Training on Log-based Field Failure Data Analysis

Domenico Cotroneo
cotroneo@unina.it

Data-driven Analysis

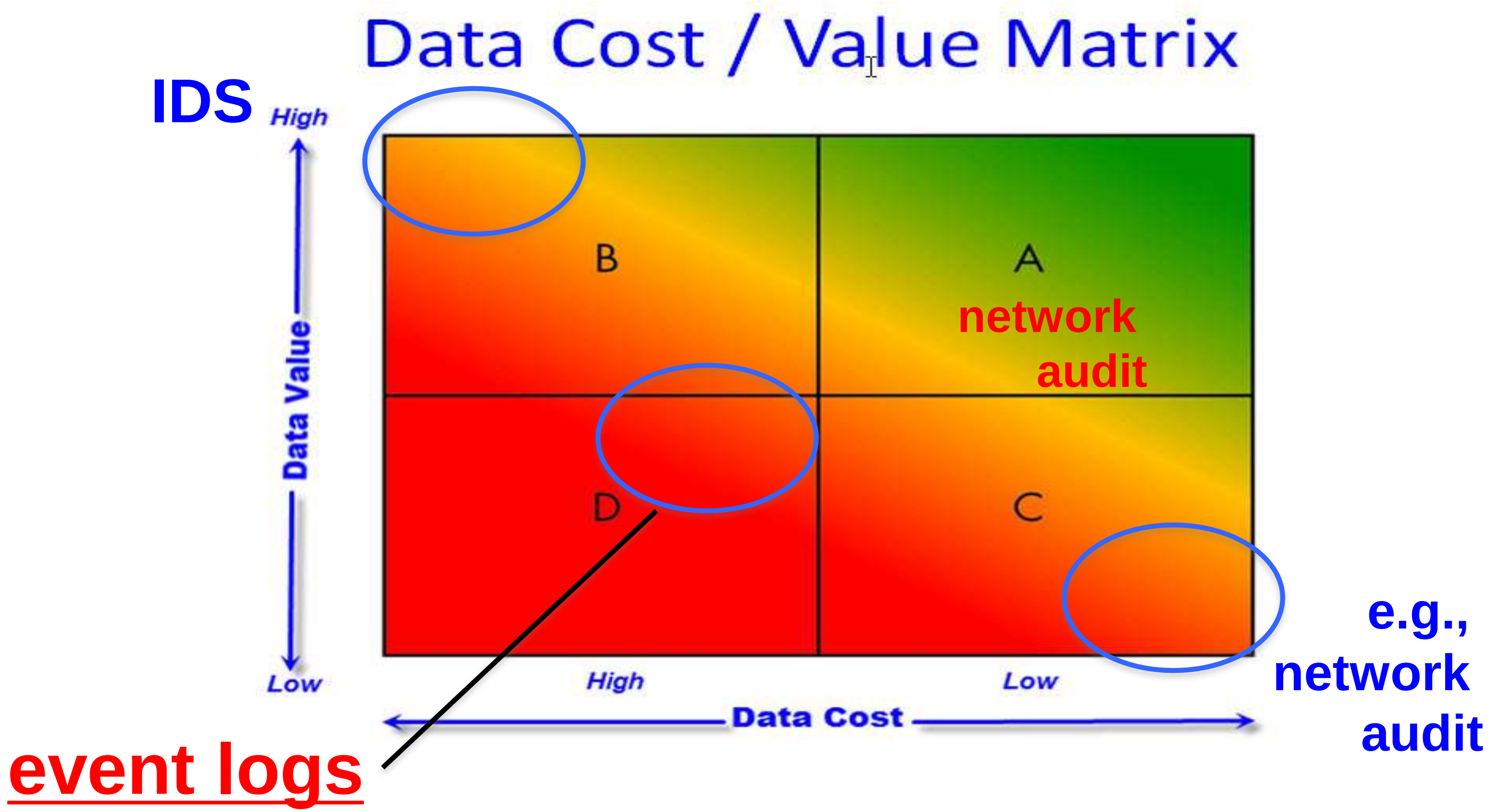
- Textual logs and numeric data produced by applications, systems, and IDSs are a key resource to:
  - **characterize failures**;
  - predict problems;
  - pinpoint security attacks;
  - improve the system;

- **Data are available in many critical systems!**





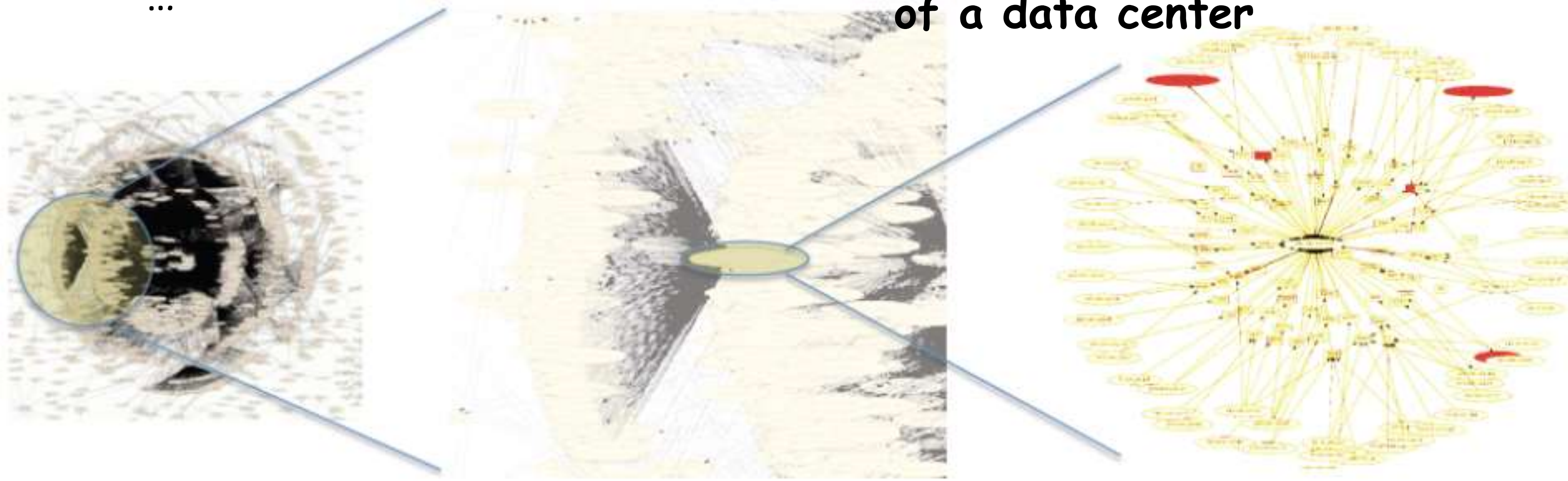
Not all the data is the same



Challenges

- **Dealing with operational data is extremely hard:**
  - increasing network traffic and failure rates;
  - increasing data volumes;
  - data quality and heterogeneity;
  - lack of systematic procedures;
  - low automation degree;
  - …

5*min* in-out traffic of a data center



What is the training lesson about

- Conducting a log-based FFDA of complex, large-scale systems.
- **Real data** collected from:
  - **Mercury** at the National Center for Supercomputing Application (NCSA at University of Illinois);
  - **BlueGene/L** at the Lawrence Livermore National Labs (LLNL, CA, USA).

Goals

- Objectives: to learn the basics in
  - data classification/analysis;
  - determining the coalescence window to group a set of error entries;
  - performing a failure analysis;
  - reliability modelling via well-know distributions;
  - analysis at system and node level;
  - ...

Why is it hard?

- Monitoring in a datacenter day …

7M network packets (netflows)

25K SSH connections

2K IDS alerts

400K log entries (syslog)

300 major security alerts

- Even if the data is available, the real challenge is still being actually able to use it!

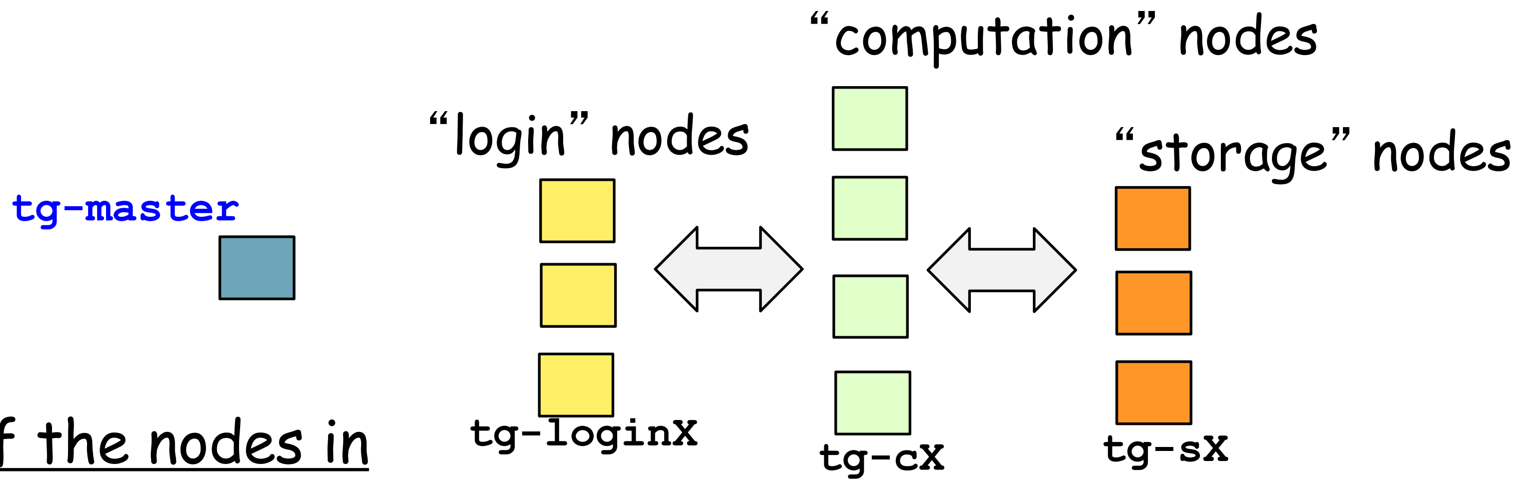
Why?

- Failure characterization;
- Failure correlation/localization;
- Failure prediction;
- Cause analysis;
- Check-pointing tuning;
- Failure-aware job allocation;
- ...

Correct understanding of the failure behavior is a critical task!

Mercury cluster: overview

- Mercury consists of IBM nodes. The cluster has a 3-layer architecture + a management node (`tg-master`)



ID of the nodes in the log (X is an integer)

- Myrinet. The nodes run a RedHat 9.0 OS
- Entries have been collected via the syslog daemon

Mercury event log

- 80,854 log filtered and anonimized entries (*only* FATAL *error entries are given in the log*)
- Format:
    - Timestamp
    - Originating node
    - Originating subsystem
    - Text-free message

```
1167667229 tg-c238 DEV
Component Info: Vendor Id =* *, Device Id =* *, Class Code =*
*, Seg Bus Dev Func =* * * * *
```

Examples of error entries

- [DEV]

  +Platform PCI Component Error Info Section

- [MEM]

  Mem Error Detail: Physical Address *, Address Mask:*, Node:*, Card:*, Module: *, Bank:*, Device:*,Row:*, Column:*

- [NET]

  connection down

- [I-O] - error, dev *:* (hda), sector *

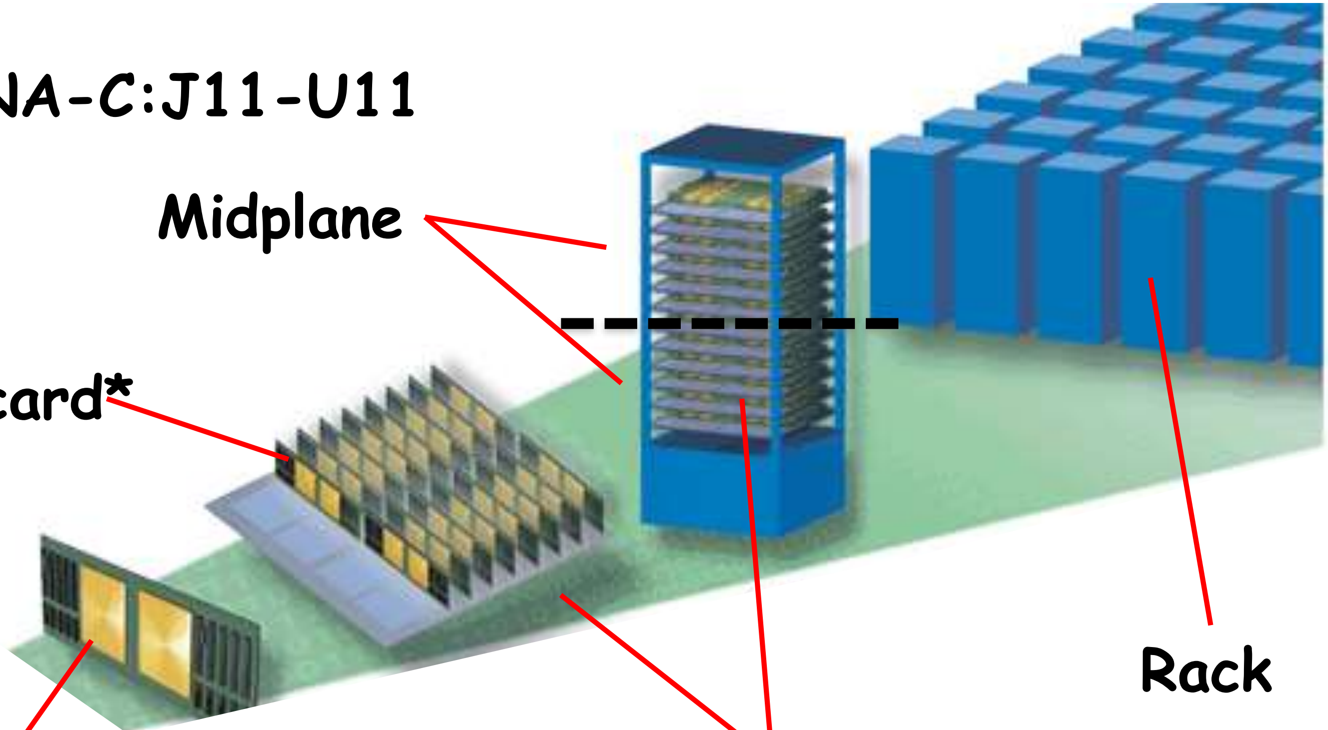
  - hda: packet command error: error=*

- [PRO]

  +BEGIN HARDWARE ERROR STATE AT CMC

BG/L Sample Diagram

R27-M0-NA-C:J11-U11



*16 cards / node (J02-J17)
for each midplane N0, N4, N8, NC
have and additional I/O card (J18)

BG/L event log*

- 125,624 filtered log entries (*only* FATAL *error entries are given in the log*)

- Format:
  - Timestamp
  - Originating node
  - Originating card
  - Text-free message

```
1128621351 R04-M0-N0 J18-U01
Lustre mount FAILED : bglio66 : block_id : location
```

*Log made publicly available by J. Stearley from Sandia National Labs
http://www.cs.sandia.gov/~jrstear/logs/bgl.gz

Examples of error entries

1135619785 R72-M1-NC J09-U11 parity error in read queue 0...................0

1135780093 R35-M0-N3 J12-U01 machine check interrupt (bit=0x1d): L2 dcache unit write data parity error

1132173448 R46-M1-N8 J04-U11 Error receiving packet on tree network, expecting type 57 instead of type 3 (softheader=0030114e 602d0003 00000002 00000000) PSR0=10011f01 PSR1=00000000 PRXF=00000002 PIXF=00000007

...

Tools usage and sample analysis*

*Results shown in the rest of the presentation are based on a subset of 10,000 entries of Mercury log (MercuryErrorLogTEST.txt)

Tools needed for the analysis

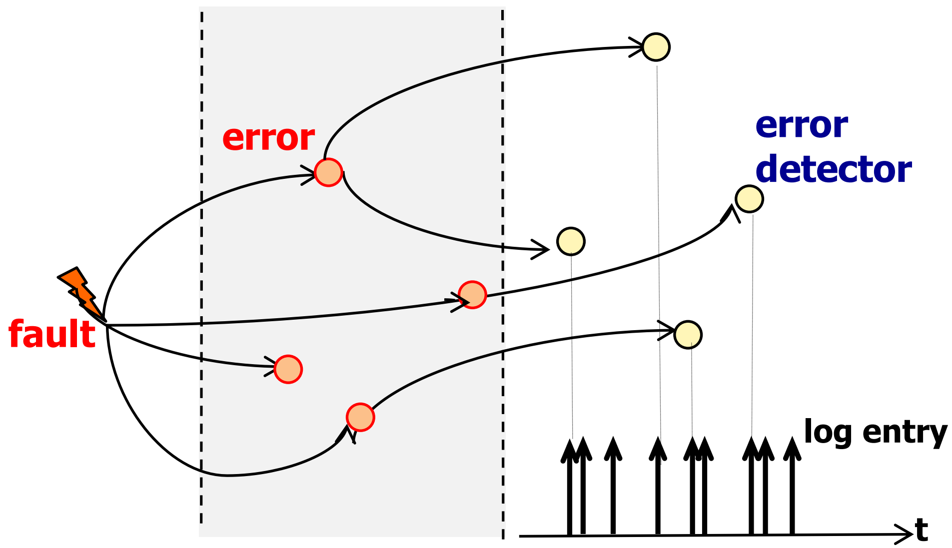
- A suite of home-made BASH scripts (tuple count, grouping of the error entries; total 4 scripts)
  - the basic scripts can be used sequentially to perform the FFDA of the log;
  - the usage of the scripts is described in the presentation;
  - students are free to modify/improve or to write further scripts to automate the analysis;
  - a baseline ffdatoolset.zip archive is provided to the students.
- Matlab & statistics toolbox (TTF estimation, curve fitting, …)

How many failures?

- The same problem is reported multiple times in the log
- Entries that are possibly related to the same problem must be grouped (or coalesced).

```
$ head -n 10 MercuryErrorLogTEST.txt
1167637660 tg-c645 PRO  tg-c645__+BEGIN HARDWARE ERROR STATE AT CMC
1167637660 tg-c645 PRO  tg-c645__Device Error Info Section
[… omissis …]
1167637720 tg-c645 PRO  tg-c645__Device Error Info Section
1167637720 tg-c645 PRO  tg-c645__Error Map: *
1167655228 tg-c238 DEV  tg-c238__+BEGIN HARDWARE ERROR STATE AT CPE
1167655228 tg-c238 DEV  tg-c238__+Platform PCI Component Error Info Section
[… omissis …]
```

How many failures?



- The same fault can be activated multiple times and generate multiple errors that propagate within the system.

Determining the coalescence window

How many failures?

- The script tupleCount_func_CWIN.sh analyzes how the *tuple count varies for increasing values of the coalescence window*
- Execution takes several *min/hour* depending on the number of entries in the log*
- The output (CWIN,tuple-count pairs) is saved in the tupleCount-<log_filename>.txt file

```
$./tupleCount_func_CWIN.sh MercuryErrorLogTEST.txt tentative-Cwin.txt

$ cat tupleCount-logMERCURY-TEST.txt
1          2049
2          1238
3          1063
4          1033
[... omissis ...]
```

CWIN size

count

*around 1*hour* for the **MercuryErrorLogTEST.txt** log; estimated around 6/7*hours* for the exercise log (run with an Intel Core2 Duo @2.4Ghz processor)

Determining the "optimal" CWIN

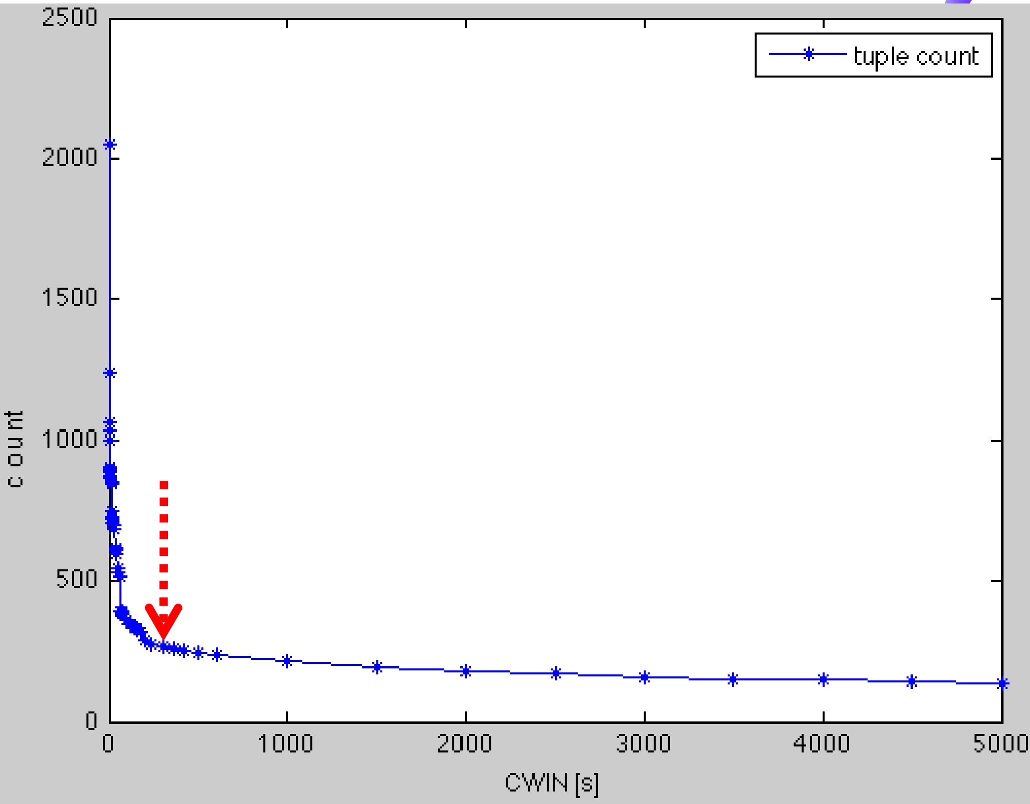
- Plotting the tuple-count curve

```
>> load tupleCount-MercuryErrorLogTEST.txt
>> plot(tupleCount_MercuryErrorLogTEST (:,1),tupleCount_MercuryErrorLogTEST
(:,2),'-*b');
>> xlabel('CWIN [s]');ylabel('count');
>> legend('tuple count');
```

- <u>Rule:</u> a *good CWIN* is right after the "knee" of the curve

<u>CWIN: 240*sec* (4min)</u>

<u>276 tuples</u>

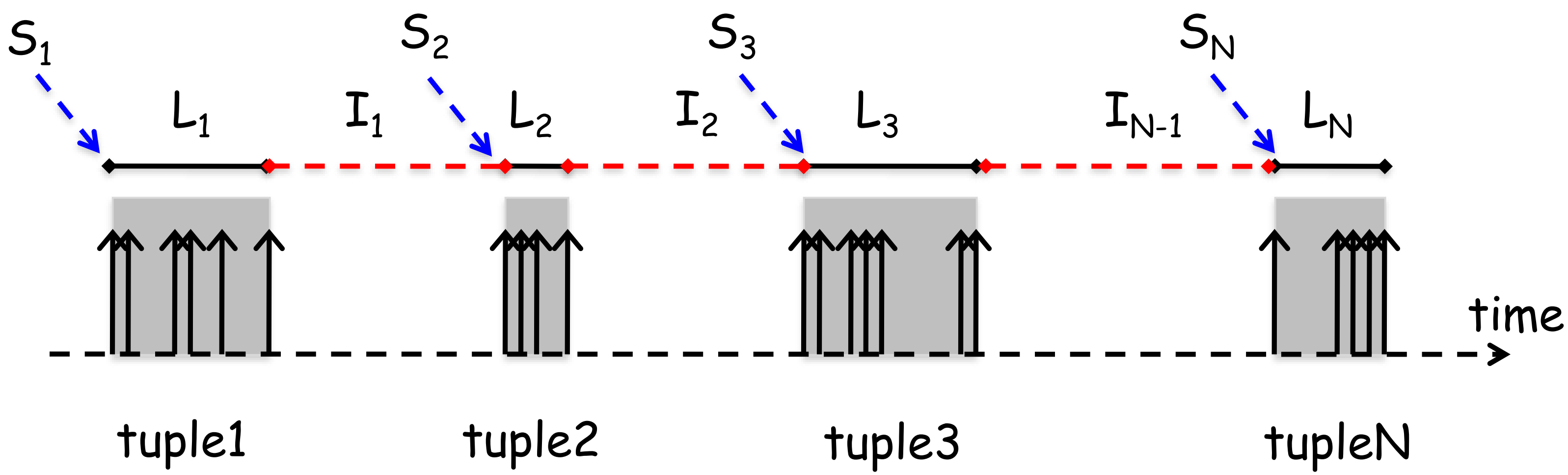


Grouping the entries with the selected CWIN

- The script tupling_with_CWIN.sh coalesces the entries with the selected CWIN (the script takes around 2*min*).
- Output (summary files + content of each individual tuple) is saved in the directory tupling_<log-filename>-<CWIN>

```
$ ./tupling_with_Cwin.sh MercuryErrorLogTEST.txt 240
*** Grouping the entries for CWIN=240 ***
  - creating tuple # 1
  - creating tuple # 2
  [...]
$ cd tupling_MercuryErrorLogTEST-240/
$ ls
interarrivals.txt    tuple_16    tuple_30    tuple_45
lenghts.txt          tuple_17    tuple_31    tuple_46
startingPoints.txt   tuple_18    tuple_32    tuple_47
tuple_1              tuple_76    tuple_90    [...]
```

Output files



- L_1 , L_2 , … , L_N tuple lenghts (*lenghts.txt*)
- I_1 , I_2 , … , I_{N-1} interarrivals (*interarrivals.txt*)
- S_1, S_2 , … , S_N timestamps of the init point of the tuples (*1st column ofstartingPoints.txt*)

* NOTE: each interarrival is a TTF sample *

Issue #1: truncations

- A single failure might be broken into multiple tuples

Tuple #4

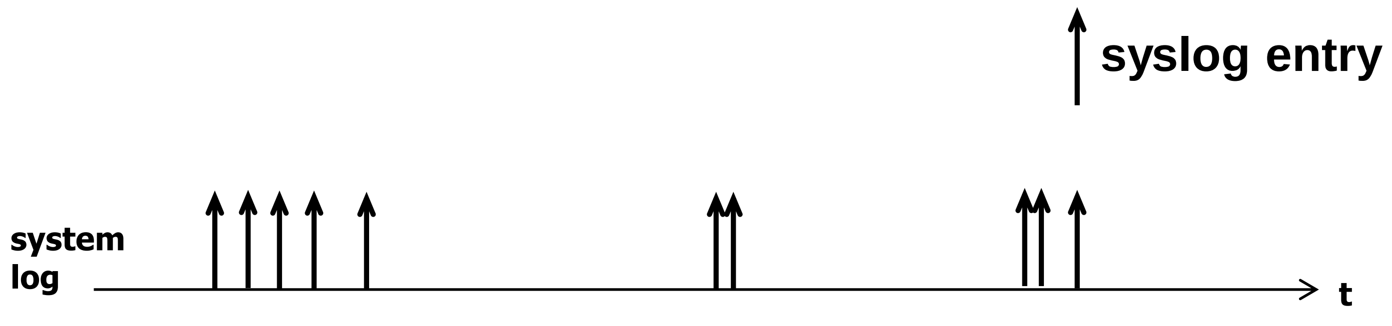
…
1167657550 tg-c238 MEM Physical Address x, Address Mask: x, Node: x, Card: x, Module: x, Bank: x, Device: x, Row: x, Column: x,
1167657550 tg-c238 MEM Physical Address x, Address Mask: x, Node: x, Card: x, Module: x, Bank: x, Device: x, Row: x, Column: x,

Tuple #5

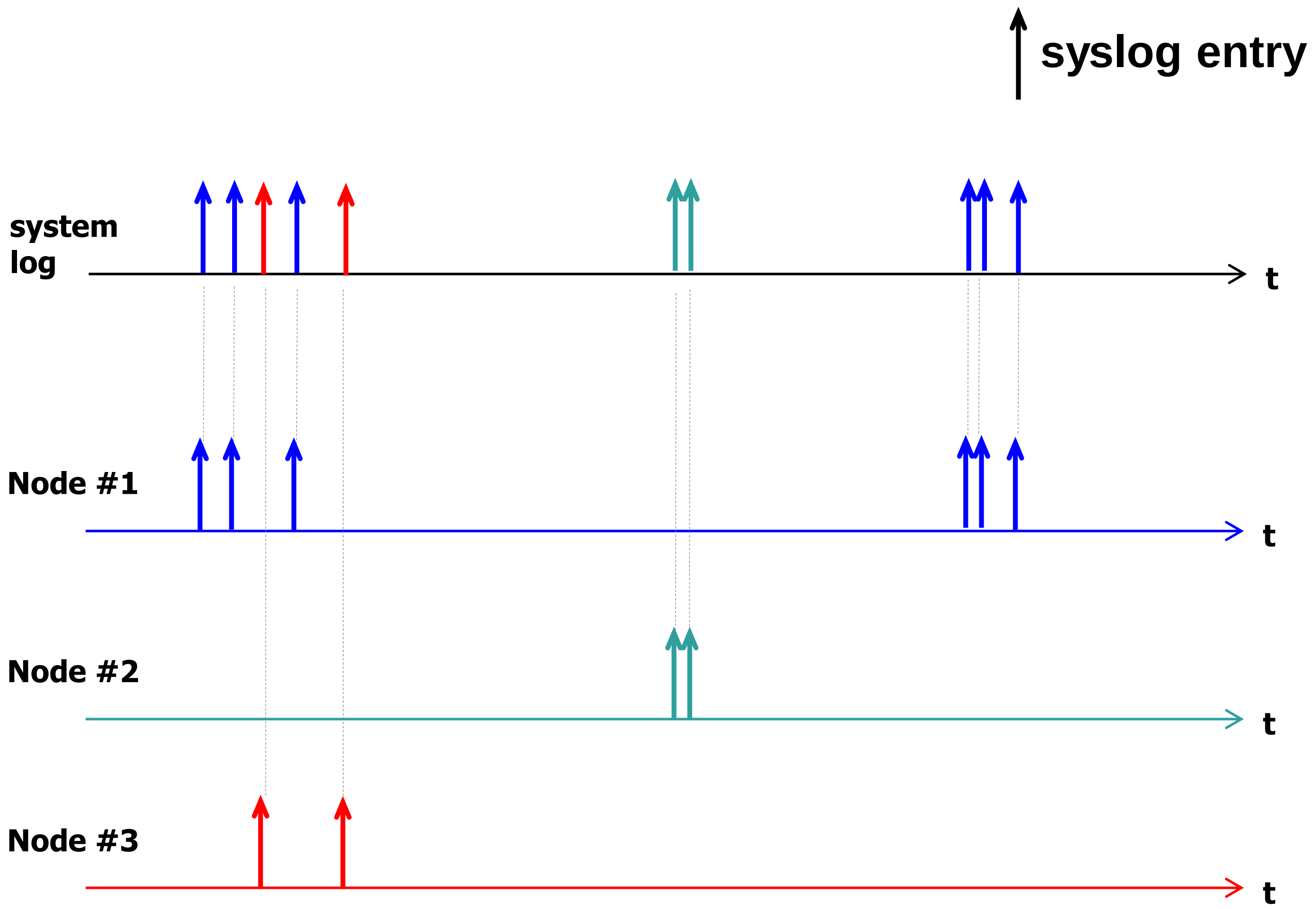
1167657941 tg-c238 DEV +BEGIN HARDWARE ERROR STATE AT CPE
1167657941 tg-c238 DEV +Platform PCI Component Error Info Section

- Time distance between tuples is 390*sec*

Issue #2: collisions

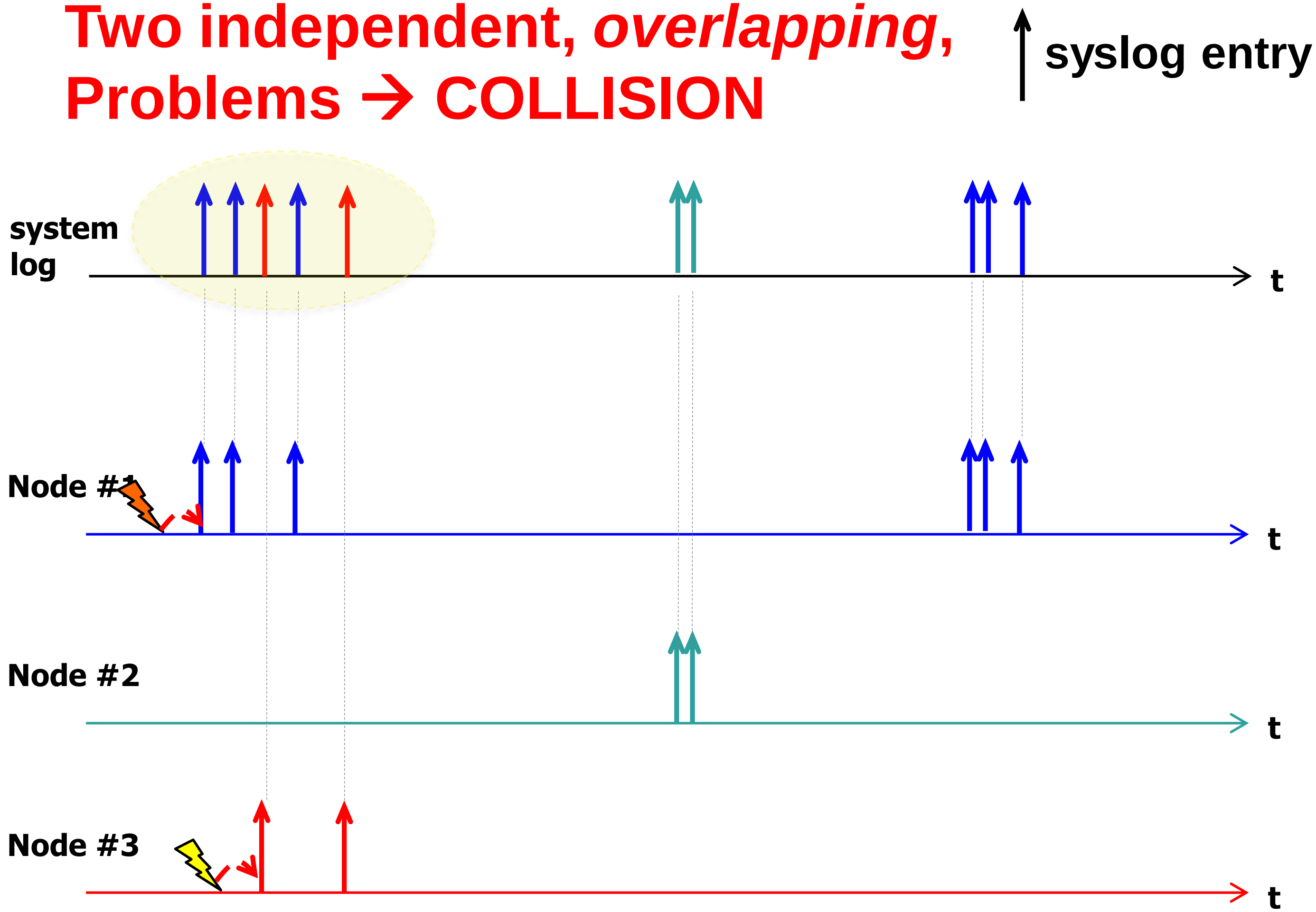


Issue #2: collisions

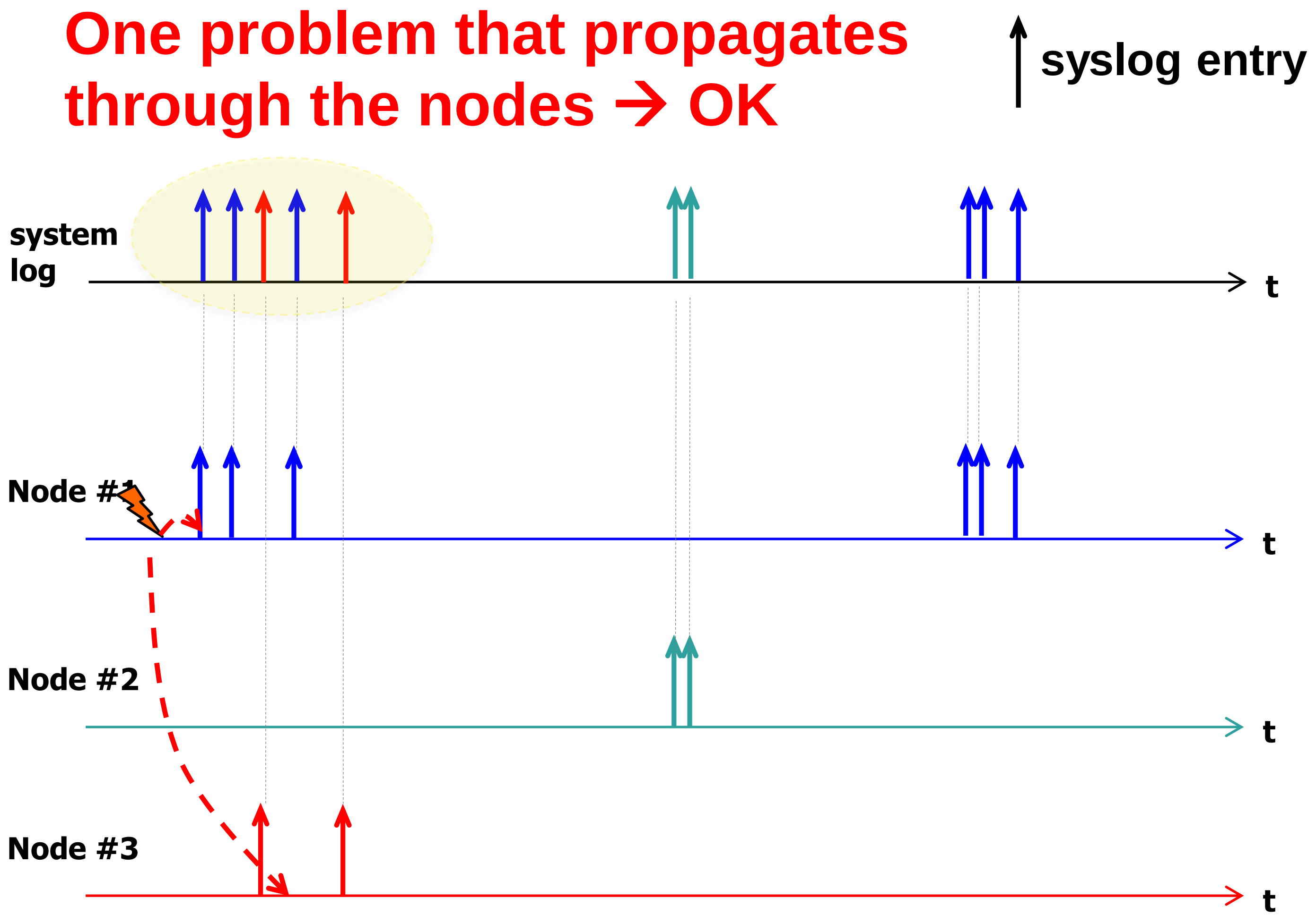


Issue #2: collisions

Two independent, *overlapping*, Problems → COLLISION



Issue #2: collisions



Example

- Tuple #61 includes entries produced by 7 nodes! The tuple lasts 1244 sec (20 min)

- Are the events correlated?

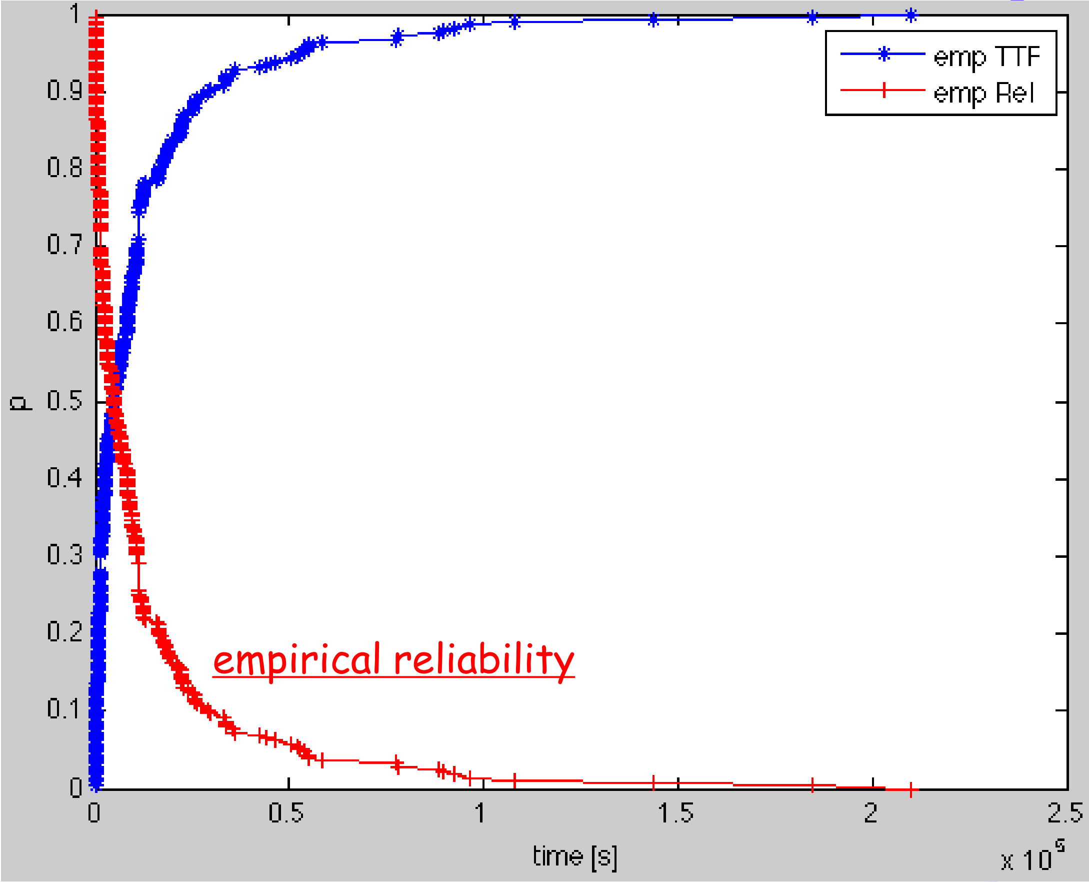
3	tg-c669
3	tg-login1
144	tg-login2
144	tg-login3
2	tg-login4
8	tg-s131
28	tg-s159

Reliability modeling

TTF: empirical distribution

- The interarrivals.txt file is imported in Matlab to obtain the *empirical CDF* of the TTF of the system.

```
>> load interarrivals.txt
>> [y,t]=cdfcalc(interarrivals);
>> empTTF=y(2:size(y,1));
>> empRel=1-empTTF;
>> plot(t,empTTF,'-*b',t,empRel,'-+r');
>> xlabel('time [s]'); ylabel('p');
>> legend('emp TTF', 'emp Rel');
```

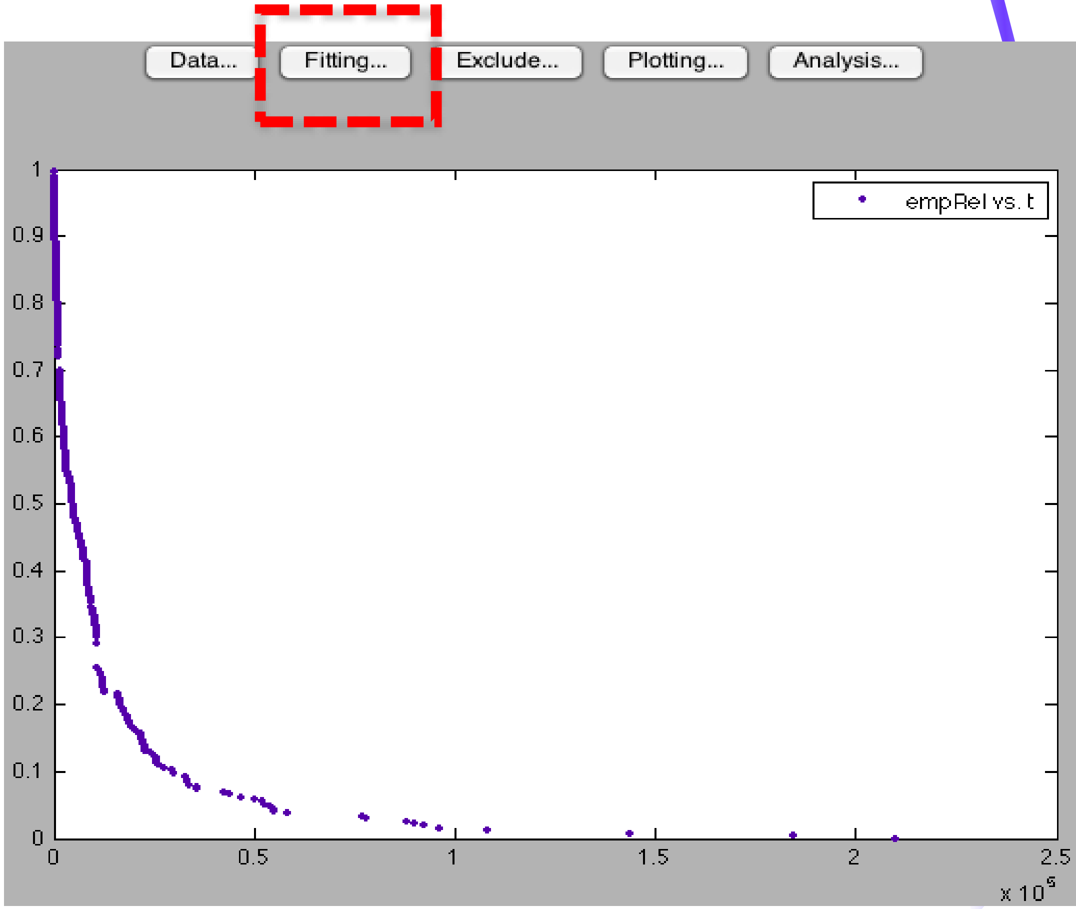


- Given t, TTF(t) is the probability that the system exhibits a failure after t*sec* since the last fail. occurred.

Fitting the empirical reliability

- The cftool (statistics toolbox) can be used to fit a theoretical distribution (thRel) to the empirical (empRel) one ("Fitting" button)

```
>> cftool(t,empRel);
```



- Tentative type of distribution:

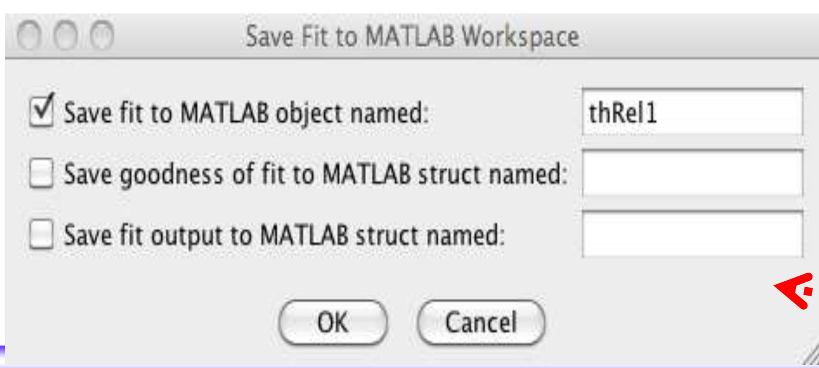
exponential

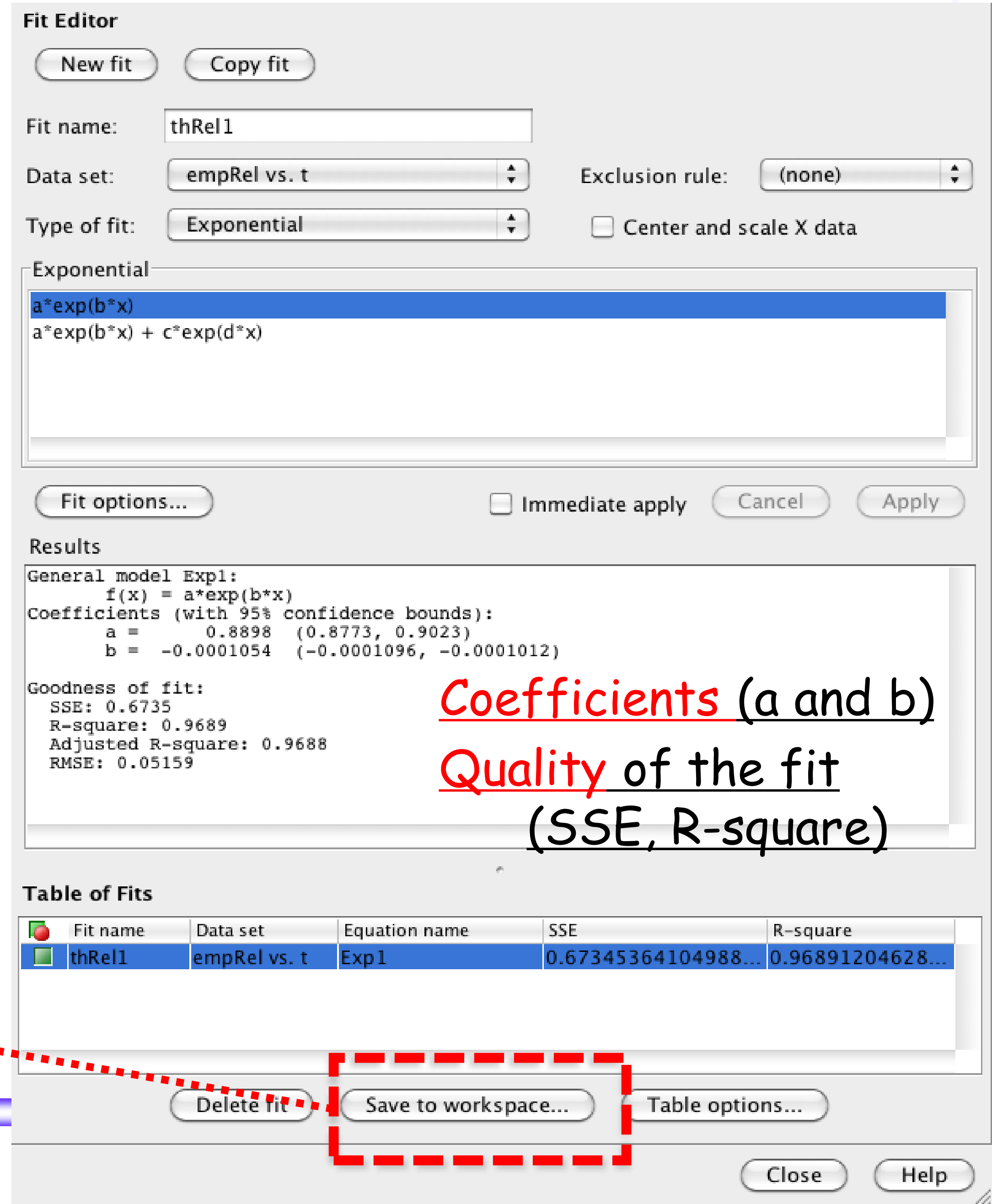
$thRel(t) = ae^{bt}$ $a \geq 0$

$b < 0$

Using the "Fit Editor"

1) Press "New fit"
2) Give a name to the fit
3) Select the type of fit (e.g., exponential)
4) Select the equation
5) Press "Apply"
6) Save the results to Matlab workspace





Assessing the quality of the fit

- *Is the theoretical distribution a good fit for the empirical one?*

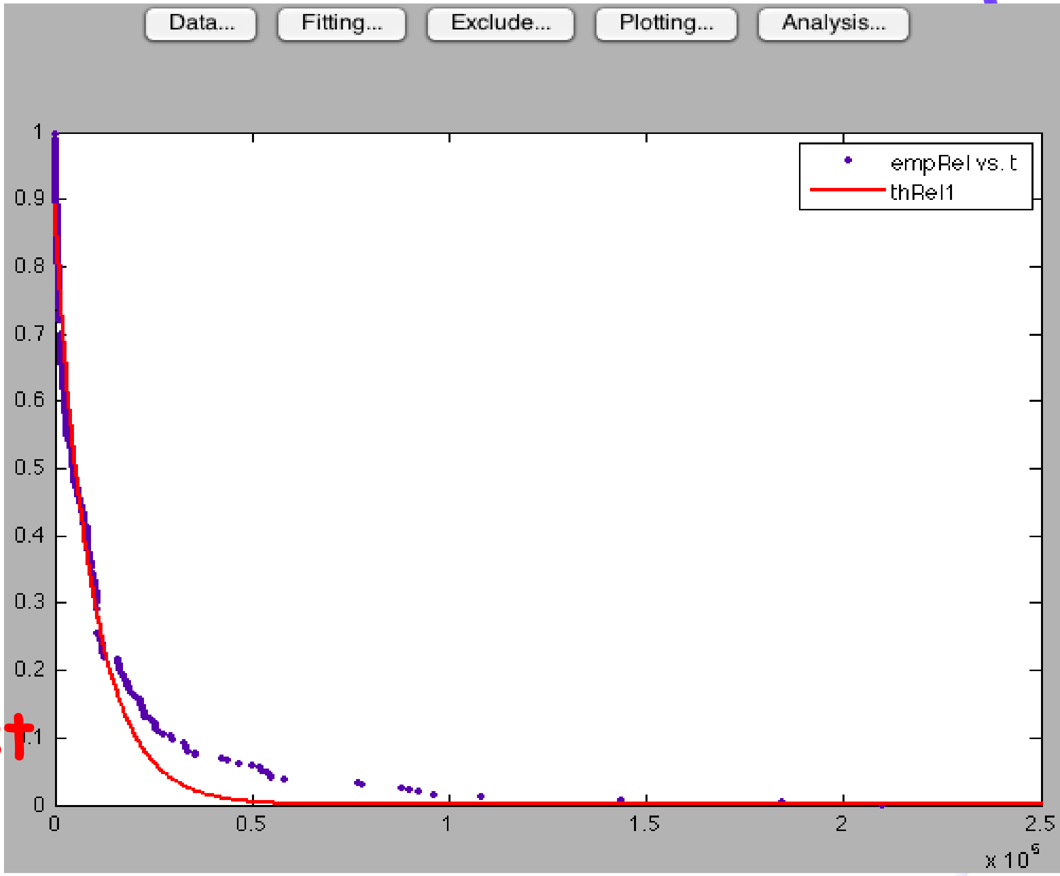
- Indicators:

  SSE: Sum of Squares due to Errors (has to be close to 0)

  R-square: "*ability*" of the fit to explain the variation in the data

thRel1: SSE=0.6735

R-square=0.9688



- **Kolmogorov-Smirnov test**

Analysis at *node/* *error-category* level

Overview of the log

- The script logStatistics.sh provides an overview of the data in the log (useful to figure out entries-prone categories and/or nodes)

```
$ ./logStatistics.sh MercuryErrorLogTEST.txt

== Total error entries ==
10000

== Breakup by CATEGORY ==
DEV 5576
PRO 1292
MEM 1198
I-O 1195
NET 713
OTH 26

== Breakup by NODE* ==
tg-c572 4030
tg-c238 1272
tg-master 817
tg-c242 622
tg-c648 616
tg-c550 253
tg-c894 217
tg-c284 180
tg-c451 159
tg-s176 156
 * only the 10 most occurring nodes are reported
```

- distribution of the entries by error category
- nodes with the largest number of entries

Filtering the log

- The script filter.sh is then used to extract entries (by node or category) from the log.

  Output is saved in a file named
  <filterName>-<systemLog>.txt

logFile filterName

```
$./filter.sh MercuryErrorLogTEST.txt tg-master

Filtering by NODE: tg-master ... [DONE]

$./filter.sh MercuryErrorLogTEST.txt DEV

Filtering by CATEGORY/CARD: DEV ... [DONE]
```

- *Filtering can be done either by node or by category*

- Filtering can be **sequential**: e.g., all MEM errors of tg-c572

  $./filter.sh MercuryErrorLog.txt tg-c572
  $./filter.sh tg-c572-MercuryErrorLog.txt MEM
  result is MEM-tg-c572-MercuryErrorLog.txt

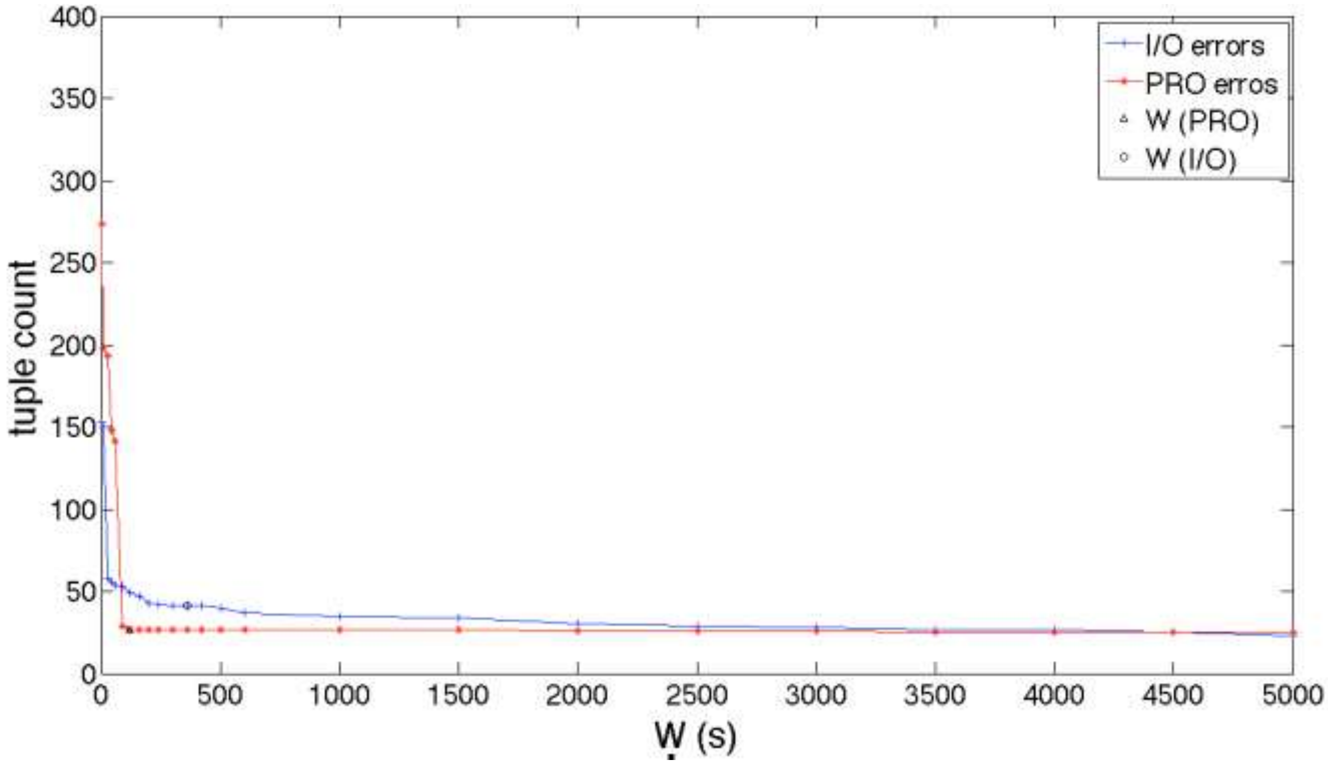
Error distribution by node

- logStatistics.sh applied to a single node returns the most-occurring categories for that node

```
./logStatistics.sh tg-master-MercuryErrorLogTEST.txt

== Total error entries ==
817

== Breakup by CATEGORY ==
NET 663
I-O 152
OTH 1
DEV 1

== Breakup by NODE* ==
tg-master 817
  * only the 10 most occurring nodes are reported
```

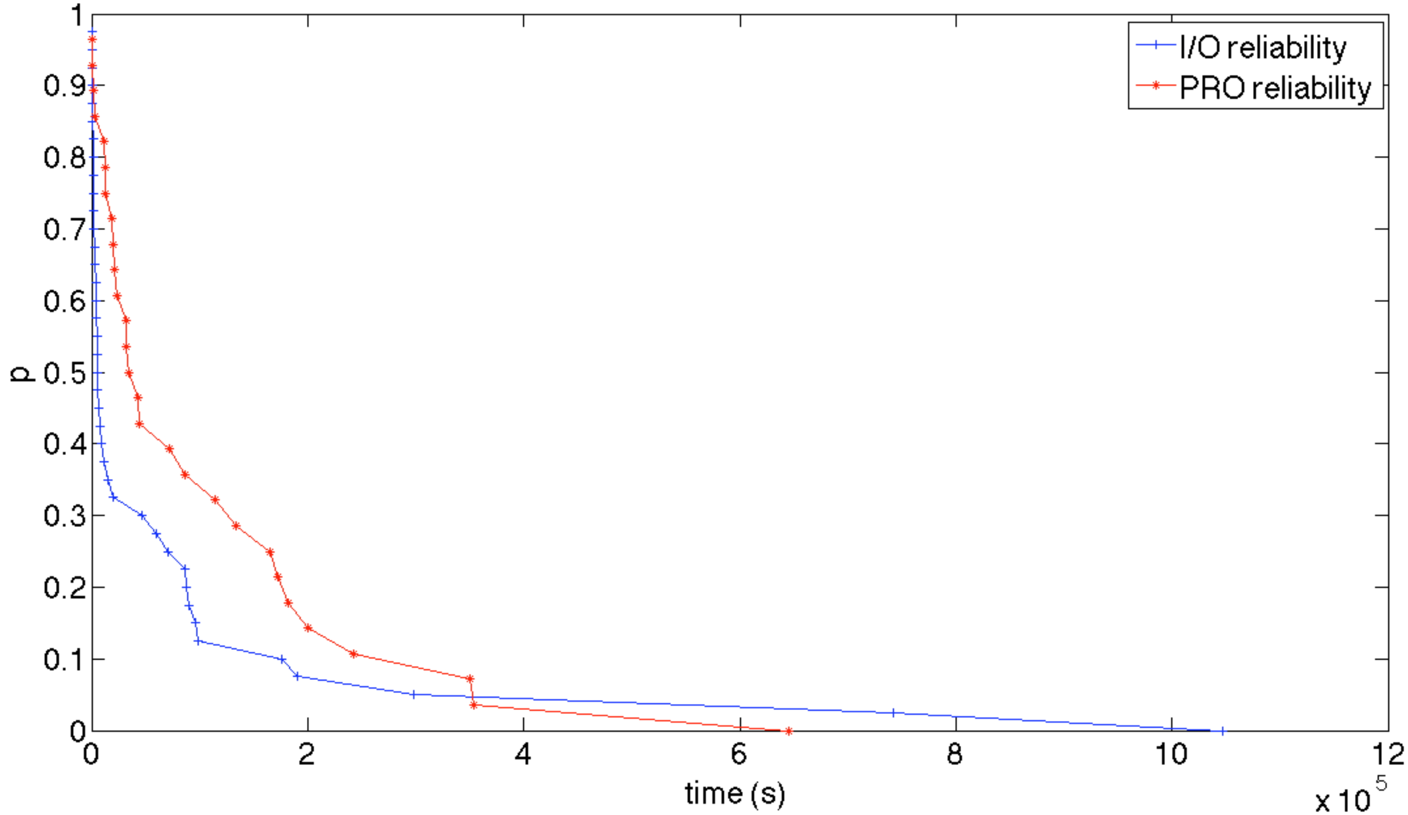
Example: error categories

```
$./filter.sh MercuryErrorLogTEST.txt PRO
$./filter.sh MercuryErrorLogTEST.txt I-O
$./tupleCount_func_CWIN.sh PRO-MercuryErrorLogTEST.txt tentative-Cwin.txt
$./tupleCount_func_CWIN.sh I-O-MercuryErrorLogTEST.txt tentative-Cwin.txt
```



Example: error categories

```
$./tupling_with_Cwin.sh I-O-MercuryErrorLogTEST.txt 360
$./tupling_with_Cwin.sh PRO-MercuryErrorLogTEST.txt 90
```



Exercises

- Repeat the analysis for **MercuryErrorLog.txt** and **BGLErrorLog.txt**:
  - tuple count plot + selected CWIN;
  - grouping the entries with CWIN;
  - empirical distributions;
  - fit the empirical reliability (exponential model + other tentative distributions, e.g., **weibull**);
  - analysis of the fit via the K-S test;

COMPARE RESULTS ACROSS THE SYSTEMS!

Exercises

- Conduct the analysis to answer the following questions:
  - can the same coalesce windows be used across different nodes (Mercury, BG/L) and error categories (Mercury) ?
  - what about the relationship between the system reliability and nodes reliability?
  - there exist dependability-bottlenecks (i.e., top-contributors to the total number of failures)?
  - do similar functional nodes (e.g., two Mercury tg-cX computing nodes or BG/L I/O nodes Ri:Mx:Nz) exhibit similar reliability parameters?
  - there exist a relationship between the error types and the node (Mercury)?
  - Whatever additional considerations and results!