

# Availability in Globally Distributed Storage Systems

Robert Kozikowski

# Introduction

Designing and optimizing the distributed systems for goals such as data availability relies on models of system behavior. This includes quantifying the impact of failures. Models are derived from studying a year of live operation at Google.

This presentation is based on a paper "Availability in Globally Distributed Storage Systems" written by google engineers.

# Presentation plan

- Background
- Component Availability
  - Compare mean time to failure
  - Classify the failure causes for storage nodes
  - Apply a clustering heuristic
  - Quantify how likely a failure burst is associated with a given failure domain
- Data Availability
  - Demonstrate the importance of modeling correlated failures when predicting availability
  - Formulate a Markov model for data availability
  - Introduce multi-cell replication schemes
  - Show the impact of hardware failure is smaller than tuning recovery

# Component Availability

# Background

Studies are performed on cloud computing storage environment. These environments use loosely coupled distributed systems such as GFS.

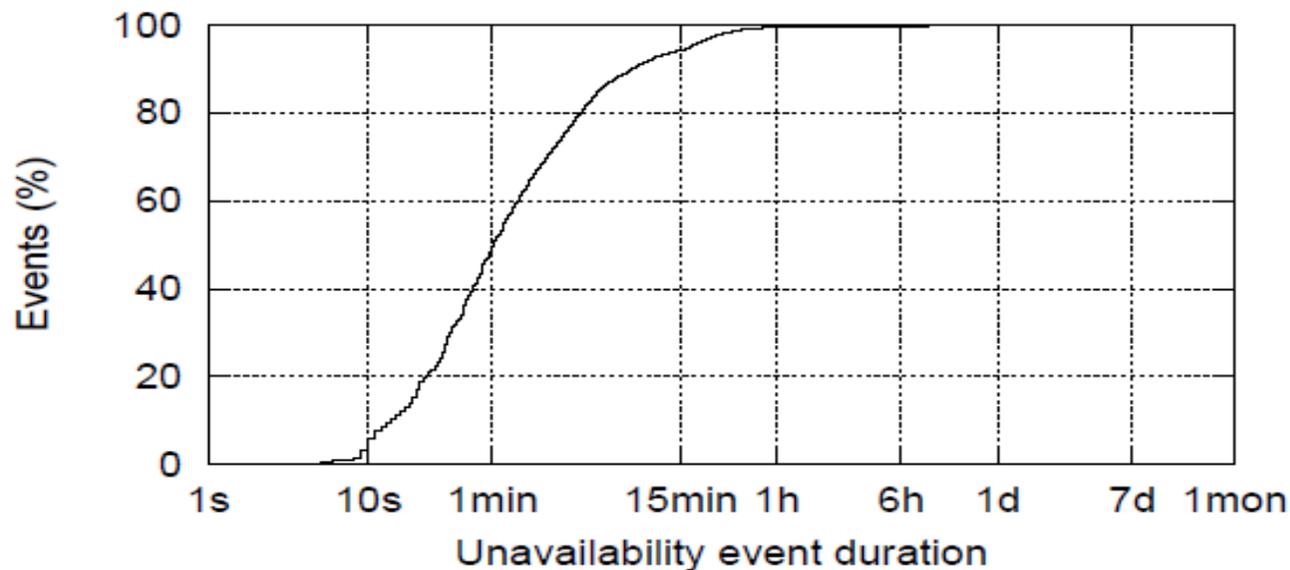
A single storage server is called a node.

A group of 40-80 nodes physically placed together are called a rack.

A large collection of nodes, usually from 1000 to 7000 nodes along with their coordination processes are called a cell.

# Availability

Node is considered unavailable if it fails to respond to health checking pings. Later in presentation, only failures shorter than 15 minutes are considered.



# Measures

Through the presentation there will be used primarily two metrics.

By  $A_N$  we will understand average percentage up-time for a node in a cell.

By MTTF we will understand Mean time to failure. It's uptime divided by number of failures.

# Data Replication

There are two common schemes to data replications.

Data is divided into a set of stripes, each of them is a set of fixed size data blocks called chunks.

By  $R=n$  we mean that chunks is replicated  $n$  times in a stripe.

By  $RS(n,m)$ , Reed-Solomon erasure encoding, we mean that stripe have size  $(n+m)$  and it can be restored from any  $n$  chunks.

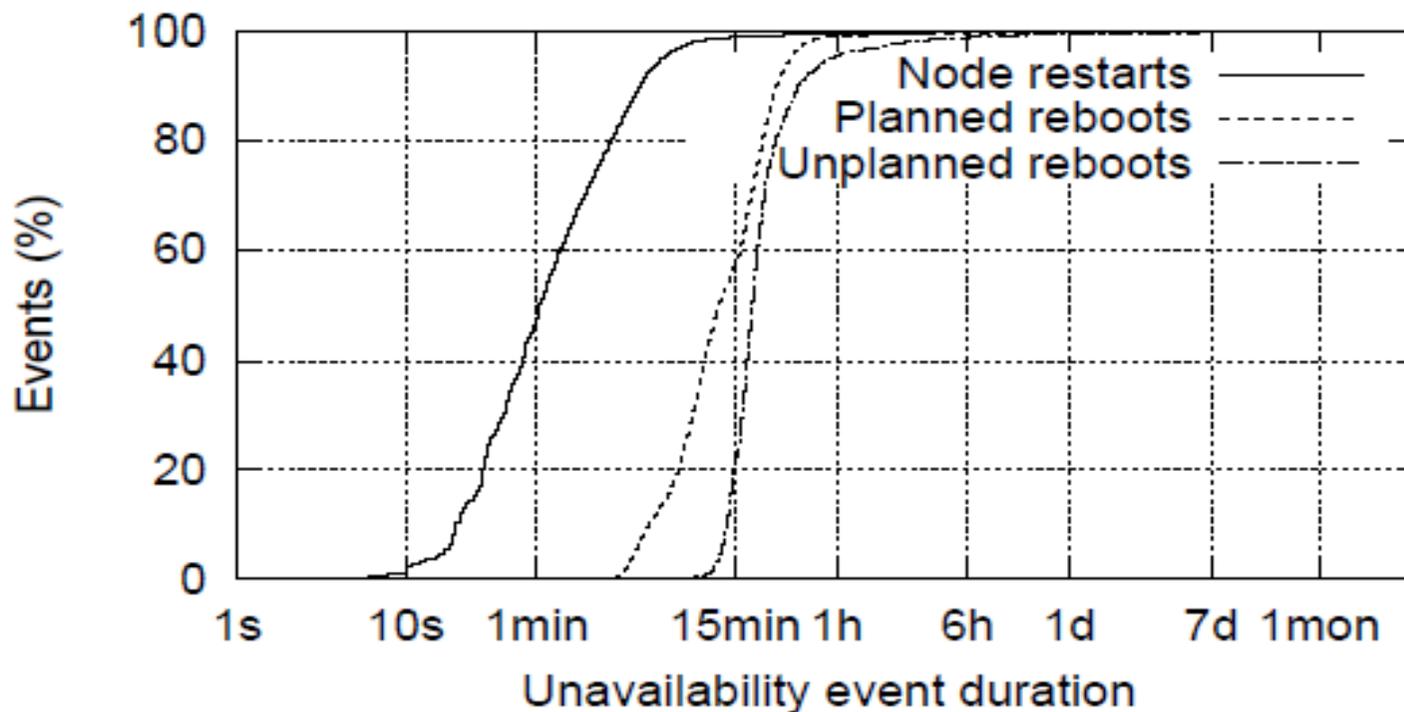
# Types of failures

In this presentation we look at errors from the perspective of application layer. We divide errors in four groups:

1. Node restarts - software restarts of the storage program running on each machine
2. Planned machine reboots
3. Unplanned machine reboots
4. Unknown

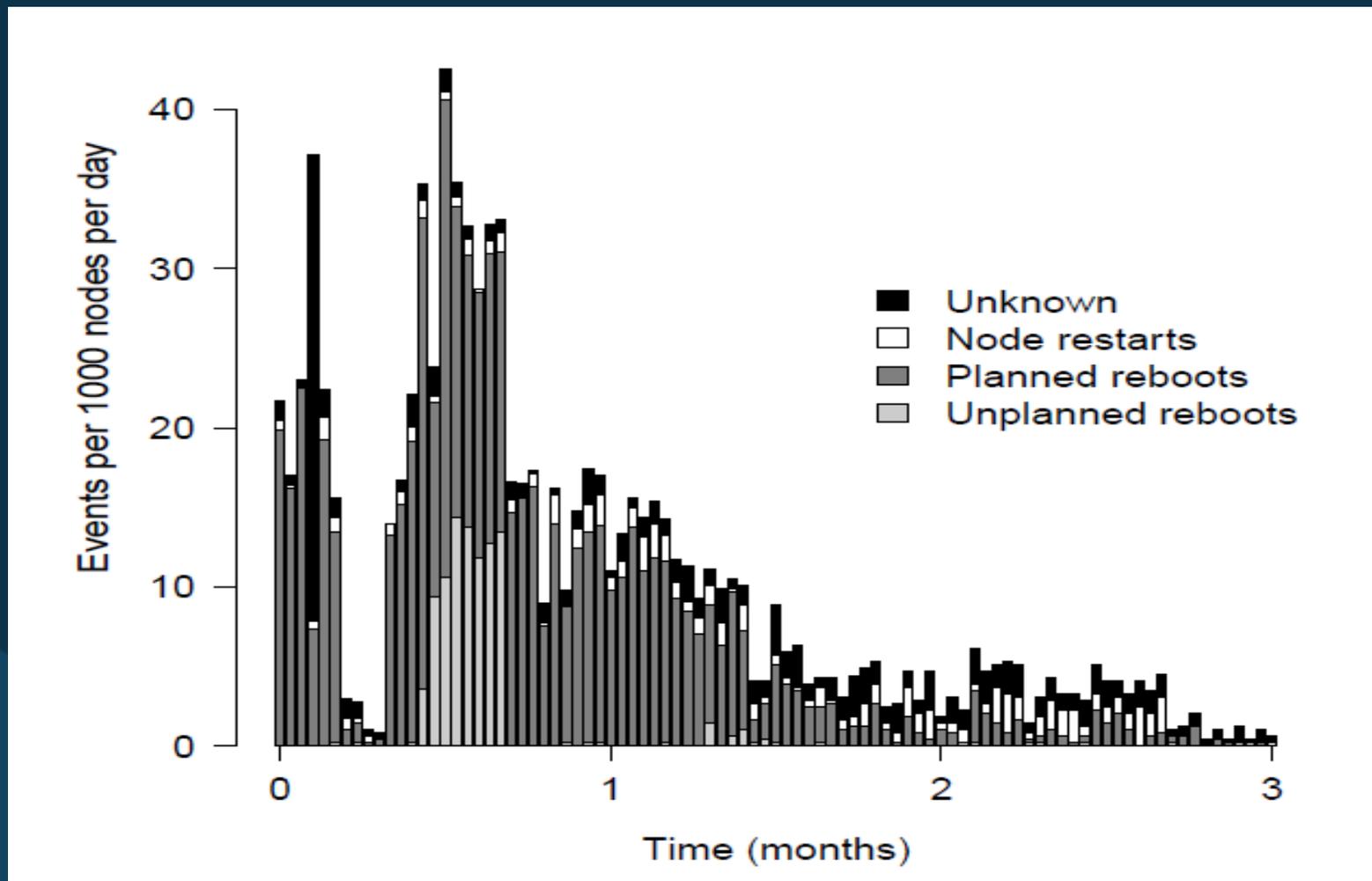
# Unavailability Event duration

Cumulative distribution function of node unavailability duration by cause



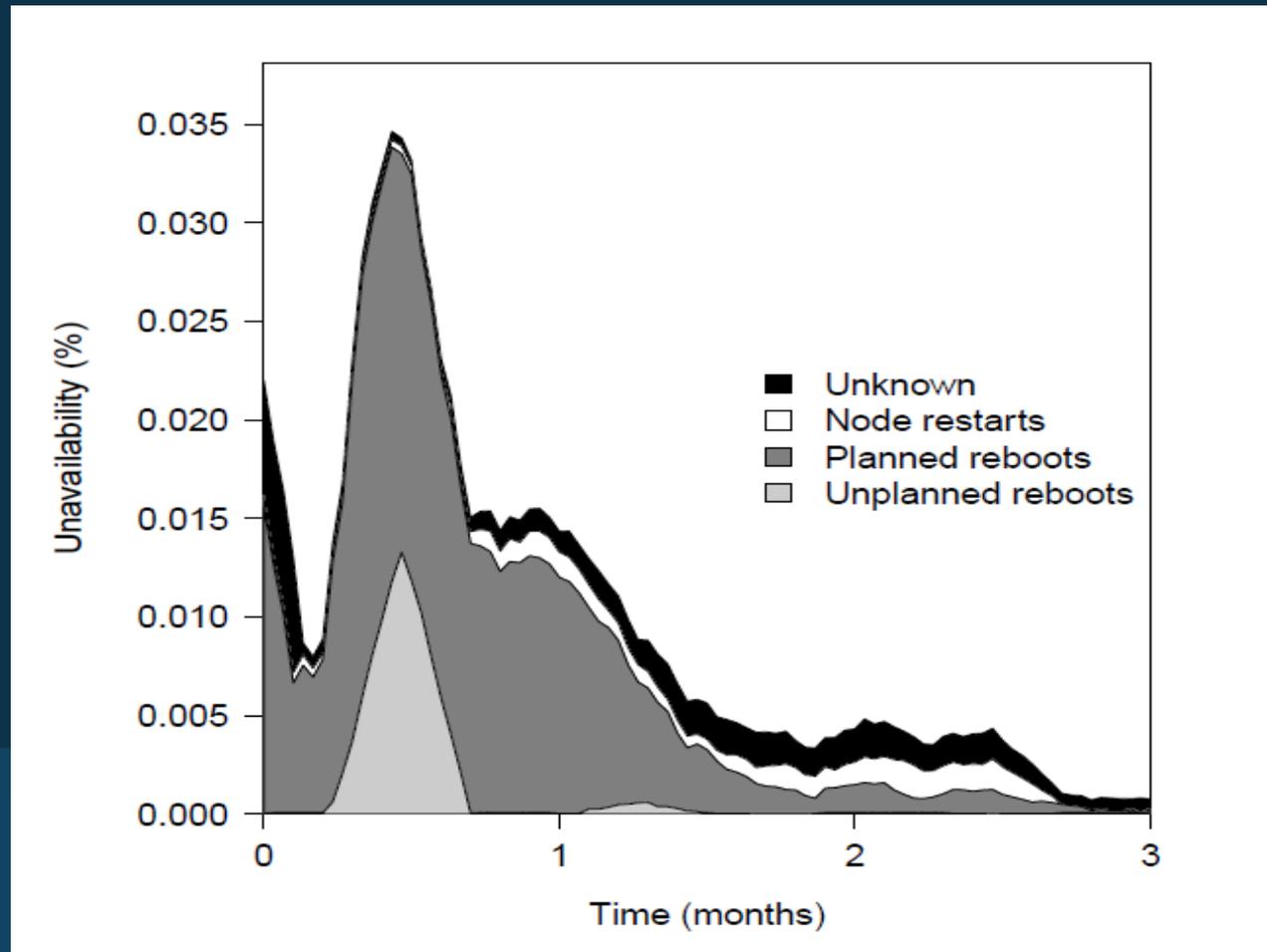
# Rate of events

Rate of events per 1000 nodes per day



# Storage node unavailability

Storage node unavailability computed with a one week rolling window



# Unavailability attributed to different failure causes, over the full set of cells

Cause	Unavailability (%) average / min / max
Node restarts	0.0139 / 0.0004 / 0.1295
Planned machine reboots	0.0154 / 0.0050 / 0.0563
Unplanned machine reboots	0.0025 / 0.0000 / 0.0122
Unknown	0.0142 / 0.0013 / 0.0454

Table 1: Unavailability attributed to different failure causes, over the full set of cells.

# Failure Bursts

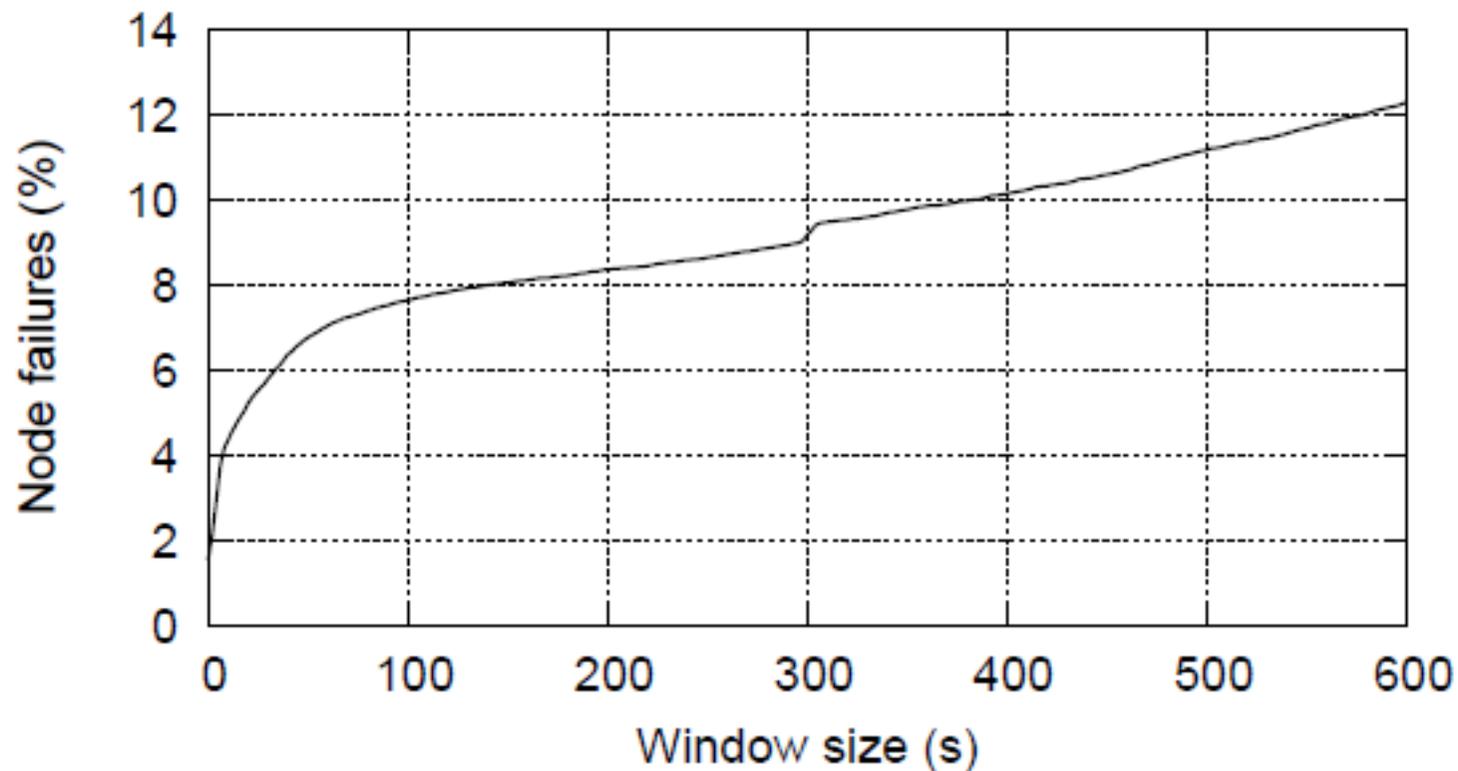
Often errors have tendency to happen together. It is critical to take into account the statistical behavior of correlated failures to understand data availability.

Failure burst is a set of failures, each one occurring within a time window  $w = 120s$ .

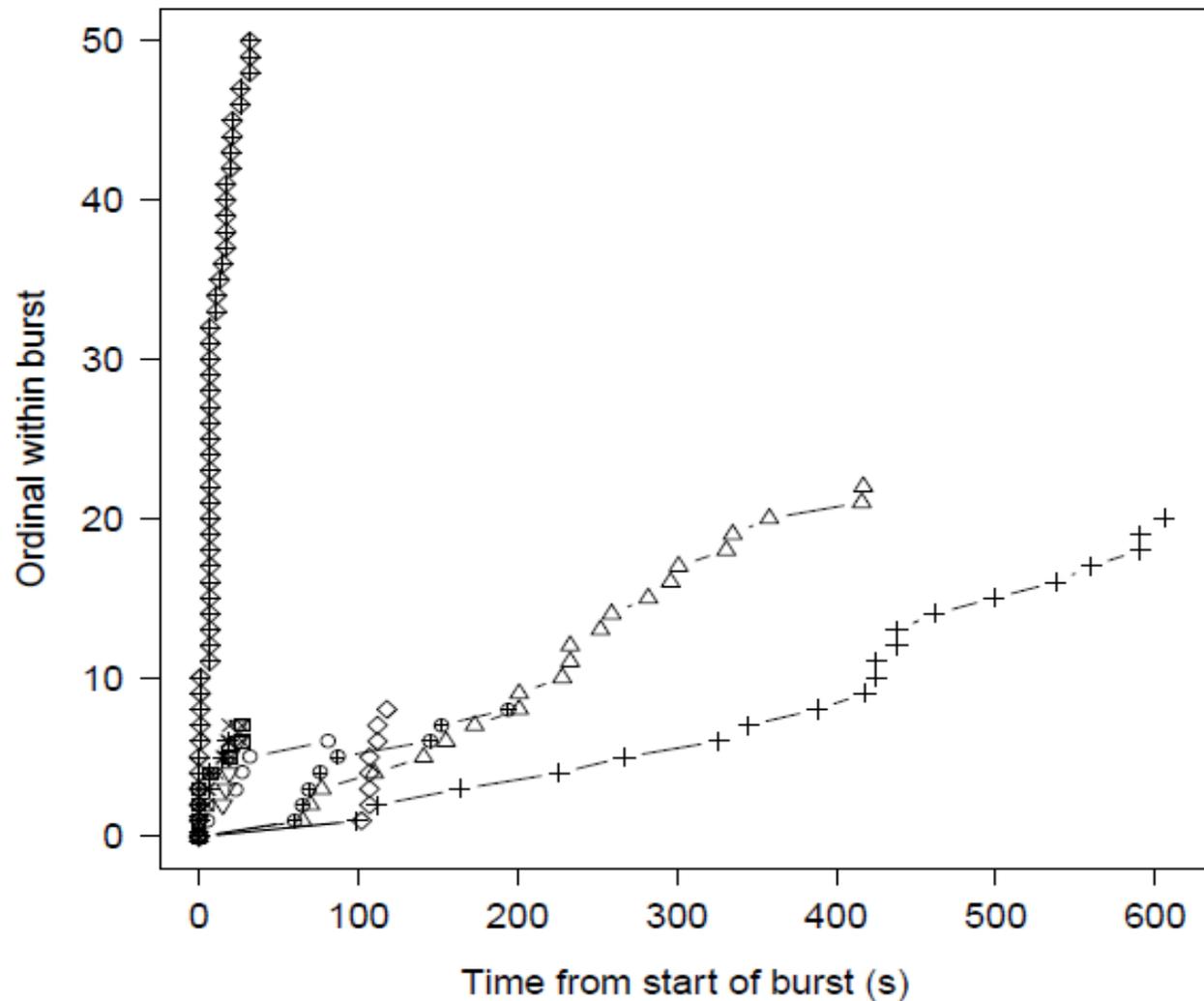
By failure domain, we mean a set of machines which we expect to simultaneously suffer from a common source of failure.

For that time window probability that two random failures will be included into a same failure burst is only 8.0% . The probability that random failure gets in a burst of at least 10 nodes is only 0.068%.

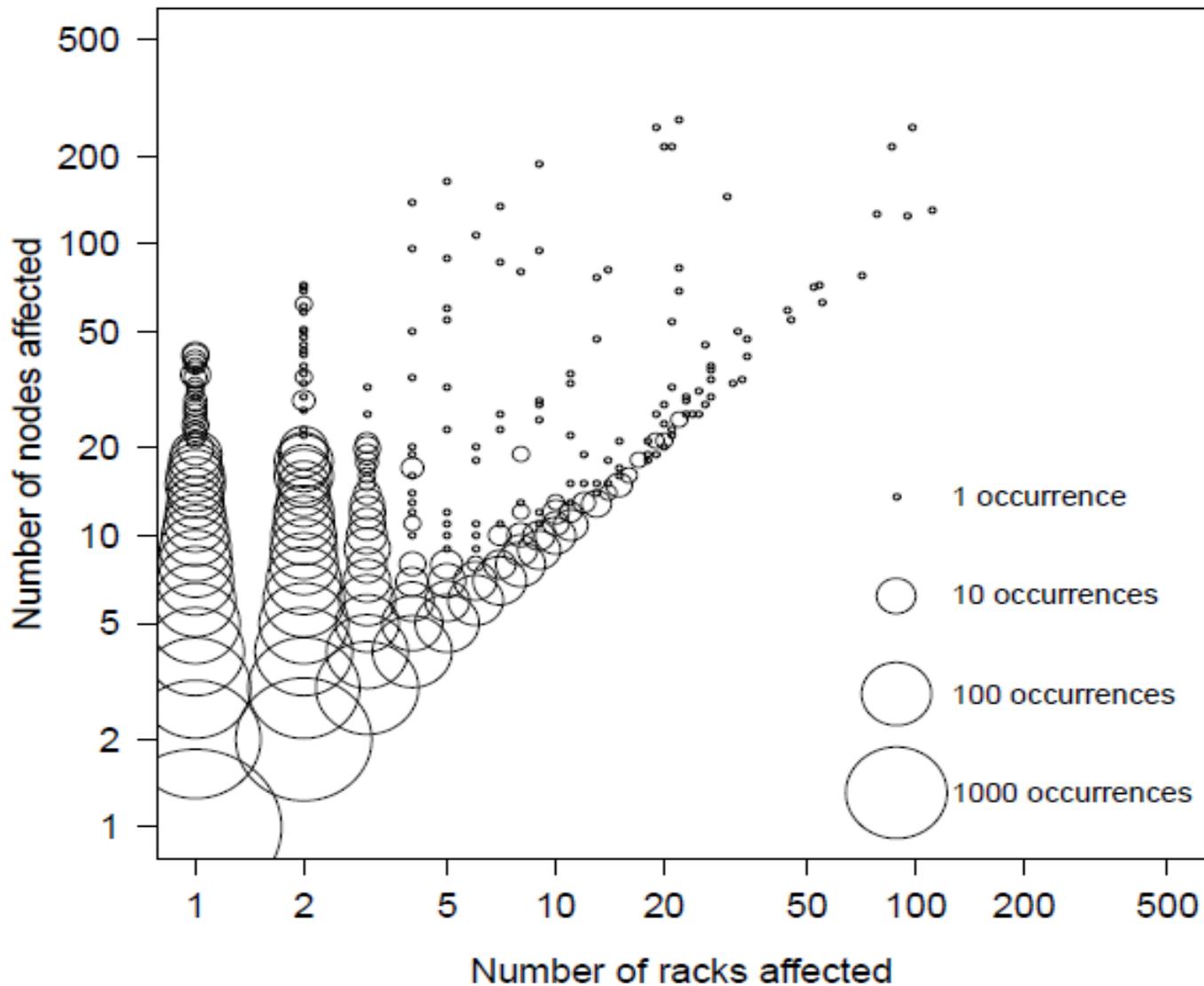
# Effect of the window size on the fraction on individual failures that get clustered into bursts of at least 10 nodes



# Development of failure bursts in one example cell



# Frequency of failure bursts sorted by racks and nodes affected



# Identifying domain-related failures

We encode a failure burst as a  $n$ -tuple  $(k_1, \dots, k_n)$ , where  $k_1 \leq k_2, \dots, k_{n-1} \leq k_n$ .  $k_i$  gives the number of nodes affected in the  $i$ -th rack, where the racks are ordered so that the values are increasing.

We define the rack-affinity score to be:

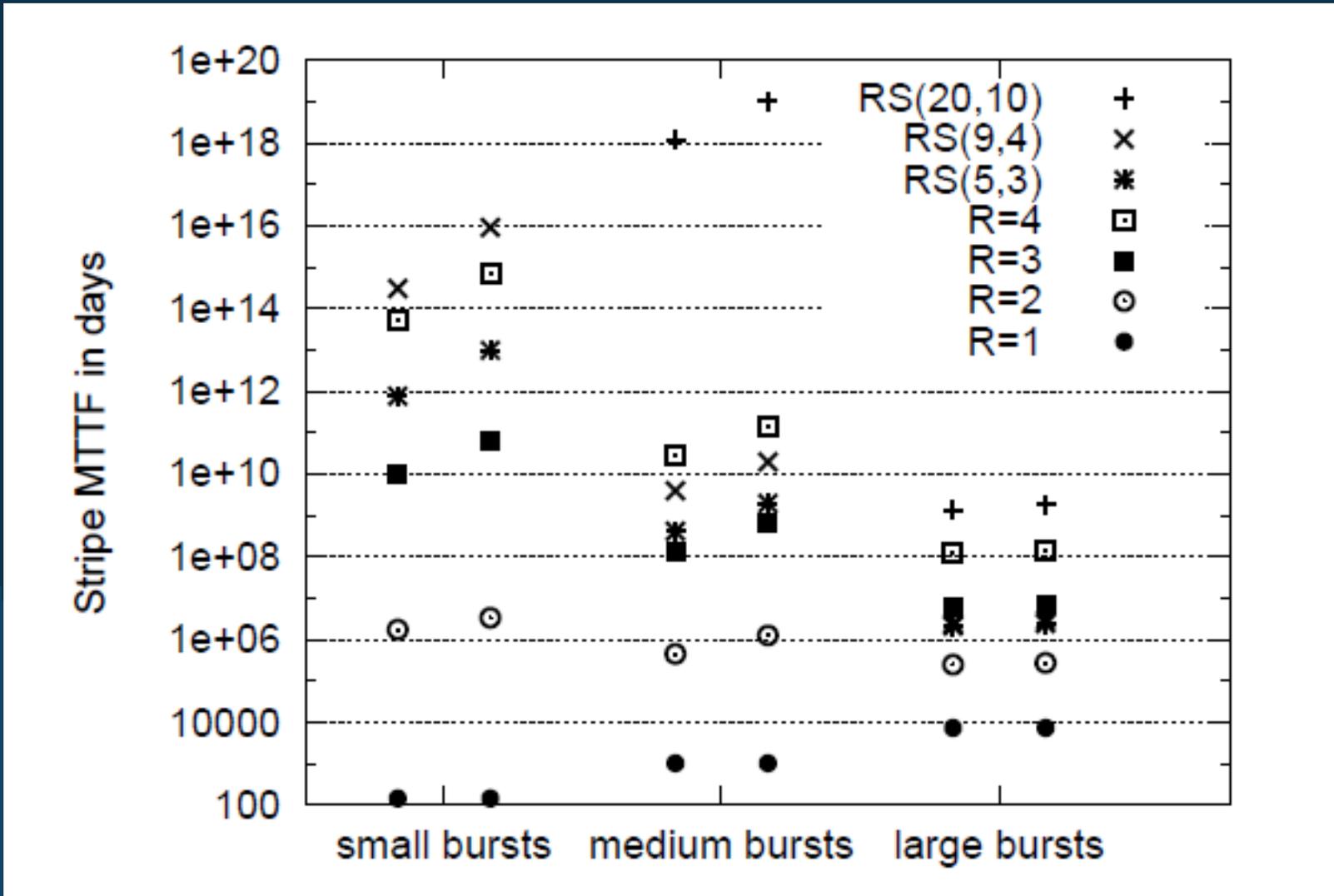
$$\sum_{i=1}^n \frac{k_i(k_i - 1)}{2}$$

# Data availability

# Data replication and recovery

Replication or erasure encoding schemes provide resilience to individual node failures. When a node failure causes the unavailability of a chunk within a stripe, we initiate a recovery operation for that chunk from the other available chunks remaining in the stripe.

Stripe MTTF due to different burst sizes. Burst sizes are defined as a fraction of all nodes. The left column represents uniform random placement, and the right column represents rack-aware placement.



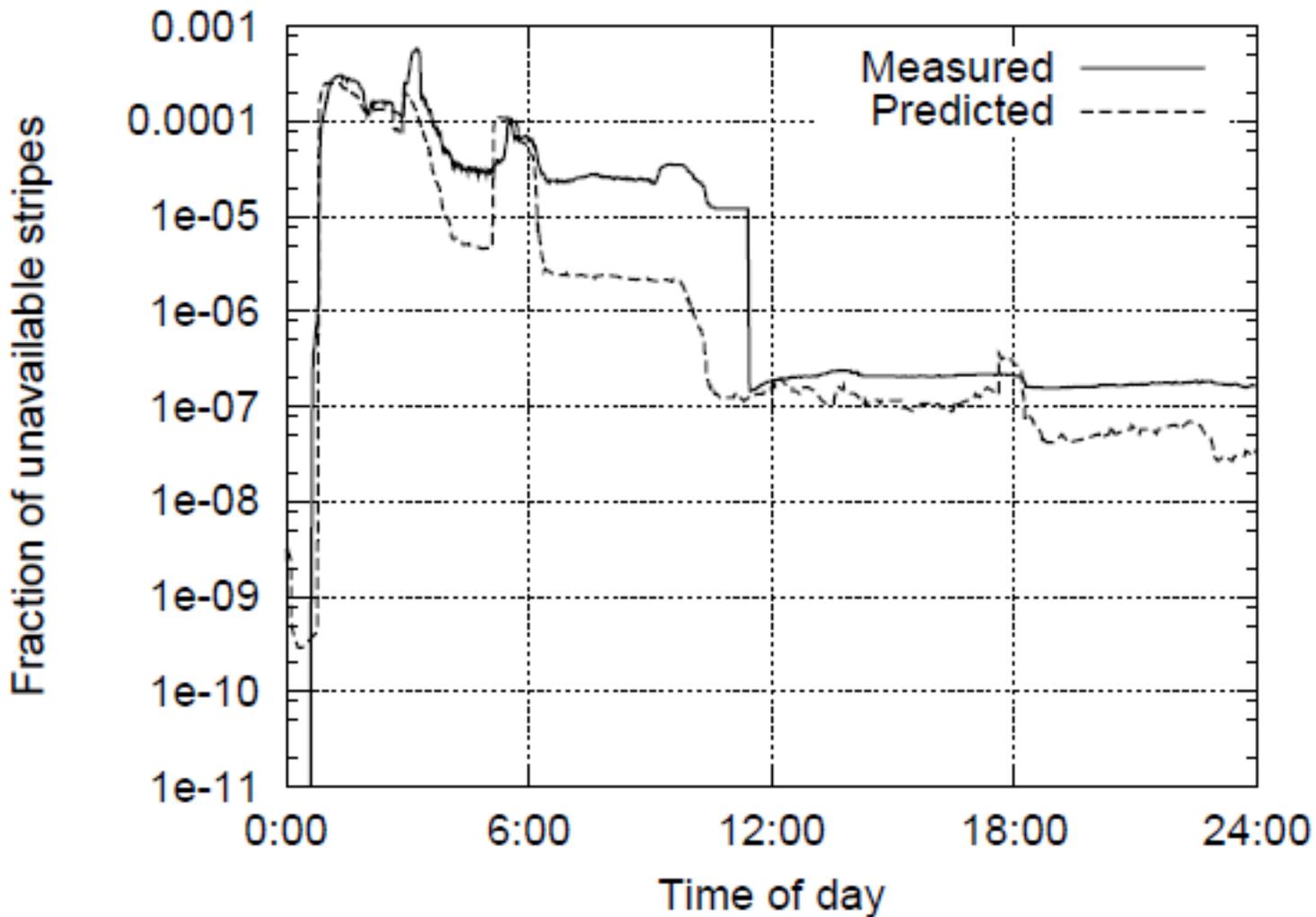
# Trace-based simulation

We can replay observed or synthetic sequences of node failures and calculate the resulting impact on stripe availability.

We are interested in the expected number of stripes that are unavailable for at least 15 minutes, as a function of time.

We can use combinatorial calculations to obtain the expected number of unavailable stripes given a set of down nodes.

# Unavailability prediction over time for a particular cell



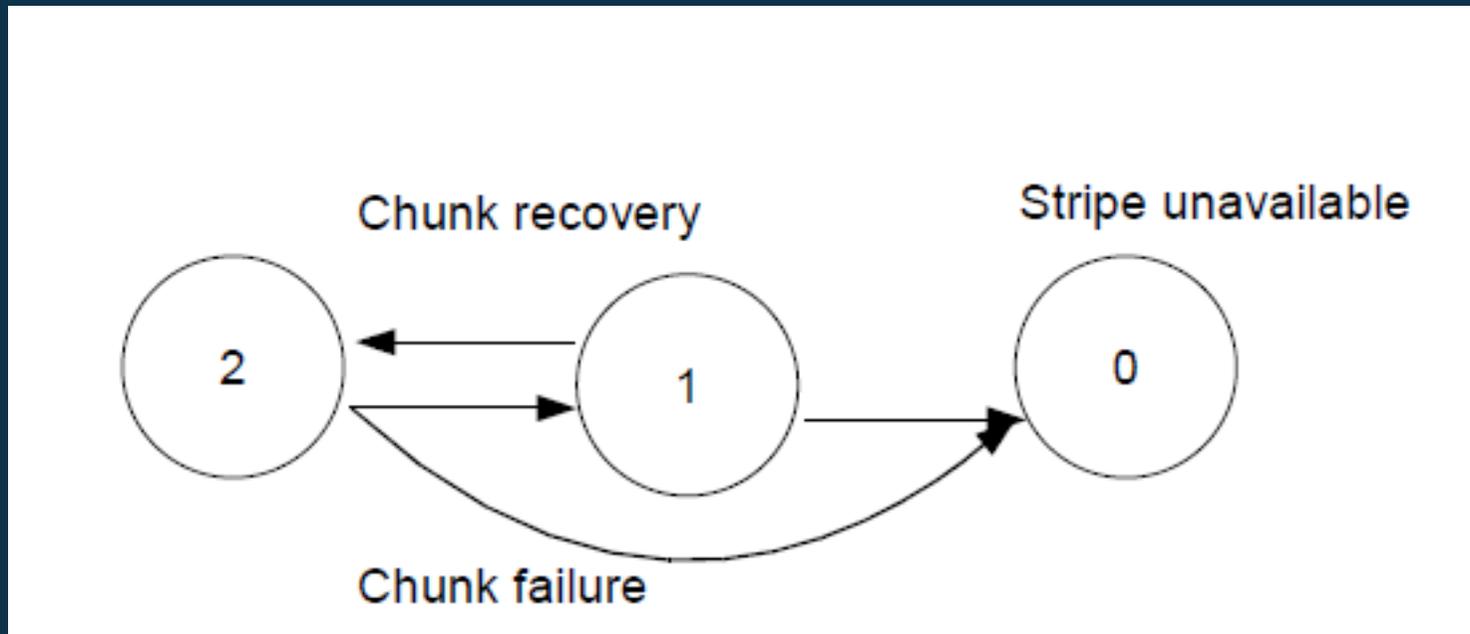
# Markov Model of Stripe Availability

If we look at the individual stripe we can simulate its availability as a Markov chain.

The state of stripe is represented by the number of available chunks.

The Markov chain transitions are specified by the rates at which a stripes moves from one state to another, due to chunk failures and recoveries.

# The Markov chain for a stripe encoded using $R=2$



# Markov model validation

Markov model was validated by comparing predicted MTTF with actual MTTF.

Although the difference was significant it was in the same order of magnitude, which is enough.

In one cell MTTF was  $1.76E+6$  days, while predicted MTTF was  $5E+6$  days.

In another it was  $29.52E+8$  days, while model predicted  $5.77E+8$ .

# Stripe MTTF in days, corresponding to various data redundancy policies

Policy (% overhead)	MTTF(days) with correlated failures	MTTF(days) w/o correlated failures
$R = 2$ (100)	$1.47E + 5$	$4.99E + 05$
$R = 3$ (200)	$6.82E + 6$	$1.35E + 09$
$R = 4$ (300)	$1.40E + 8$	$2.75E + 12$
$R = 5$ (400)	$2.41E + 9$	$8.98E + 15$
$RS(4, 2)$ (50)	$1.80E + 6$	$1.35E + 09$
$RS(6, 3)$ (50)	$1.03E + 7$	$4.95E + 12$
$RS(9, 4)$ (44)	$2.39E + 6$	$9.01E + 15$
$RS(8, 4)$ (50)	$5.11E + 7$	$1.80E + 16$

# Stripe MTTF and inter-cell bandwidth, for various multi-cell schemes and inter-cell recovery times

Policy (recovery time)	MTTF (days)	Bandwidth (per PB)
$R = 2 \times 2(1\text{day})$	$1.08E + 10$	6.8MB/day
$R = 2 \times 2(1\text{hr})$	$2.58E + 11$	6.8MB/day
$RS(6, 3) \times 2(1\text{day})$	$5.32E + 13$	97KB/day
$RS(6, 3) \times 2(1\text{hr})$	$1.22E + 15$	97KB/day

# Conclusions

Data from Google clusters implies that correlation among node failures dwarfs all other contributions to unavailability.

Paper presented simple time-window-based method to group failure events. There were also developed analytical models to reason about past and future availability in Google cells.

Inside Google, the analysis described in this paper has provided a picture of data availability at a finer granularity than previously measured.