1048576
50	1125899906842624
100	1267650600228229401496703205376

# State-space explosion

Disks	States
1	2
2	4
6	64
10	1024
20	1048576
50	1125899906842624
100	1267650600228229401496703205376
150	1427247692705959881058285969449495136382746624

# State-space explosion

Disks	States
1	2
2	4
6	64
10	1024
20	1048576
50	1125899906842624
100	1267650600228229401496703205376
150	1427247692705959881058285969449495136382746624

$2^{150}$  states

# State-space explosion

Disks	States
1	2
2	4
6	64
10	1024
20	1048576
50	1125899906842624
100	1267650600228229401496703205376
150	1427247692705959881058285969449495136382746624

$2^{150}$  states =  $2^{152}$  bytes

# State-space explosion

Disks	States
1	2
2	4
6	64
10	1024
20	1048576
50	1125899906842624
100	1267650600228229401496703205376
150	1427247692705959881058285969449495136382746624

$$2^{150} \text{ states} = 2^{152} \text{ bytes} = 2^{82} \times 2^{70} \text{ bytes}$$

# State-space explosion

Disks	States
1	2
2	4
6	64
10	1024
20	1048576
50	1125899906842624
100	1267650600228229401496703205376
150	1427247692705959881058285969449495136382746624

$$2^{150} \text{ states} = 2^{152} \text{ bytes} = 2^{82} \times 2^{70} \text{ bytes} = 2^{82} \text{ zettabytes}$$

# State-space explosion

Disks	States
1	2
2	4
6	64
10	1024
20	1048576
50	1125899906842624
100	1267650600228229401496703205376
150	1427247692705959881058285969449495136382746624

$$2^{150} \text{ states} = 2^{152} \text{ bytes} = 2^{82} \times 2^{70} \text{ bytes} = 2^{82} \text{ zettabytes}$$

Been there ...

# State-space explosion

Disks	States
1	2
2	4
6	64
10	1024
20	1048576
50	1125899906842624
100	1267650600228229401496703205376
150	1427247692705959881058285969449495136382746624

$$2^{150} \text{ states} = 2^{152} \text{ bytes} = 2^{82} \times 2^{70} \text{ bytes} = 2^{82} \text{ zettabytes}$$

Been there ... done that ...

# State-space explosion

Disks	States
1	2
2	4
6	64
10	1024
20	1048576
50	1125899906842624
100	1267650600228229401496703205376
150	1427247692705959881058285969449495136382746624

$$2^{150} \text{ states} = 2^{152} \text{ bytes} = 2^{82} \times 2^{70} \text{ bytes} = 2^{82} \text{ zettabytes}$$

Been there ... done that ... got the t-shirt!

# The t-shirt



# The t-shirt



Nice t-shirt ...

# The t-shirt



Nice t-shirt . . . wish it came in a larger size.

## Disk model in SRMC

```
DiskA::{  
  Working = (read, r).Working  
           + (write, w).Working  
           + (fail, f).Failed;  
  Failed  = (correct, c).Working;  
};
```

- We have  $W$  working disks  
and  $F$  failed ( $W + F = N$ ).

## Disk model in SRMC

```
DiskA::{  
  Working = (read, r).Working  
           + (write, w).Working  
           + (fail, f).Failed;  
  Failed  = (correct, c).Working;  
};
```

- We have  $W$  working disks and  $F$  failed ( $W + F = N$ ).
- Working disks fail at rate  $f \times W$ .

## Disk model in SRMC

```
DiskA::{  
  Working = (read, r).Working  
           + (write, w).Working  
           + (fail, f).Failed;  
  Failed  = (correct, c).Working;  
};
```

- We have  $W$  working disks and  $F$  failed ( $W + F = N$ ).
- Working disks fail at rate  $f \times W$ .
- Failures are corrected at rate  $c \times F$ .

## Disk model in SRMC

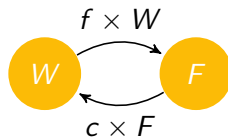
```
DiskA::{  
  Working = (read, r).Working  
           + (write, w).Working  
           + (fail, f).Failed;  
  Failed  = (correct, c).Working;  
};
```

- We have  $W$  working disks and  $F$  failed ( $W + F = N$ ).
- Working disks fail at rate  $f \times W$ .
- Failures are corrected at rate  $c \times F$ .

## Disk model in SRMC

```
DiskA::{\n  Working = (read, r).Working\n          + (write, w).Working\n          + (fail, f).Failed;\n  Failed  = (correct, c).Working;\n};
```

- We have  $W$  working disks and  $F$  failed ( $W + F = N$ ).
- Working disks fail at rate  $f \times W$ .
- Failures are corrected at rate  $c \times F$ .



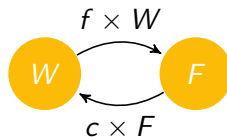
# Disk model in SRMC

```

DiskA::{
  Working = (read, r).Working
          + (write, w).Working
          + (fail, f).Failed;
  Failed  = (correct, c).Working;
};

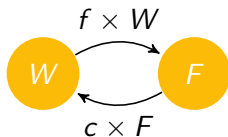
```

- We have  $W$  working disks and  $F$  failed ( $W + F = N$ ).
- Working disks fail at rate  $f \times W$ .
- Failures are corrected at rate  $c \times F$ .



$$\begin{aligned}
 dW/dt &= -f \times W + c \times F \\
 dF/dt &= f \times W - c \times F
 \end{aligned}$$

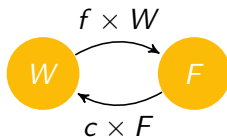
# Fluid-flow disk model



$$dW/dt = -f \times W + c \times F$$

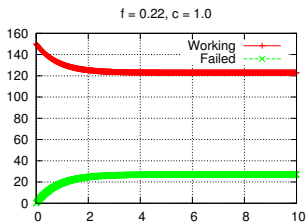
$$dF/dt = f \times W - c \times F$$

# Fluid-flow disk model

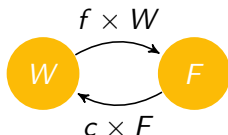


$$dW/dt = -f \times W + c \times F$$

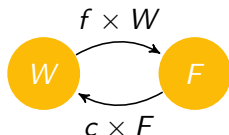
$$dF/dt = f \times W - c \times F$$



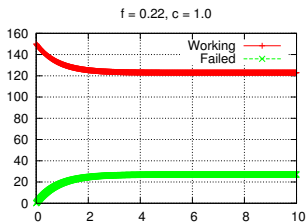
# Fluid-flow disk model



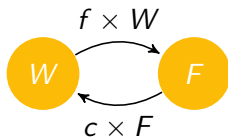
$$\begin{aligned} dW/dt &= -f \times W + c \times F \\ dF/dt &= f \times W - c \times F \end{aligned}$$



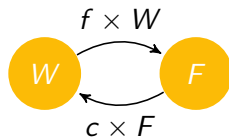
$$\begin{aligned} 0 &= -f \times W + c \times F \\ 0 &= f \times W - c \times F \end{aligned}$$



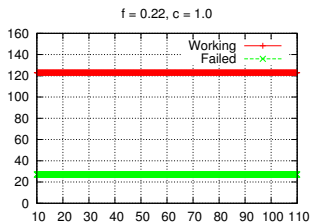
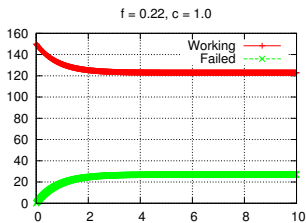
# Fluid-flow disk model



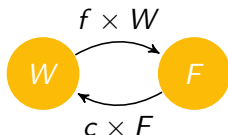
$$\begin{aligned} dW/dt &= -f \times W + c \times F \\ dF/dt &= f \times W - c \times F \end{aligned}$$



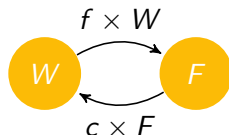
$$\begin{aligned} 0 &= -f \times W + c \times F \\ 0 &= f \times W - c \times F \end{aligned}$$



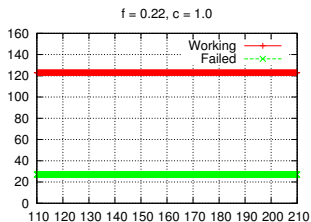
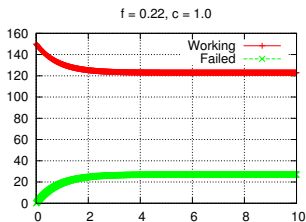
# Fluid-flow disk model



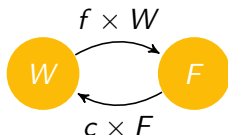
$$\begin{aligned} dW/dt &= -f \times W + c \times F \\ dF/dt &= f \times W - c \times F \end{aligned}$$



$$\begin{aligned} 0 &= -f \times W + c \times F \\ 0 &= f \times W - c \times F \end{aligned}$$

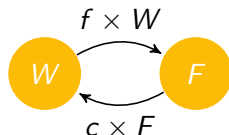


# Fluid-flow disk model



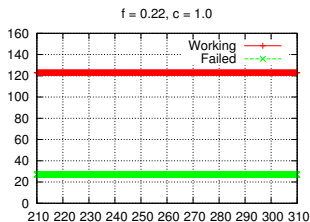
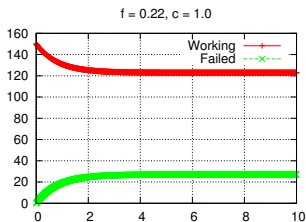
$$\frac{dW}{dt} = -f \times W + c \times F$$

$$\frac{dF}{dt} = f \times W - c \times F$$

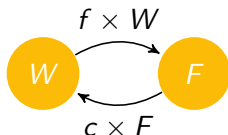


$$0 = -f \times W + c \times F$$

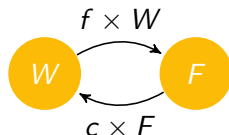
$$0 = f \times W - c \times F$$



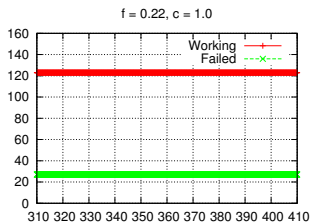
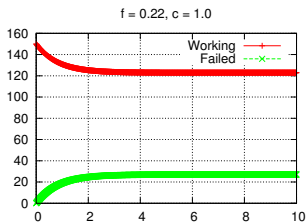
# Fluid-flow disk model



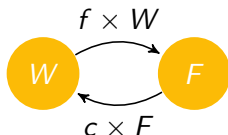
$$\begin{aligned} dW/dt &= -f \times W + c \times F \\ dF/dt &= f \times W - c \times F \end{aligned}$$



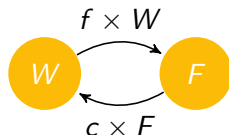
$$\begin{aligned} 0 &= -f \times W + c \times F \\ 0 &= f \times W - c \times F \end{aligned}$$



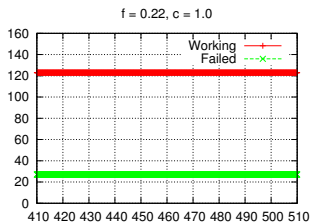
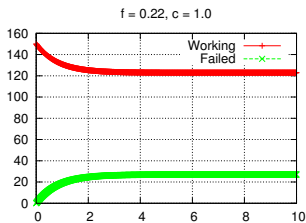
# Fluid-flow disk model



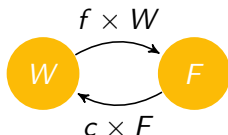
$$\begin{aligned} dW/dt &= -f \times W + c \times F \\ dF/dt &= f \times W - c \times F \end{aligned}$$



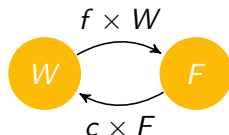
$$\begin{aligned} 0 &= -f \times W + c \times F \\ 0 &= f \times W - c \times F \end{aligned}$$



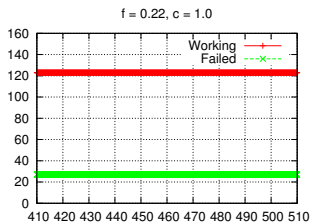
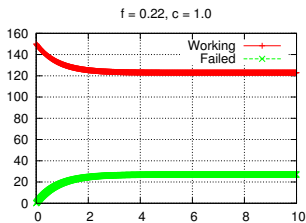
# Fluid-flow disk model



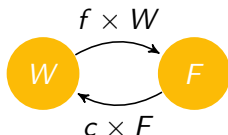
$$\begin{aligned} dW/dt &= -f \times W + c \times F \\ dF/dt &= f \times W - c \times F \end{aligned}$$



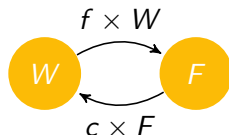
$$\left. \begin{aligned} W &= c/f \times F \\ F &= f/c \times W \end{aligned} \right\} W + F = N$$



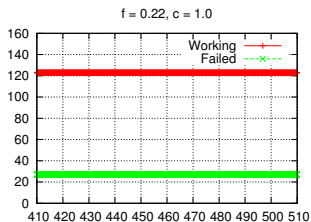
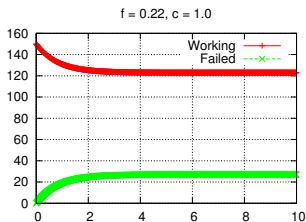
# Fluid-flow disk model



$$\begin{aligned} dW/dt &= -f \times W + c \times F \\ dF/dt &= f \times W - c \times F \end{aligned}$$

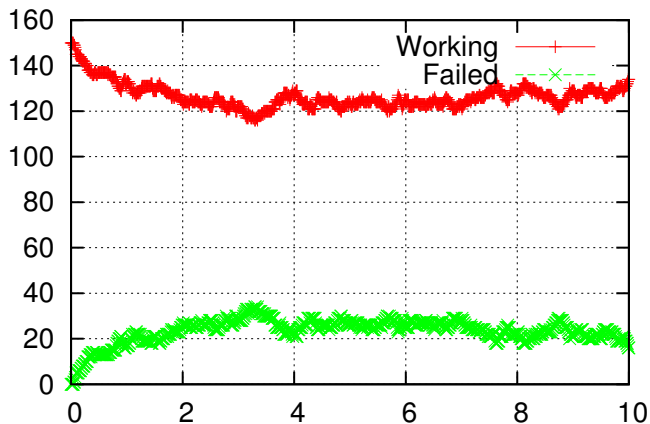


$$\begin{aligned} W &= N/(f/c + 1) \\ F &= N/(c/f + 1) \end{aligned}$$



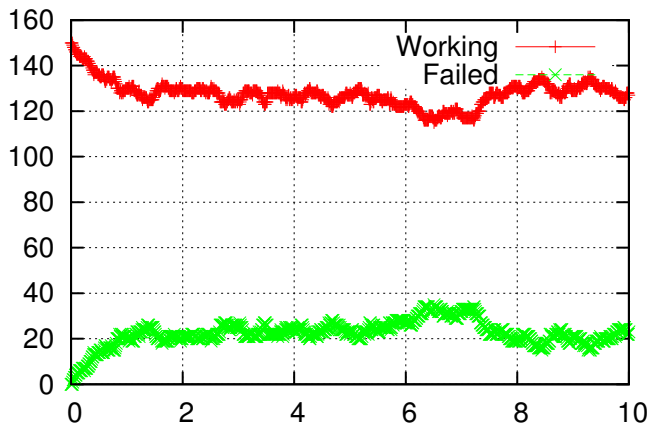
# Markov chain simulation of the disk model

Simulation run A :  $f = 0.22$ ,  $c = 1.0$



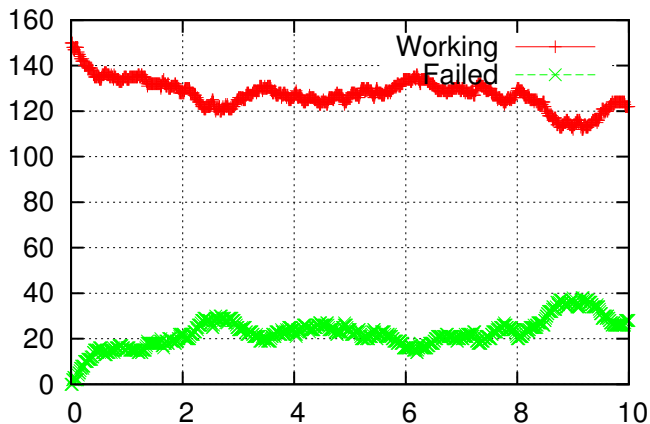
# Markov chain simulation of the disk model

Simulation run B :  $f = 0.22$ ,  $c = 1.0$



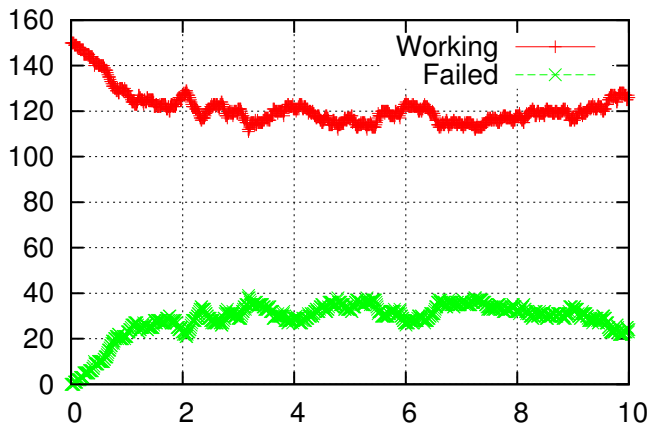
# Markov chain simulation of the disk model

Simulation run C :  $f = 0.22$ ,  $c = 1.0$



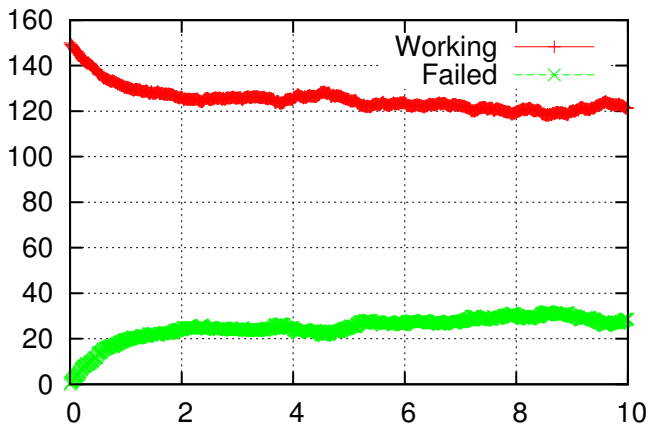
# Markov chain simulation of the disk model

Simulation run D :  $f = 0.22$ ,  $c = 1.0$



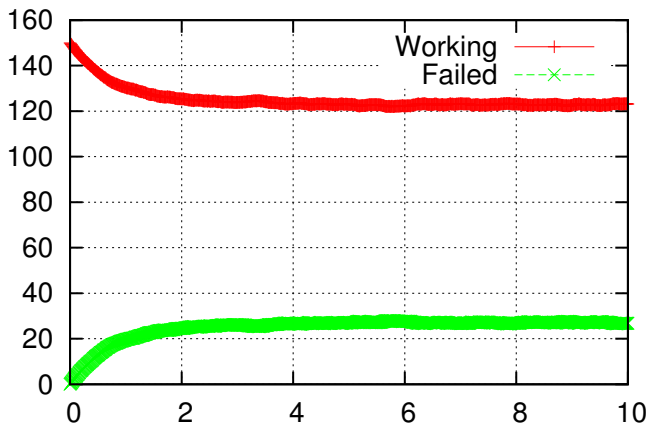
# Markov chain simulation of the disk model

Ave. of 10 simulation runs :  $f = 0.22$ ,  $c = 1.0$



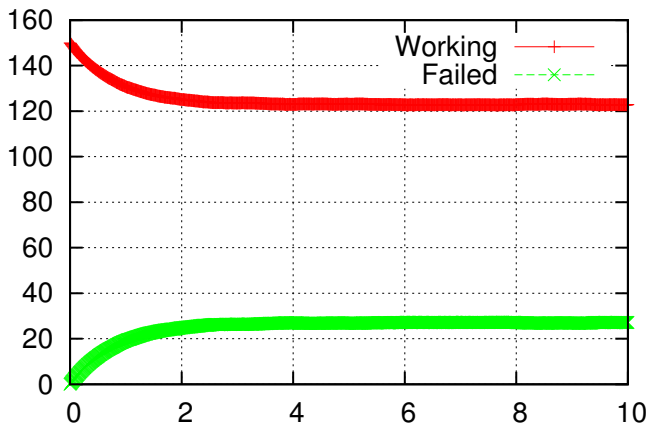
# Markov chain simulation of the disk model

Ave. of 100 simulation runs :  $f = 0.22$ ,  $c = 1.0$



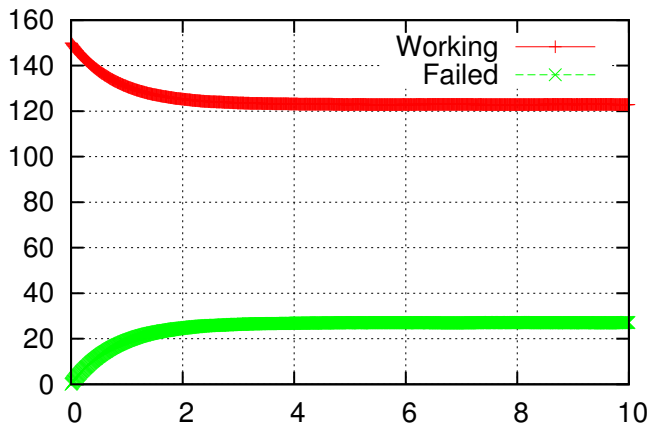
# Markov chain simulation of the disk model

Ave. of 1,000 simulation runs :  $f = 0.22$ ,  $c = 1.0$

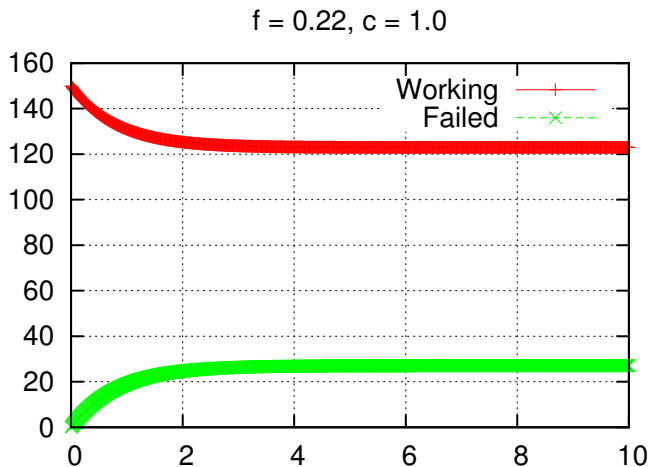


# Markov chain simulation of the disk model

Ave. of 10,000 simulation runs :  $f = 0.22$ ,  $c = 1.0$



# ODE analysis of the disk model



# The new t-shirt



# The new t-shirt



Nice t-shirt ...

# The new t-shirt



Nice t-shirt . . . available in XXXL!

# Breakthrough paper



Jane Hillston.

Fluid flow approximation of PEPA models.

Proceedings of the Second International Conference on the Quantitative Evaluation of Systems, Torino, Italy, 2005.

# Outline

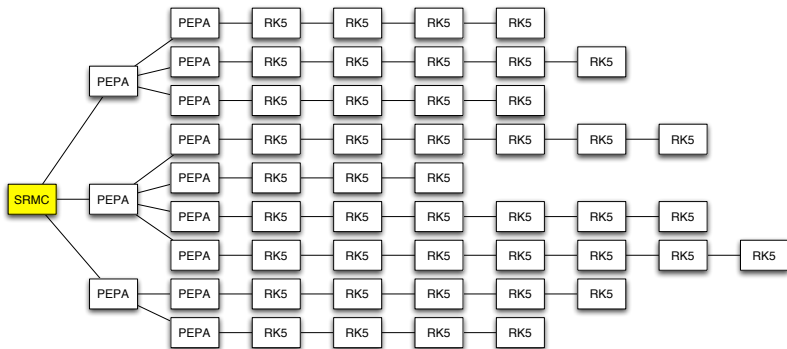
- 1 Quantitative Analysis
- 2 Scalability and Large-Scale Systems**
- 3 Case Study

# Large-scale systems

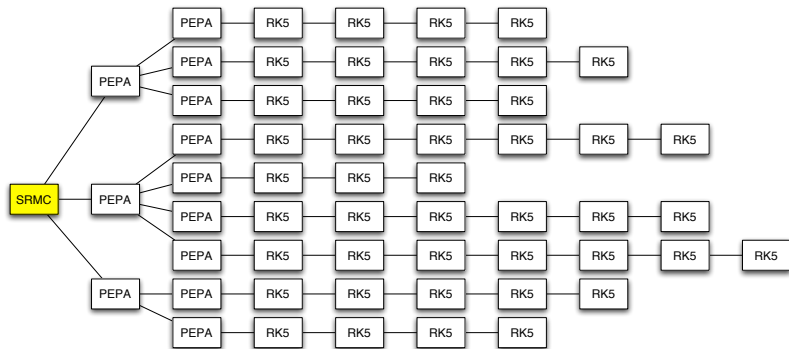
- Size is not the only difficulty in modelling large-scale systems.
- We need to accept that large-scale systems are naturally heterogeneous.
- We need to represent uncertainty about components, performance and configuration.
- We would like our analysis to be robust in the face of likely changes to the system.



# The evaluation model

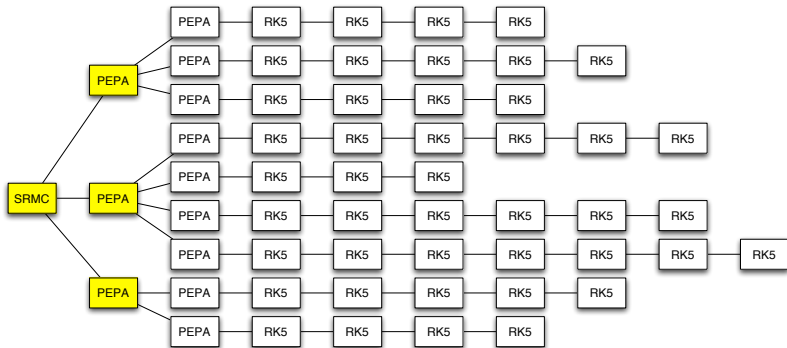


# The evaluation model

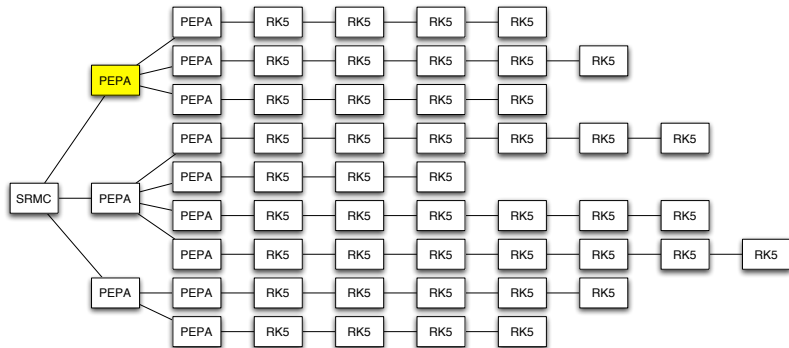


$\text{Disk} ::= \{ \text{DiskA}, \text{DiskB}, \text{DiskC} \}$

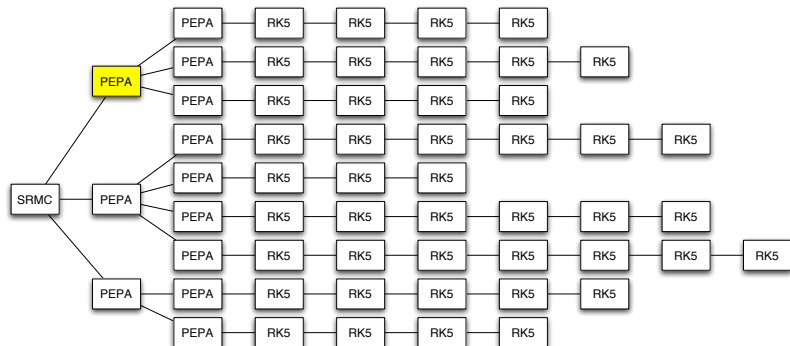
# The evaluation model



# The evaluation model

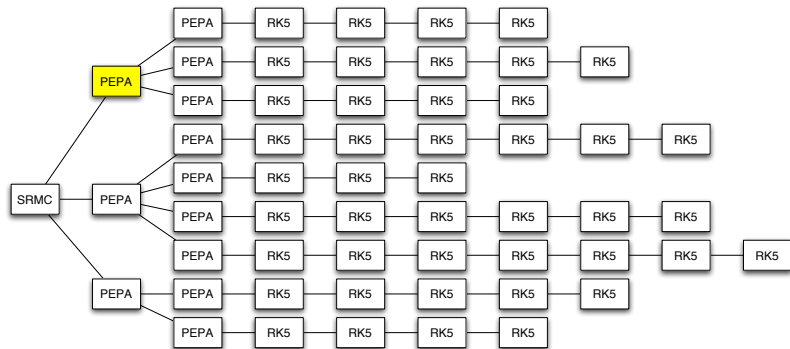


# The evaluation model



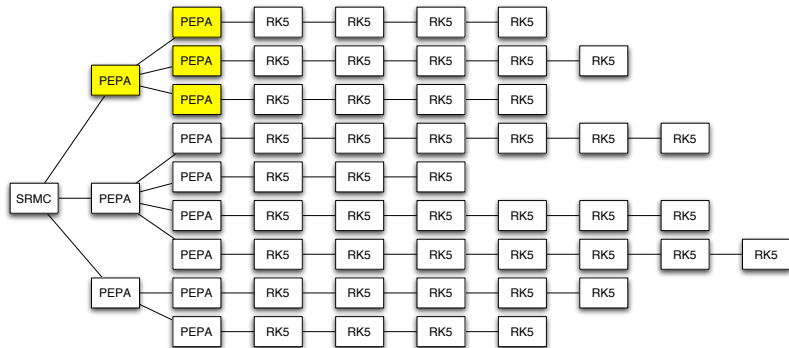
`HTTPserver[?n+1] ==> HTTPserver[?n]`

# The evaluation model



`FTPserver[?n] ==> FTPserver[?n+1]`

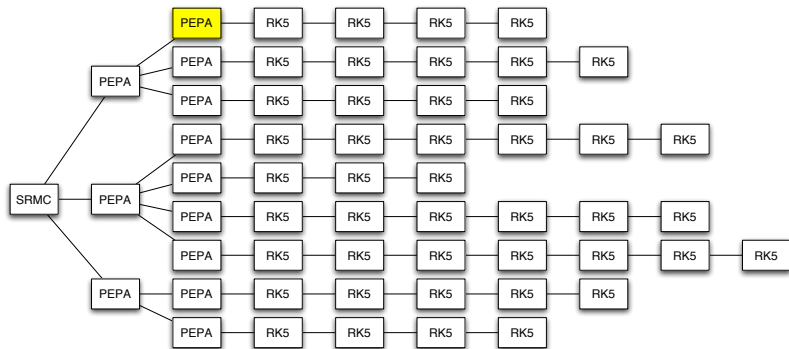
# The evaluation model



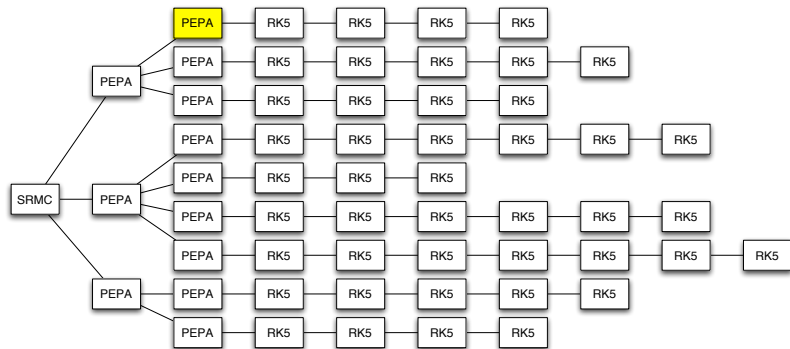




# The evaluation model

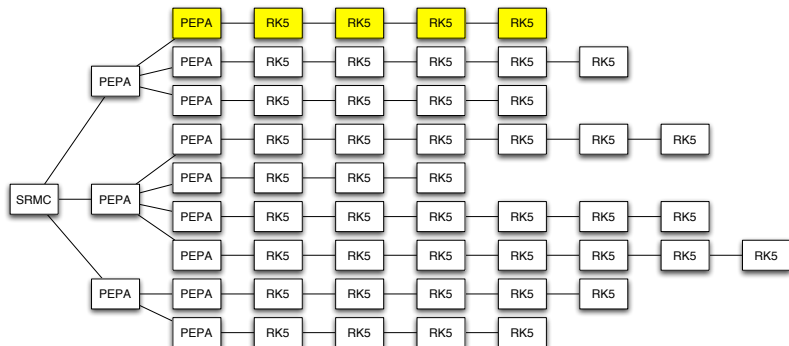


# The evaluation model



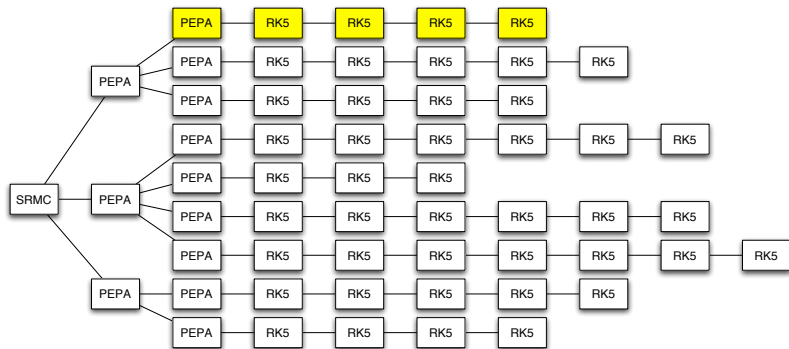
$$f = \{ 0.075, 0.1, 0.22, 0.325 \}$$

# The evaluation model



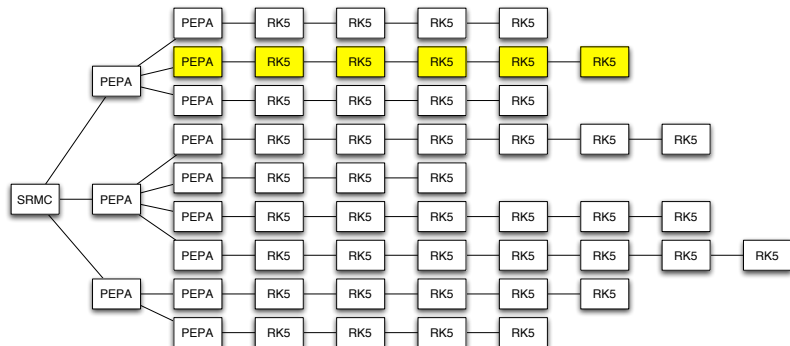
$$f = \{ 0.075, 0.1, 0.22, 0.325 \}$$

# The evaluation model



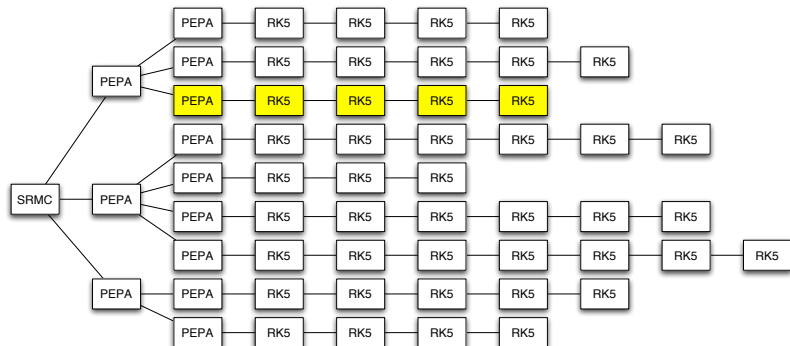
Execute fifth-order Runge-Kutta integrator (RK5)

# The evaluation model



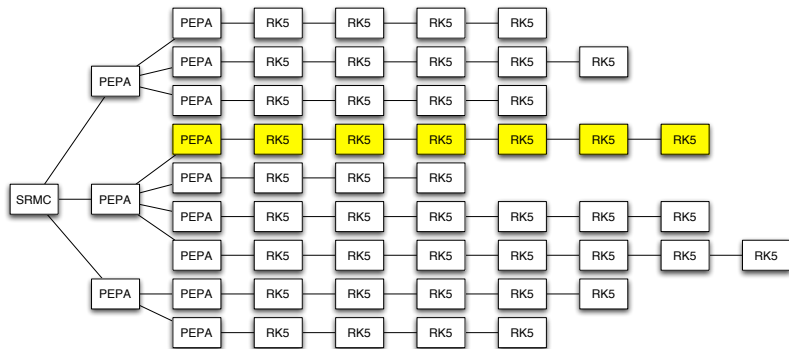
Execute fifth-order Runge-Kutta integrator (RK5)

# The evaluation model



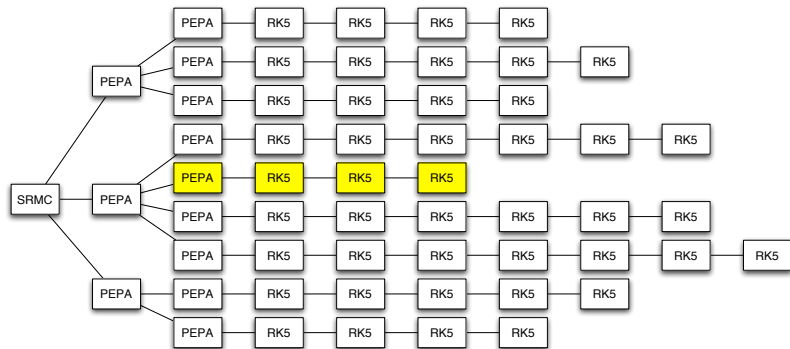
Execute fifth-order Runge-Kutta integrator (RK5)

# The evaluation model



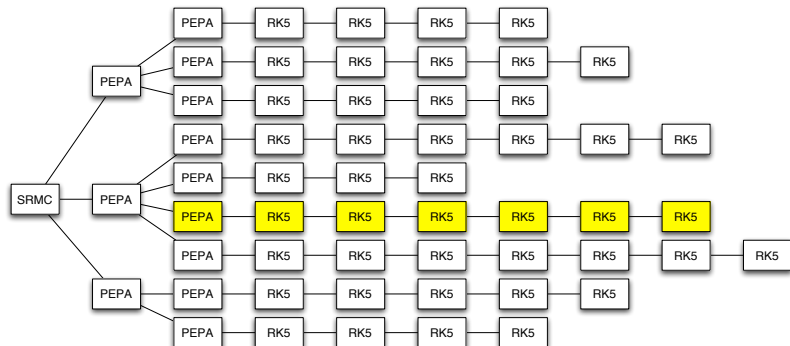
Execute fifth-order Runge-Kutta integrator (RK5)

# The evaluation model



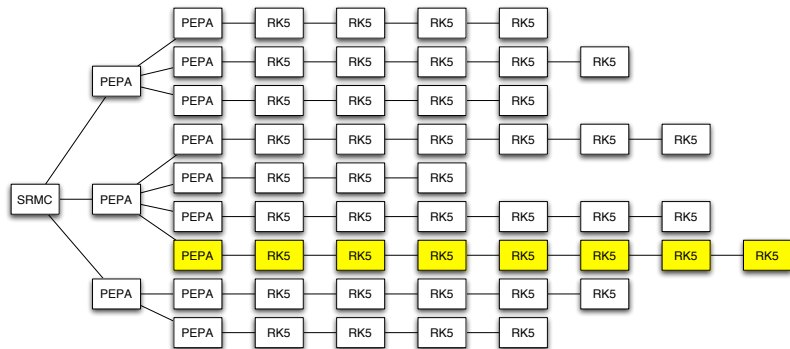
Execute fifth-order Runge-Kutta integrator (RK5)

# The evaluation model



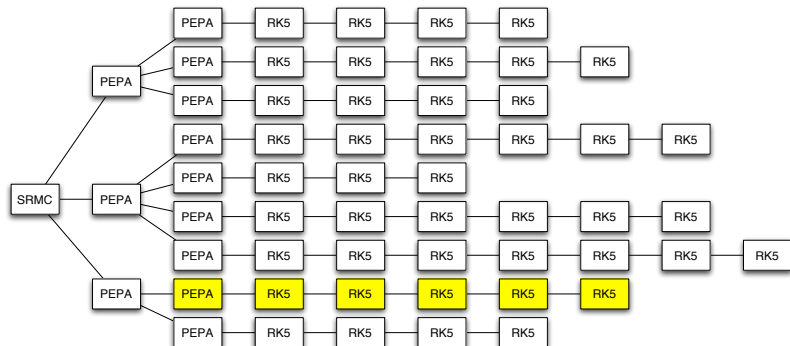
Execute fifth-order Runge-Kutta integrator (RK5)

# The evaluation model



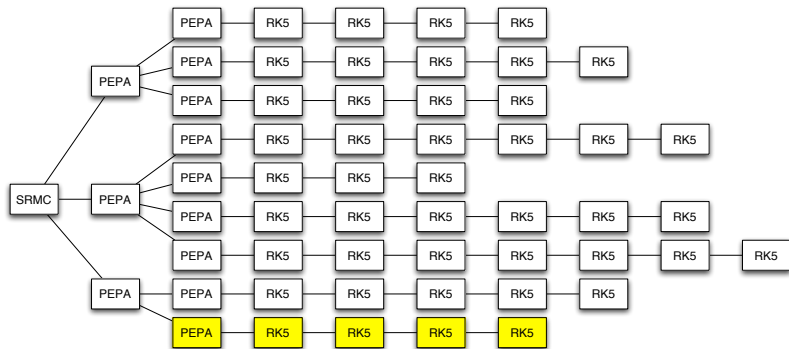
Execute fifth-order Runge-Kutta integrator (RK5)

# The evaluation model



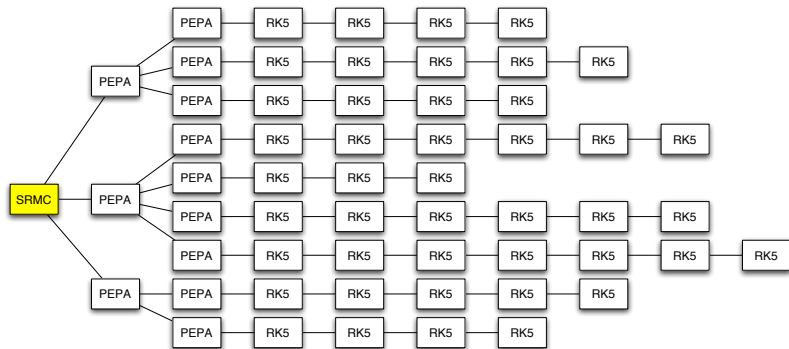
Execute fifth-order Runge-Kutta integrator (RK5)

# The evaluation model

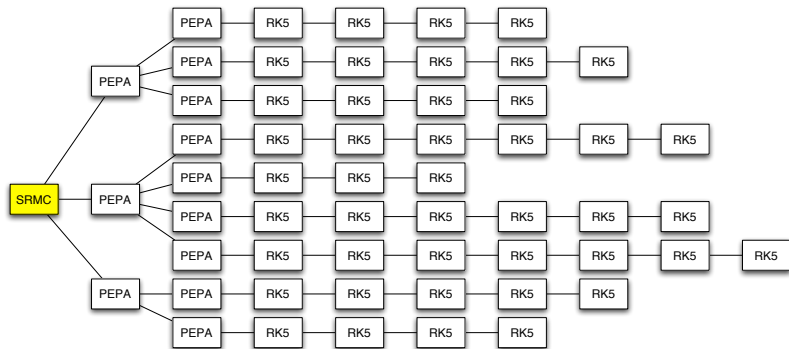


Execute fifth-order Runge-Kutta integrator (RK5)

# The evaluation model

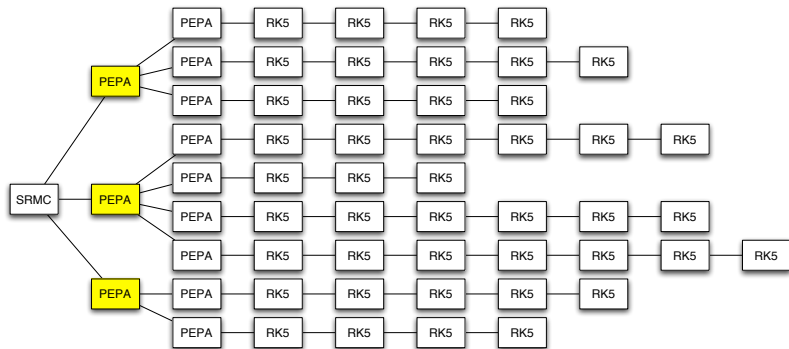


# The evaluation model



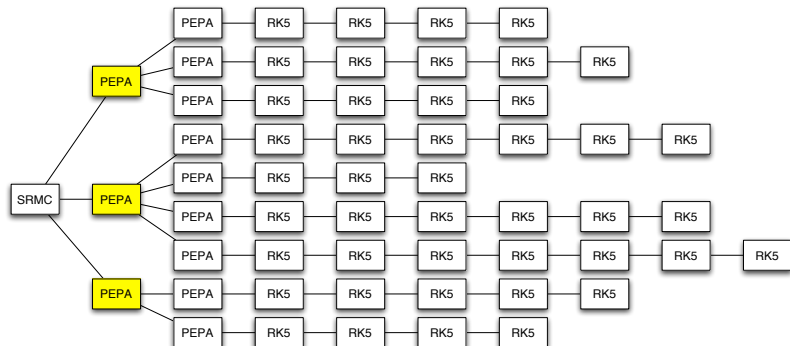
Resolve bindings

# The evaluation model



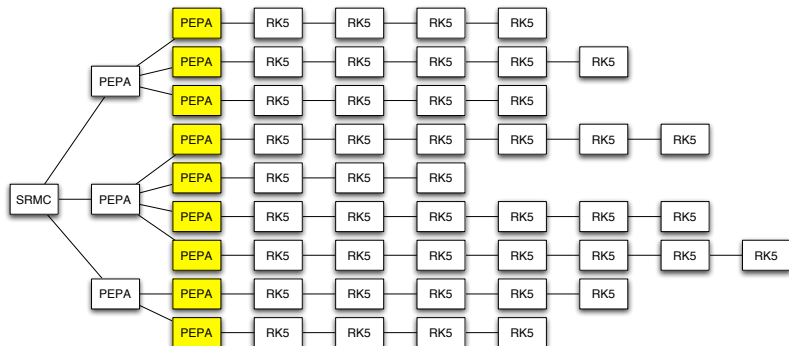
Resolve bindings

# The evaluation model



Apply model transformations

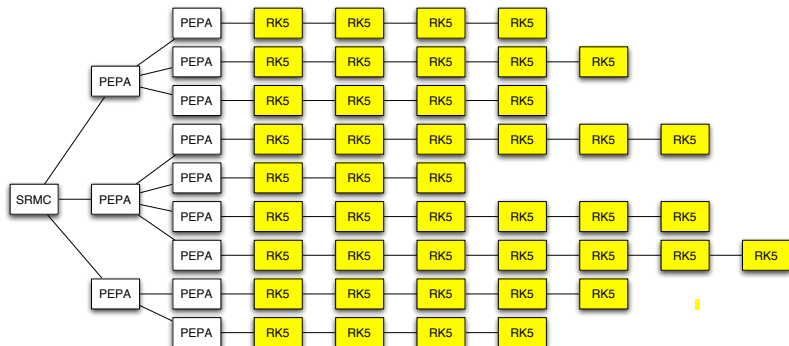
# The evaluation model



Apply model transformations

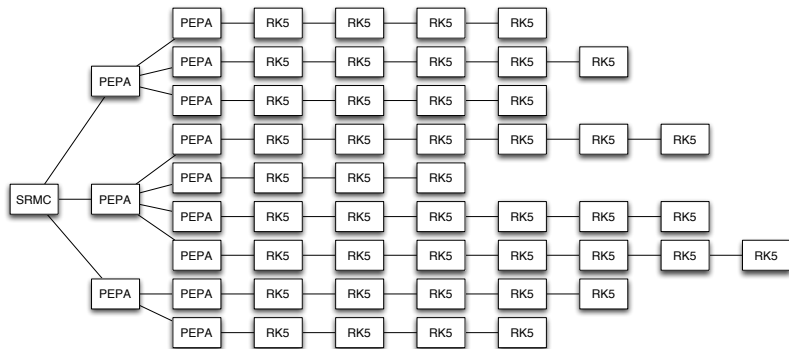


# The evaluation model



Parameter sweep

# The evaluation model



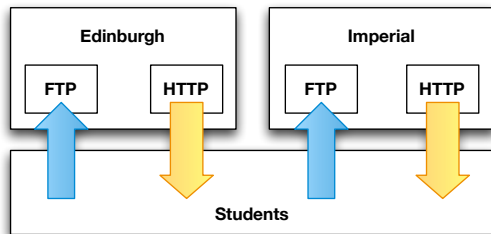
# Outline

- 1 Quantitative Analysis
- 2 Scalability and Large-Scale Systems
- 3 Case Study**

# Virtual university



# Virtual university



- Two sites: Edinburgh and Imperial
- Students upload coursework to FTP servers
- Students download course content from HTTP servers
- We consider the system under different workloads

## Edinburgh site

```
Edinburgh::{  
  mu = 0.0001; gamma = 0.125; // rates of fail and repair  
  avail = {0.6,0.7,0.8,0.9,1.0}; // server availability  
  phi = 10.0; psi = 7.0; // rates for download and upload  
  
  // The HTTP server  
  Http::{  
    Idle = (download, avail * phi).Idle + (fail, mu).Broken;  
    Broken = (repair, gamma).Idle;  
  };  
  // The FTP server  
  Ftp::{  
    Idle = (upload, avail * psi).Idle + (fail, mu).Broken;  
    Broken = (repair, gamma).Idle;  
  };  
};
```

# Imperial site

```
Imperial::{  
  mu = 0.006; gamma = 0.125; // failures are more likely  
  avail = {0.6,0.7,0.8,0.9,1.0}; // availability is the same  
  phi = 20.0; psi = 15.0; // download and upload are faster  
  
  // The HTTP server  
  Http::{  
    Idle = (download, avail * phi).Idle + (fail, mu).Broken;  
    Broken = (repair, gamma).Idle;  
  };  
  // The FTP server  
  Ftp::{  
    Idle = (upload, avail * psi).Idle + (fail, mu).Broken;  
    Broken = (repair, gamma).Idle;  
  };  
};
```

# Harry, a client

```
Harry::{  
  c = { 0.01, 0.02, 0.03 }; d = 1.0;  
   $\delta$  = { 0.01, 0.02, 0.03 };  $\mu$  = 1.0;  
  
  // Model components  
  Idle = (connect, c/2).Upload + (connect, c/2).Download;  
  Upload = (upload,  $\mu$ ).Disconnect;  
  Download = (download,  $\delta$ ).Disconnect;  
  Disconnect = (disconnect, d).Idle;  
  
  // Virtual components  
  Uploading = [ Upload ];  
  Downloading = [ Download ];  
  Inservice = [ Uploading + Downloading ];  
};
```

## Sally, another client

```
Sally::{  
  c = { 0.009, 0.0095, 0.01 };    d = 0.5;  
   $\delta$  = { 0.01, 0.02, 0.03 };  $\mu$  = 0.2;  
  
  Idle = (connect, c/3).Upload  
        + (connect, c/3).Download1  
        + (connect, c/3).Download2 ;  
  Upload  = (upload,  $\mu$ ).Disconnect;  
  Download1 = (download,  $\delta$ ).Download2;  
  Download2 = (download,  $\delta$ ).Disconnect;  
  Disconnect = (disconnect, d).Idle;  
  
  Uploading  = [ Upload ];  
  Downloading = [ Download1 + Download2 ];  
  Inservice  = [ Uploading + Downloading ];  
};
```

# Model composition

## Configuration

```
Client ::= { Harry, Sally };  
Http   ::= { Edinburgh::Http, Imperial::Http };  
Ftp    ::= { Edinburgh::Ftp, Imperial::Ftp };
```

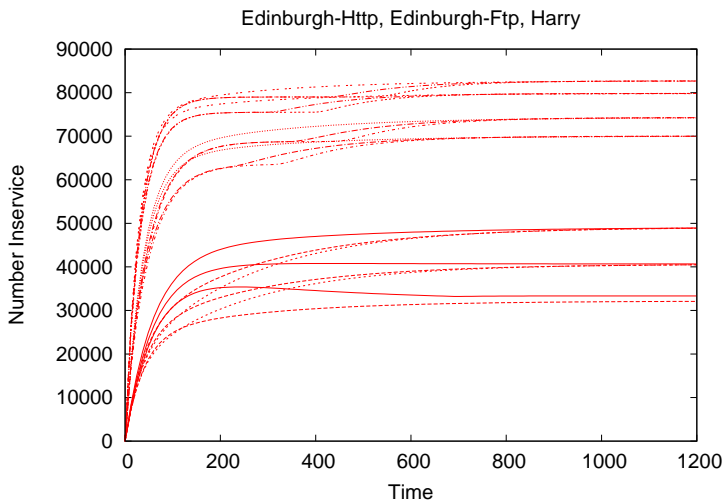
## Parameters

```
servers = { 3, 4, 5, 6 }; threads = 20; clients = 100000;
```

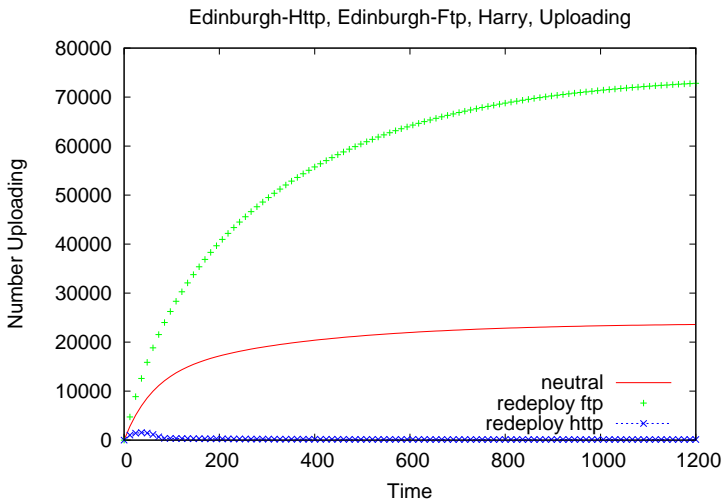
## Model

```
Client::Idle[clients]  
    <download, upload> (Http::Idle[servers * threads]  
                        ||Ftp::Idle[servers * threads])
```

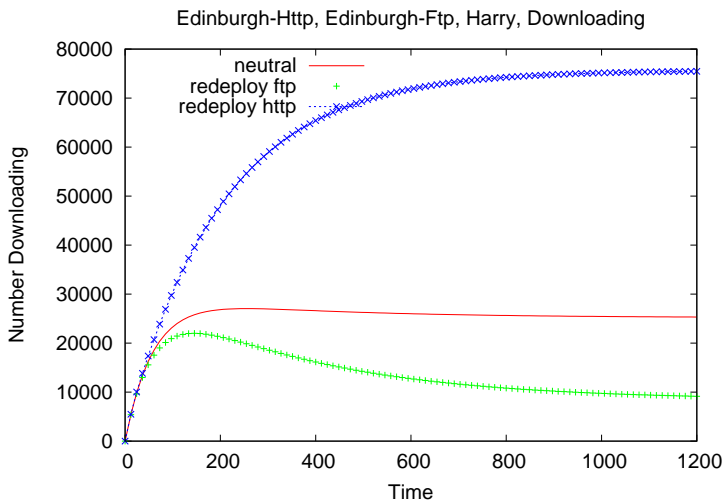
# First configuration



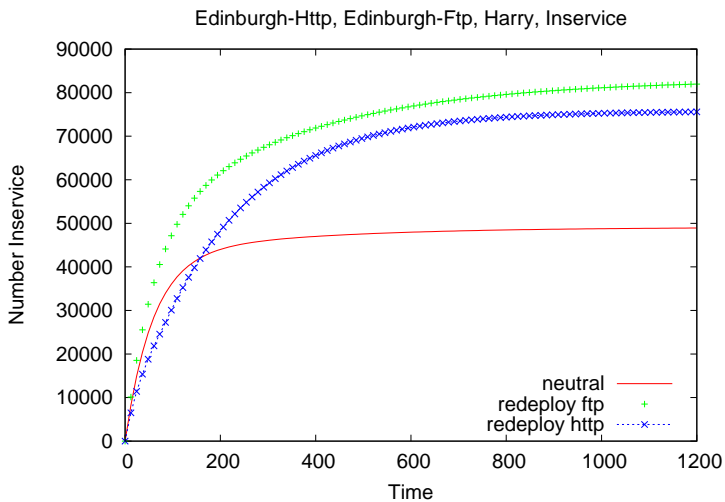
# First configuration, uploading



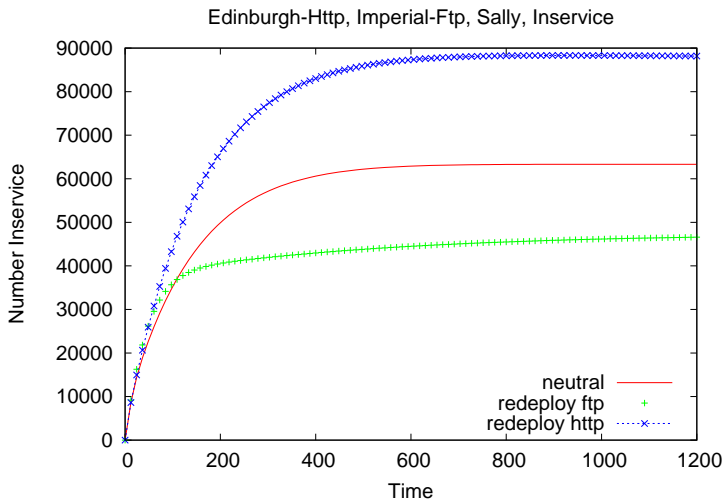
# First configuration, downloading



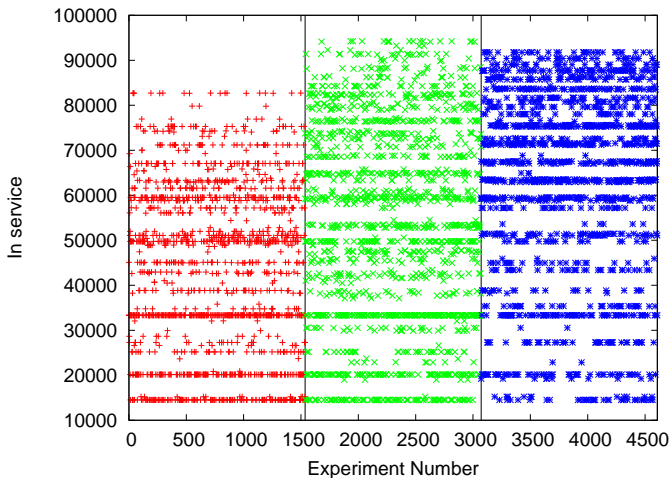
# First configuration, number in service



## Second configuration, number in service

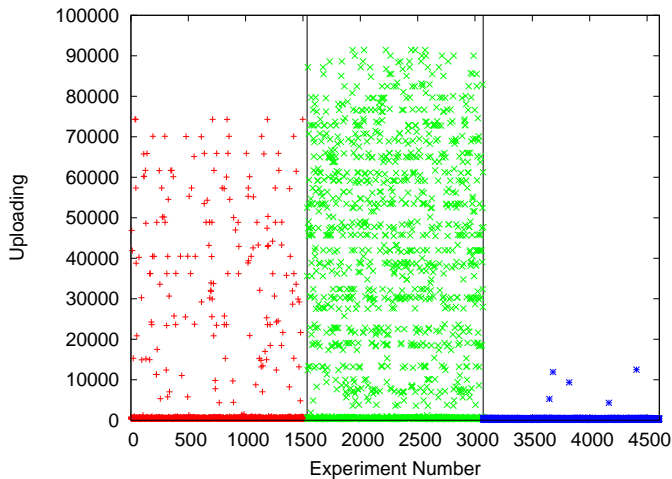


# Effect of redeployments on number in service



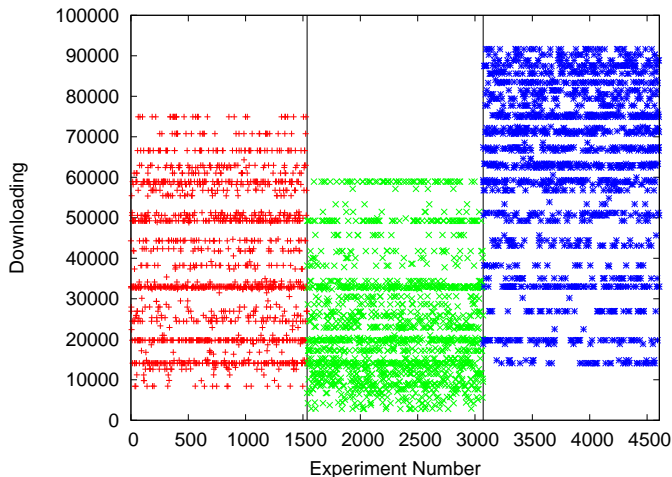
Red: no change, Green: redeploy FTP, Blue: redeploy HTTP

# Effect of redeployments on uploading



Red: no change, Green: redeploy FTP, Blue: redeploy HTTP

# Effect of redeployments on downloading



Red: no change, Green: redeploy FTP, Blue: redeploy HTTP

# Conclusions

- Scalable systems are difficult to develop and difficult to analyse.
- Component-based modelling supported by divide-and-conquer analysis seems a promising approach.
- We analyse configurations by cases and perform parameter sweep across uncertain parameters.
- Hopefully a “real world” approach to a “real world” problem.

Website

<http://groups.inf.ed.ac.uk/srmc/>

# Acknowledgements

## Co-authors

Allan Clark and Mirco Tribastone.

## PEPA group and collaborators at Edinburgh

Federica Ciocchetta, Jie Ding, Adam Duguid, Vashti Galpin, Maria Luisa Guerriero, Jane Hillston, Michael Smith, Hao Wang.

## PEPA users and collaborators elsewhere

Jeremy Bradley, Muffy Calder, Nick Dingle, Peter Harrison, Richard Hayden, William Knottenbelt, Nigel Thomas, Yishi Zhao.

## Sensoria project

José Fiadeiro, Howard Foster, Reiko Heckel, Mieke Massink, Laura Semini, Dániel Varró, Martin Wirsing.