

# Impact of Fault-Tolerant policies: Feasibility Study

Geff Vallee

Anand Tikotekar

Stephen L Scott

Oak Ridge National Laboratory

Chokchai Leangsuksun

Louisiana Tech University

# Outline

- Introduction & Scope
- Problem
- Fault-Tolerant (FT) policies
- Existing work
- Simulation
- Feasibility of FT policies
- Summary and Conclusion

# Introduction & Scope

- Large-scale long running parallel MPI applications
- Failures resulting in application outages
- Loss of work time from application outages is very high
  - $(T_{app} > MTBF_{WorkNodes})$
  - Some applications wont even complete
- Fault-Tolerance (FT) is costly
  - But not as costly as the loss of work time.

# Problem

- Impact of failures on application execution
  - Shorter MTBF
  - Longer applications on larger nodes
- Fault-Tolerance (FT) is getting costlier
  - More frequent use
  - Large number of nodes
- Fitting FT policies to keep overhead within a given range
  - Overhead to FT policy and feasibility
- Impact of a given FT policy on an application
  - FT policy to Overhead

# Fault-Tolerant policies

- Reactive FT
  - Checkpoint/Restart
- Proactive FT
  - Failure prediction
  - Migration
- Reactive + proactive FT
  - Checkpoint/Restart
  - Failure prediction and Migration
- No FT

# FT policies -- Reactive

- Checkpoint cost  $C_c = n_c * c_c$
- Restart cost  $C_r = n_f * c_r$
- Total cost =  $C_{app} = C_{noappfailure} + C_{appfailure}$
- Checkpoint periodicity
  - Checkpoint period: checkpoint interval + Per checkpoint cost  $C_{period} = C_{interval} + c_c$
- Rollback time
  - Rollback time = Work time lost + checkpoint cost to the application  $C_{roll} = C_{wrklost} + C_{appfailure}$

# FT policies -- Proactive

- Failure prediction

- Predictable failures  $(N_p)$

- Unpredictable failures  $(N_{\neg p})$

- Migration cost

- Migration cost to the application  $C_{migapp} = C_m * (N_p + N_{fa})$

- Rollback time

- Rollback time = Work time lost

- $$C_{roll} = \sum_{i=1}^{N_{\neg p}} t_i$$

# FT policies – Proactive + Reactive

- Combining proactive and reactive
- Combining failure prediction and checkpoint/restart
- Targeted at reducing loss of work time by predicting as well as checkpointing
- FT overhead is essentially:
  - Checkpoint/restart
  - Migration

# Existing work

- Existing work is more tuned towards the FT frameworks for a particular policy
  - Proactive frameworks
    - H/W, software failure prediction : Job Pause, VM migration, software rejuvenation
  - Reactive frameworks
    - FT MPI, FT schedulers, checkpoint/restart, Deja vu
  - Hybrid frameworks
    - FENCE
- our work is more tuned towards understanding FT policies themselves and their feasibility

# Simulation

- Simulator for FT policies
  - Developed to study impacts of various FT policies using given parameters
- Fitting FT policies for feasibility given the acceptable overhead
  - Simulator is not tuned for reverse engineering
  - Users only care about the acceptable overhead
  - Yet, simulator is used to test the feasibility of a specific instance of a policy
  - If any FT policies are feasible
    - Parameter bounds for a given overhead



# Impact of failures on reactive policy

Number of failures	10% Acceptable overhead	(Cc , Cr) R = 1/10	(Cc , Cr) R = 1/10
2	12	(5.45, 0.54)	(3, 3)
3	24	(7.27, 0.72)	(4, 4)
5	36	(6.55, 0.65)	(3.6, 3.6)
12	48	(3.64, 0.36)	(2, 2)
12	60	(4.56, 0.45)	(2.5, 2.5)
28	72	(2.34, 0.23)	(1.29, 1.29)
41	84	(1.86, 0.86)	(1.02, 1.02)
61	96	(0.69, 0.07)	(0.75, 0.75)
68	108	(1.44, 0.14)	(0.79, 0.79)
80	120	(1.36, 0.14)	(0.75, 0.75)
117	132	(1.03, 0.10)	(0.56, 0.56)
117	144	(1.19, 0.12)	(0.62, 0.62)

Parameters shown using two R values

Checkpoints done just before the failure

$$C_{FT} = C_c + C_r + C_{wrklost} \quad C_{FT} = n_c * c_c + n_f * c_r + \sum_{i=1}^{n_f} t_i$$

$$c_c = C_{FT} / n_f * (1 + R)$$

# Impact of failures on proactive policy

Number of failures	10% Acceptable overhead	Cost Breakdown
2	12	$2 * c_m + t_1$
3	24	$3 * c_m + t_1$
5	36	$5 * c_m + t_1$
12	48	$12 * c_m + \sum_{i=1}^4 t_i$
12	60	$12 * c_m + \sum_{i=1}^4 t_i$
28	72	$28 * c_m + \sum_{i=1}^8 t_i$
41	84	$41 * c_m + \sum_{i=1}^{12} t_i$
61	96	$61 * c_m + \sum_{i=1}^{18} t_i$
68	108	$68 * c_m + \sum_{i=1}^{20} t_i$
80	120	$80 * c_m + \sum_{i=1}^{24} t_i$
117	132	$117 * c_m + \sum_{i=1}^{35} t_i$
117	144	$117 * c_m + \sum_{i=1}^{35} t_i$

$$C_{FT} = (n_p + n_{fa}) * c_m + \sum_{i=1}^{n_{\neg p}} t_i$$

$$\text{prediction accuracy} = n_p / (n_p + n_{\neg p})$$

$$\text{false alarm rate} = n_{fa} / (n_p + n_{fa})$$

$$c_m \ll \text{avg}(t_i)$$

Prediction accuracy = 70%

False alarm rate = 30%

Analyzing  $\sum_{i=1}^k t_i$  Depends on

Failure distribution

# Impact of failures on proactive + Reactive policy

Number of failures	10% Acceptable overhead	Cost Breakdown
2	12	$2*c_m + c_c + c_r$
3	24	$3*c_m + c_c + c_r$
5	36	$5*c_m + c_c + c_r$
12	48	$12*c_m + 4*(c_c + c_r)$
12	60	$12*c_m + 4*(c_c + c_r)$
28	72	$28*c_m + 8*(c_c + c_r)$
41	84	$41*c_m + 12*(c_c + c_r)$
61	96	$61*c_m + 18*(c_c + c_r)$
68	108	$68*c_m + 20*(c_c + c_r)$
80	120	$80*c_m + 24*(c_c + c_r)$
117	132	$117*c_m + 35*(c_c + c_r)$
117	144	$117*c_m + 35*(c_c + c_r)$

$$C_{FT} = (n_p + n_{fa}) * c_m + (c_c + c_r) * n_{\neg p}$$

This table shows the best case

- Checkpoints are done just before the failures

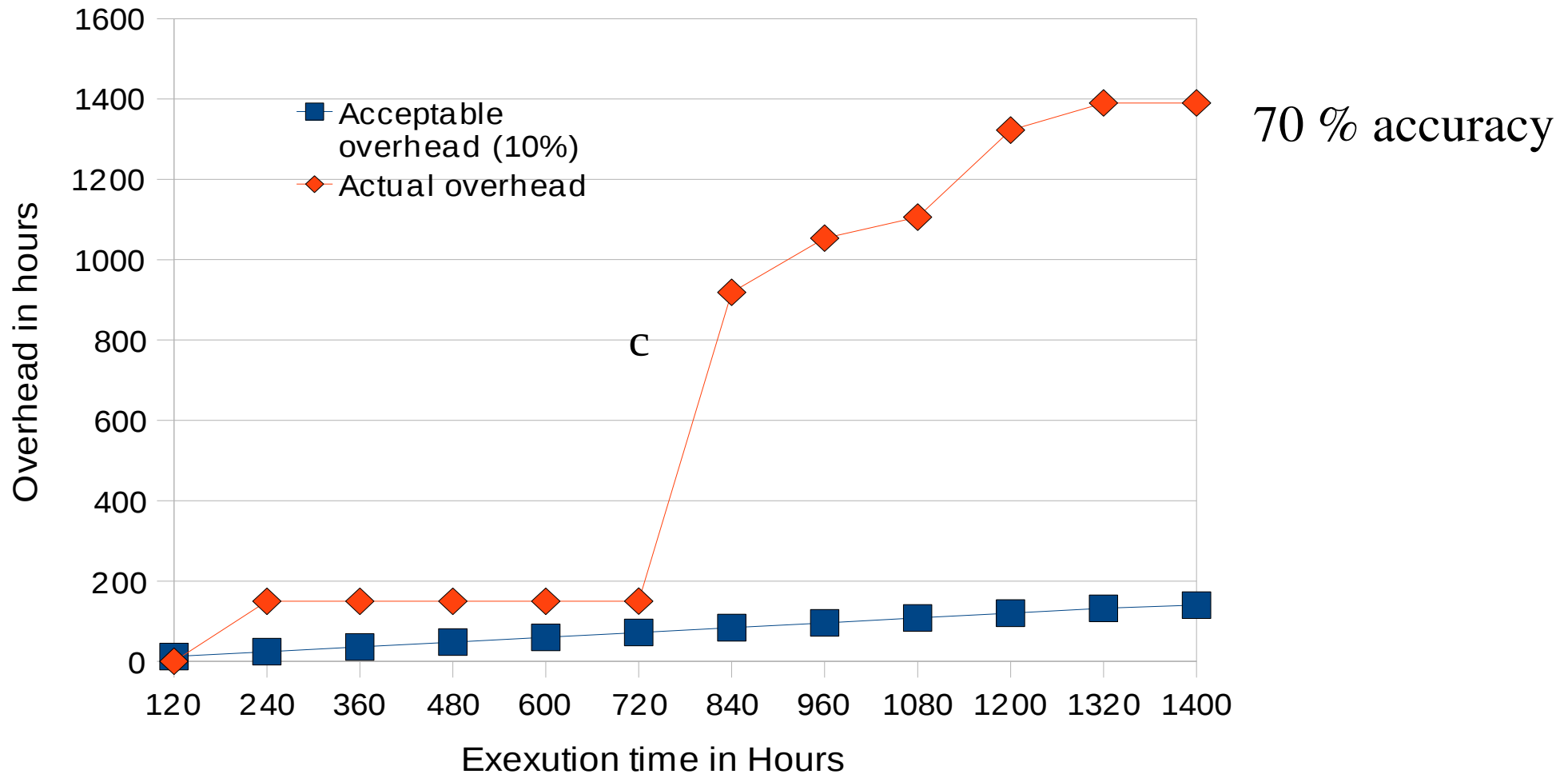
70 % accuracy

30 % false alarm rate

Migration costs are very small : Based on Xen's Live migration

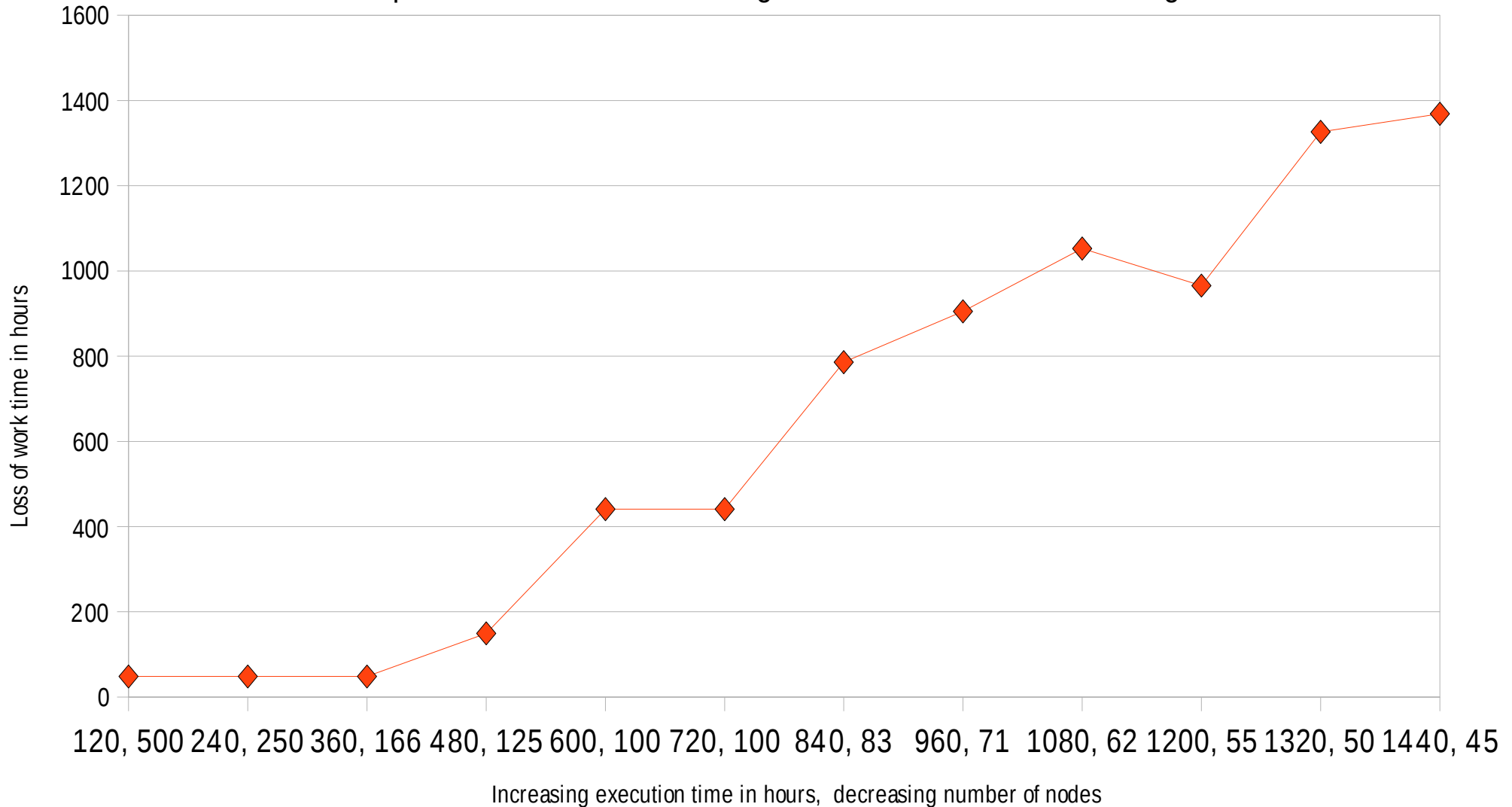
# Proactive policy: Simulation

## Impact of failures: proactive policy



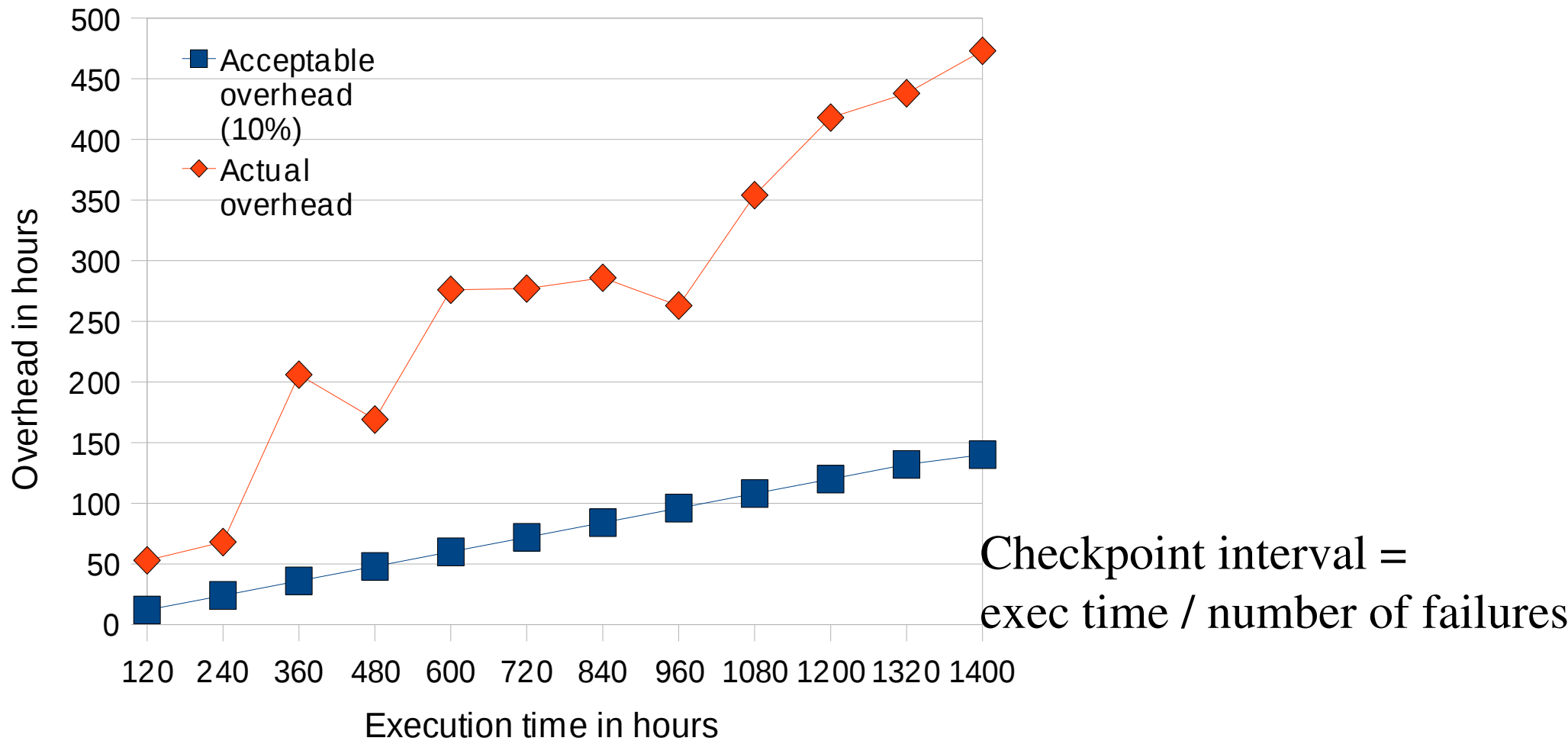
# Cost of increasing parallelism

Impact of failures on increasing execution time and decreasing nodes



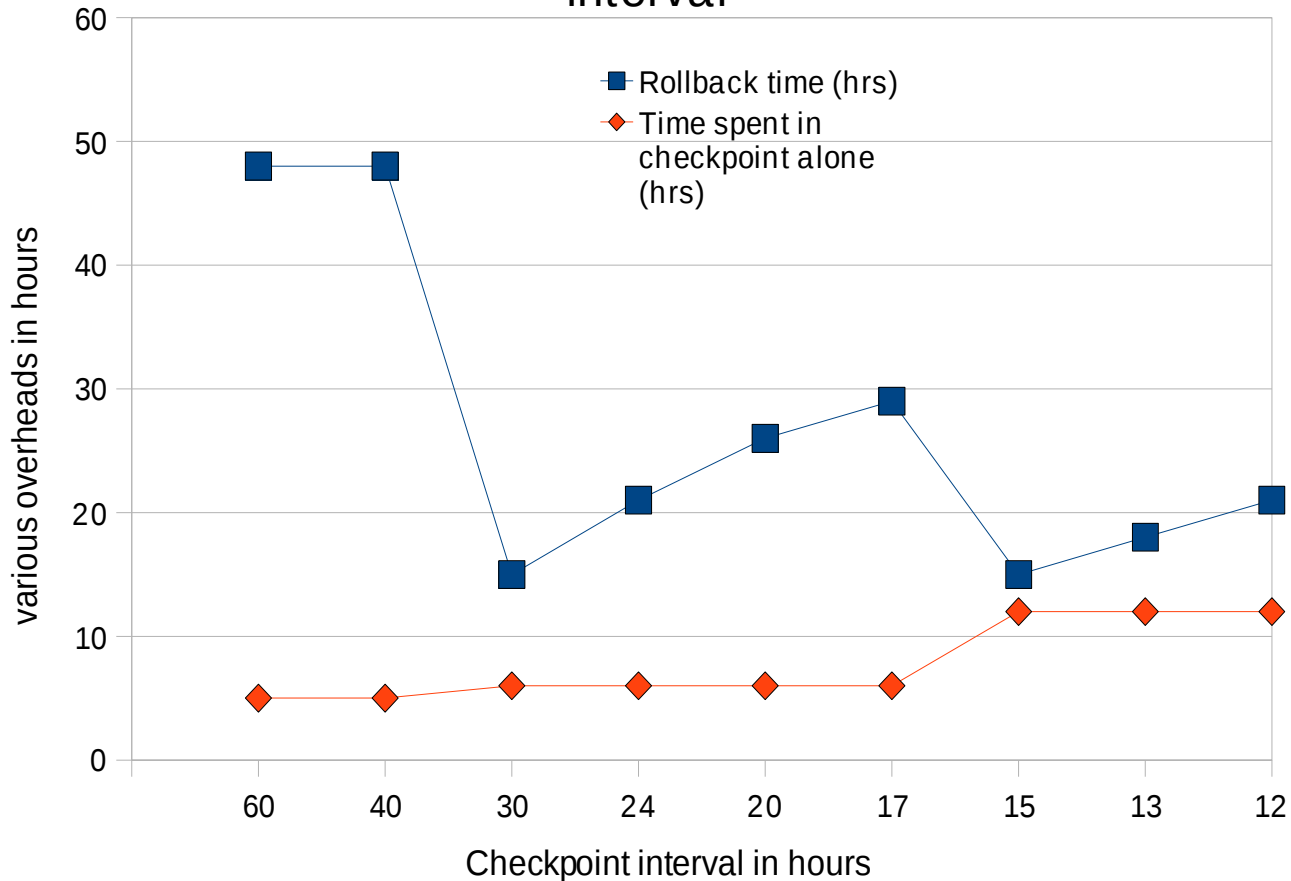
# Reactive policy: Simulation

Impact of failures: Reactive policy



# Relationship between various overheads

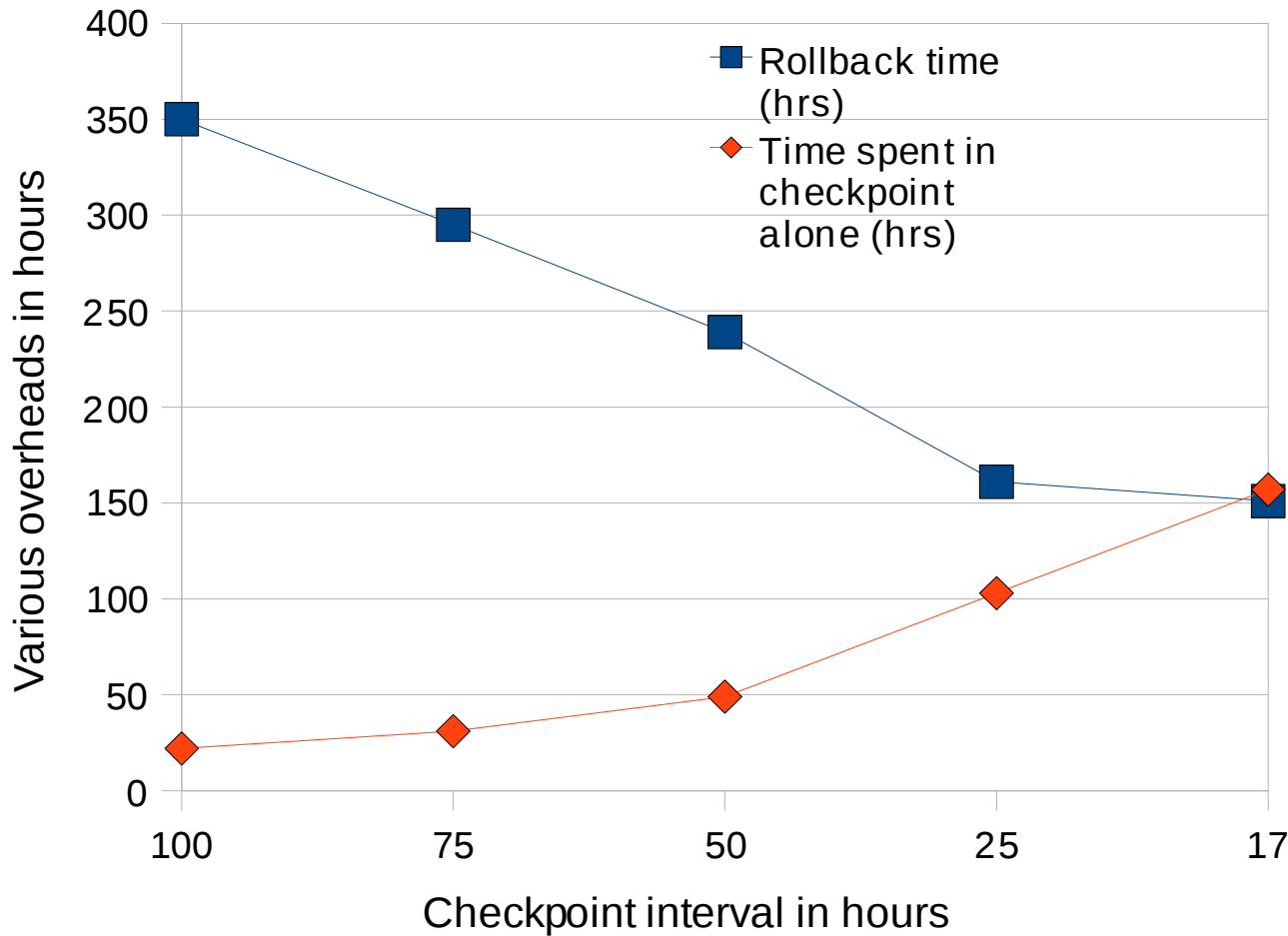
Rollback time and checkpoint cost Vs checkpoint interval



execution time = 120 hrs  
Number of failures = 2

# Relationship between various overheads

## Relationship between various overheads

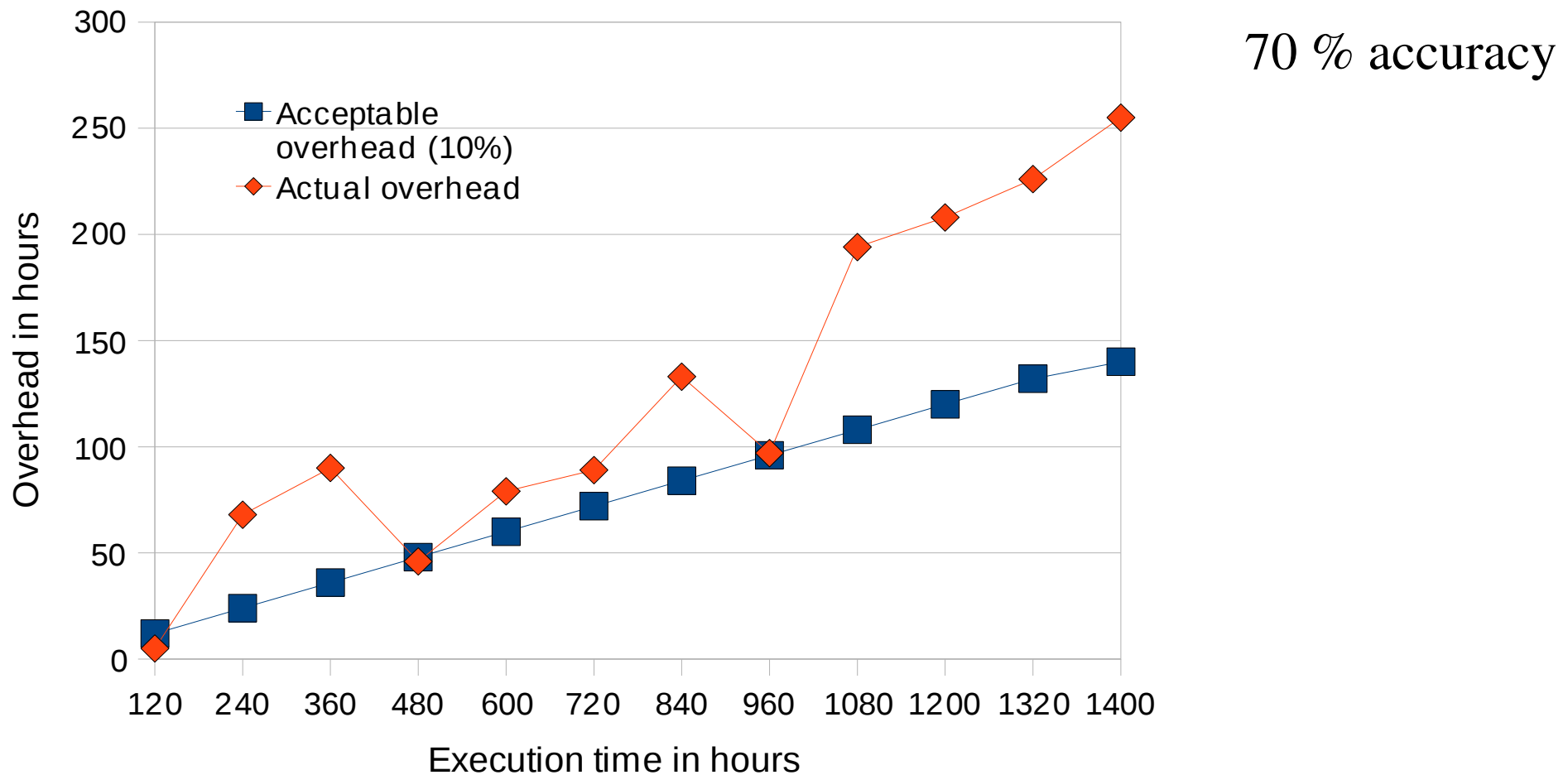


execution time = 600 hrs

Number of failures = 12

# Proactive combined with reactive : Simulation

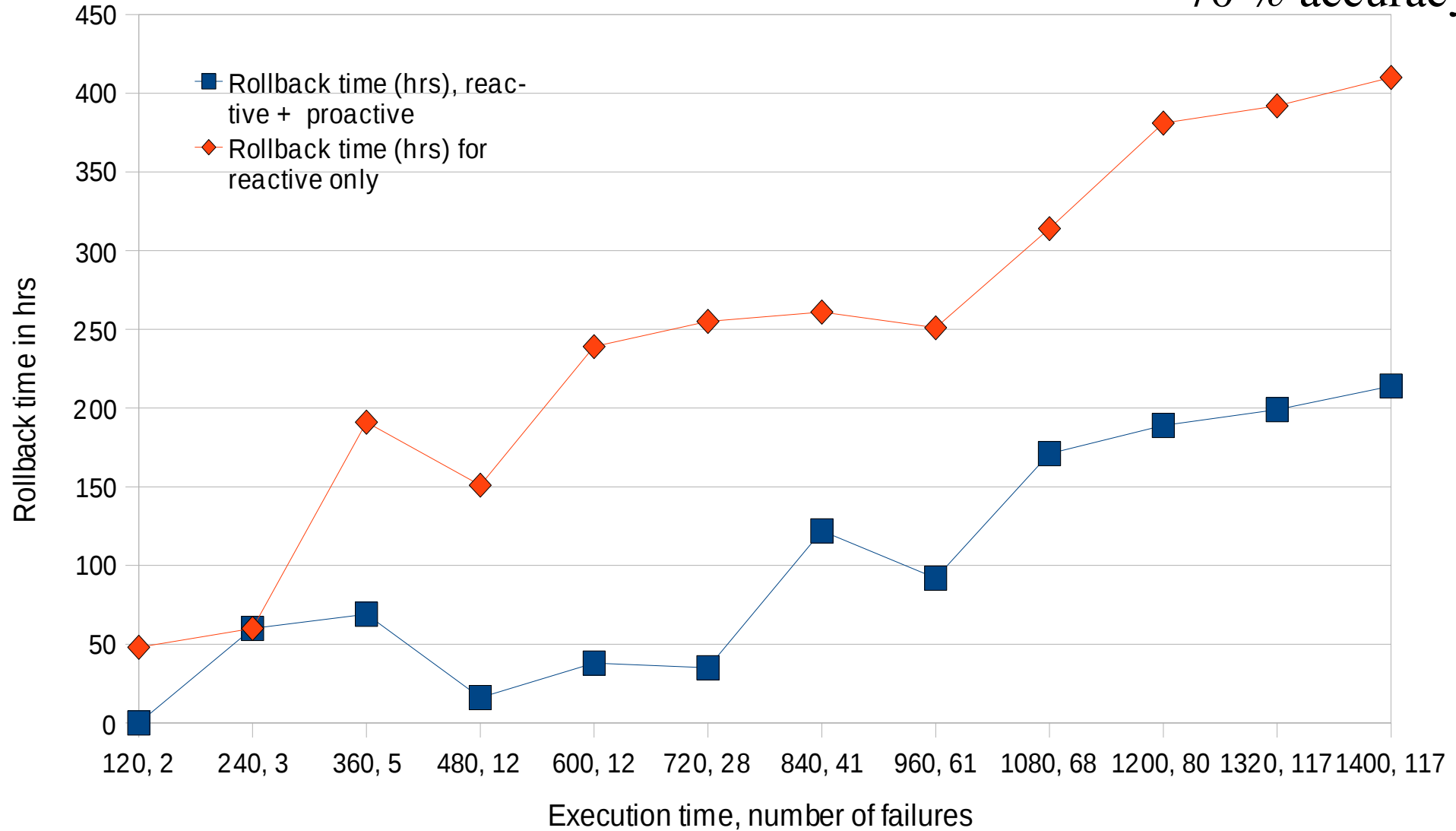
Impact of failures: Proactive combined with Reactive



# Rollback time

## Rollback time for FT policies

70 % accuracy



# Conclusion

- We have exposed the problem of fitting an FT policy within the acceptable overhead
- Simulator is a good approach to test a particular fit for a given policy
- We have tested the failure logs using our simulator and compared with the acceptable overhead
- It is difficult to determine the parameters for FT policies without the analysis of failure distribution
- 10% overhead is too optimistic for our test case