

Impact of Fault-Tolerance Policies: Feasibility study¹

Geoffroy Vallée

Oak Ridge National Laboratory
Oak Ridge, Tennessee 37830
Email: vallee@ornl.gov

Anand Tikotekar

Oak Ridge National Laboratory
Oak Ridge, Tennessee 37830
Email: tikotekaraa@ornl.gov

Chokchai Leangsuksun

Louisiana Tech University
Ruston, LA 71272, USA
Email: box@coes.latech.edu

Stephen L. Scott

Oak Ridge National Laboratory
Oak Ridge, Tennessee 37830
Email: scottsl@ornl.gov

Abstract—Large-scale systems, such as petascale, often have very low Mean Time Between Failures (MTBF). Such a scenario implies that long-running parallel applications would never finish. Therefore, Fault Tolerance (FT) has become indispensable now than it was ever before. Today several FT solutions are available, from reactive-only policies to proactive-only policies. However, because the impact of each policies depends on the execution platform characteristics (number of failures, failure distribution and so on) and on the characteristics of the fault tolerance mechanisms involved (typically their overhead) it is very difficult for a user to know which policy is adapted to his needs. This is especially critical when FT mechanisms increase the total execution time to the point where more failures occur during the total application execution time, creating a risk of domino effect.

In this document we study the impact of various fault tolerance policies on the execution of long-running applications. This study is based on system logs from the LLNL's ASCI white system. We show that different configuration leads to different needs in term of fault tolerance.

I. INTRODUCTION

Fault tolerance (FT) is a well-known issue for distributed systems and tends to become critical as the size of such systems increase quickly over time: larger the system is, more failures occur. Today, many different solutions for FT are available, from reactive-only FT (typically using checkpoint/restart mechanisms) to proactive-only FT [1], [2] (typically based on migration mechanisms).

But the interest for application users is the following: for a given platform and a given application, what is the best FT policy?

To address this question, a previous effort was the development of a simulator [3], based on real system logs, which simulates the execution of a parallel application on a 512 nodes cluster, using different FT mechanisms and policies (it is possible to change many parameters, for instance the checkpoint/restart cost). Another previous effort about the analysis of the failure distribution for this platform [4] also shows that the best-fit model for such a system is the Weibull distribution, but unfortunately, this study, like the simulator does not address questions from the user's perspective: if you have an application and if you are ready to pay a 10% overhead for FT, is the system suitable?

To answer this question, few aspects have to be included in our study: (i) the failure distribution, typically the number of

failures during a time slot depends on when the time slot is attributed during the platform life time; (ii) for a given time slot, if the user accepts to pay only a 10% overhead, what does it mean in terms of maximum FT mechanisms cost? Therefore, we use the system logs on which our simulator is based to estimate the number of failures for a given time slot, allocated at a given time on the execution platform. Then, for different FT policies, we evaluate the maximum cost of the FT mechanisms we can afford, and finally analyze which cases are doable.

For simplification, we select the following context: the user accepts to pay a maximum of 10% FT overhead; the application uses 512 nodes; we compare 3 different FT policies, a reactive-only, a proactive-only, and an hybrid policy (which combines both the reactive and the proactive approach). In this document our goal is not to give an exhaustive analysis of all the possible configurations, we select few cases and analyze them based on these selected parameters.

Before detailing on the study, we introduce the notation we use across the document:

- T_A is the theoretical application execution time on the platform,
- c_c is the a single checkpoint cost (assumed constant),
- c_r is the a single restart cost (assumed constant),
- C_c is the total checkpoint cost over a period of time,
- C_r is the total restart cost over a period of time,
- $C_{rollback}$ is the total rollback cost over all the restarts
- n_f is the number of failures,
- n_c is the number of checkpoints,
- n_r is the number of restarts,
- n_p is the number of predictable failures,
- n_{-p} is the number of unpredictable failures,
- n_{fa} is the number of false alarms.

The remainder of this paper is organized as follows: Section II presents an analysis of the failure rate from the system logs for different application (in term of execution time) and different time slot allocations; Section III presents an analysis of the impact of such failure rates when using a standard reactive-only FT policy; Section IV presents an analysis of the impact of such failure rates when using a standard proactive-only FT policy; Section V presents an analysis of the impact of such failure rates when using an FT policy combining both

TABLE I
NUMBER OF FAILURES FOR DIFFERENT APPLICATION EXECUTION TIME,
STARTING AT t_0 OF THE EXECUTION PLATFORM LIFETIME

Theoretical application execution time (in hours)	Theoretical number of failures	Actual number of number of failure during the application execution time + 10% FT overhead	Number of failures introduced
120	2	2	0
240	3	3	0
360	5	5	0
480	8	12	4
600	12	12	0
720	17	28	11
840	31	41	10
960	52	61	9
1080	62	68	6
1200	69	80	11
1320	80	117	37
1440	117	117	0

TABLE II
NUMBER OF FAILURES FOR DIFFERENT APPLICATION EXECUTION TIMES,
STARTING AT $t = 192hours$ OF THE PLATFORM LIFETIME

Theoretical application execution time (in hours)	Theoretical number of failures	Actual number of number of failure during the application execution time + 10% FT overhead	Number of failures introduced
120	1	1	0
240	3	4	1
360	9	9	9
480	9	14	5
600	25	28	3
720	38	55	17
840	55	65	10
960	65	68	3
1080	70	113	33
1200	114	114	0
1320	114	114	0
1440	114	114	0

reactive and proactive FT; and finally Section VI concludes.

II. FAILURE RATE ESTIMATION

The first difficulty to perform this kind of study is to evaluate the number of failures when running a given application during a given time slot on the execution platform. Because failures are not linearly distributed over time, we cannot assume that the failure rate remains the same over time.

Tables I, II, III, and IV show the number of failures for different time slots. These tables present two different cases: the theoretical number of failures during the theoretical application execution time (we do not include any FT overhead; we get these numbers directly from the logs) and the number of failures when maximizing the application execution time by 10% (the overhead the user accept to pay for FT). We can therefore see the number of failures introduced when extending the application execution time by using FT mechanisms. The FT overhead has to be less than 10% of the application execution time. It means that for total cost for fault tolerance has to be equal or inferior to this 10% of the application execution time:

$$total\ FT\ cost < \frac{1}{10} * T_A$$

III. REACTIVE-ONLY FAULT TOLERANCE

With a reactive fault tolerance, the FT overhead is composed of two parts: (i) the checkpoint overhead, and (ii) the restart overhead. Please note that we do not consider queue wait as part of the restart cost.

$$total\ FT\ cost = C_c + C_r + C_{rollback} \quad (1)$$

Assuming that $n_c = k * n_f$

TABLE III

NUMBER OF FAILURES FOR DIFFERENT APPLICATION EXECUTION TIMES,
STARTING AT $t = 504hours$ OF THE PLATFORM LIFETIME

Theoretical application execution time (in hours)	Theoretical number of failures	Actual number of number of failure during the application execution time + 10% FT overhead	Number of failures introduced
120	0	0	0
240	9	10	1
360	19	26	7
480	46	46	0
600	56	56	0
720	58	63	5
840	76	105	29
960	105	105	0
1080	105	105	0
1200	105	105	0
1320	105	105	0
1440	105	105	0

$$total\ FT\ cost = k * n_f * c_c + n_f * c_r + \sum_{i=1}^{n_f} t_i \quad (2)$$

To evaluate the actual cost of the checkpoint/restart mechanisms, we take the best case: the checkpoints are done right before the failures, in other words $\sum_{i=1}^{n_f} t_i = 0$, and $n_f = n_c$ and $k = 1$. Therefore, we have $total\ FT\ cost = n_f * c_c + n_f * c_r$. Then if we assume that the restart cost is proportional to the checkpoint cost: $c_r = R * c_c$, and based on this assumption, we have $c_c = \frac{(total\ FT\ cost)}{n_f * (1+R)}$. Tables V, VI, VII, VIII show the maximum cost for the checkpoint/restart in two cases: (i) $R = 10$, i.e., the restart cost 10 times more expensive than

TABLE IV

NUMBER OF FAILURES FOR DIFFERENT APPLICATION EXECUTION TIMES, STARTING AT $t = 660$ hours OF THE PLATFORM LIFETIME

Theoretical application execution time (in hours)	Theoretical number of failures	Actual number of number of failure during the application execution time + 10% FT overhead	Number of failures introduced
120	11	16	5
240	26	29	3
360	46	49	3
480	56	56	0
600	59	68	9
720	104	105	1
840	105	105	0
960	105	105	0
1080	105	105	0
1200	105	105	0
1320	105	105	0
1440	105	105	0

TABLE V

NUMBER OF FAILURES FOR DIFFERENT APPLICATION EXECUTION TIMES W/ A REACTIVE-ONLY POLICY, STARTING AT t_0 OF THE EXECUTION PLATFORM LIFETIME

Estimated number of failures	Maximum FT overhead in hours (10% FT overhead)	(c_c, c_r) with $R = 1/10$	(c_c, c_r) with $R = 1$
2	12	(5.45, 0.54)	(3, 3)
3	24	(7.27, 0.72)	(4, 4)
5	36	(6.55, 0.66)	(3.6, 3.6)
12	48	(3.64, 0.36)	(2, 2)
12	60	(4.56, 0.46)	(2.5, 2.5)
28	72	(2.34, 0.23)	(1.29, 1.29)
41	84	(1.86, 0.19)	(1.02, 10.2)
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)

the checkpoint cost (note the results are the opposite if we take $R = 1/10$, and (ii) $R = 1$, *i.e.*, the checkpoint and restart costs are the same. The two values of R have been chosen to give us an idea of the different possibilities based on different checkpoint/restart mechanisms. For instance, Table V shows that checkpoint/restart seems to be an acceptable solution for almost all the simulated applications, whenever the application is executed and whatever the execution time is. Only few cases seem to be difficult to implement: when we execute a 960 hours application on all the nodes at the beginning of the platform life time, checkpoint/restart costs are (0.69, 0.07) and (0.75, 0.75) respectively for ($R = 1/10$ and $R = 1$). This means that the checkpoint/restart has to be done in a few minutes, which may be a problem for 512 nodes. Values that are susceptible to end to such issues are highlighted in the different tables (typically cases where checkpoints and/or

TABLE VI

NUMBER OF FAILURES FOR DIFFERENT APPLICATION EXECUTION TIMES W/ A REACTIVE-ONLY POLICY, STARTING AT $t = 192$ hours OF THE PLATFORM LIFETIME

Estimated number of failures	Maximum FT overhead in hours (10% FT overhead)	(c_c, c_r) with $R = 1/10$	(c_c, c_r) with $R = 1$
1	12	(10.90, 1.09)	(6, 6)
4	24	(5.45, 0.54)	(3, 3)
9	36	(3.64, 0.36)	(2, 2)
14	48	(3.12, 0.31)	(1.71, 1.71)
28	60	(1.95, 0.19)	(1.07, 1.07)
55	72	(1.19, 0.11)	(0.65, 0.65)
65	84	(1.17, 0.11)	(0.65, 0.65)
68	96	(1.28, 0.12)	(0.71, 0.71)
113	108	(0.89, 0.08)	(0.48, 0.48)
114	120	(0.96, 0.09)	(0.53, 0.53)
114	132	(1.05, 0.10)	(0.58, 0.58)
114	144	(1.15, 0.11)	(0.63, 0.63)

TABLE VII

NUMBER OF FAILURES FOR DIFFERENT APPLICATION EXECUTION TIMES W/ A REACTIVE-ONLY POLICY, STARTING AT $t = 504$ hours OF THE PLATFORM LIFETIME

Estimated number of failures	Maximum FT overhead in hours (10% FT overhead)	(c_c, c_r) with $R = 1/10$	(c_c, c_r) with $R = 1$
0	12	(∞, ∞)	(∞, ∞)
10	24	(2.18, 0.21)	(1.2, 1.2)
26	36	(1.26, 0.12)	(0.69, 0.69)
46	48	(0.95, 0.01)	(0.52, 0.52)
56	60	(0.97, 0.01)	(0.53, 0.53)
63	72	(1.04, 0.01)	(0.57, 0.57)
105	84	(0.73, 0.01)	(0.4, 0.4)
105	96	(0.83, 0.01)	(0.46, 0.46)
105	108	(0.93, 0.01)	(0.51, 0.51)
105	120	(1.04, 0.01)	(0.57, 0.57)
105	132	(1.14, 0.01)	(0.63, 0.63)
105	144	(1.25, 0.01)	(0.69, 0.69)

TABLE VIII

NUMBER OF FAILURES FOR DIFFERENT APPLICATION EXECUTION TIMES W/ A REACTIVE-ONLY POLICY, STARTING AT $t = 660$ hours OF THE PLATFORM LIFETIME

Estimated number of failures	Maximum FT overhead in hours (10% FT overhead)	(c_c, c_r) with $R = 1/10$	(c_c, c_r) with $R = 1$
16	12	(0.68, 0.07)	(0.38, 0.38)
29	24	(0.75, 0.08)	(0.41, 0.41)
49	36	(0.67, 0.07)	(0.37, 0.37)
56	48	(0.78, 0.08)	(0.42, 0.42)
68	60	(0.80, 0.08)	(0.44, 0.44)
105	72	(0.62, 0.06)	(0.34, 0.34)
105	84	(0.73, 0.07)	(0.4, 0.4)
105	96	(0.83, 0.08)	(0.45, 0.45)
105	108	(0.94, 0.09)	(0.51, 0.51)
105	120	(1.04, 0.10)	(0.57, 0.57)
105	132	(1.14, 0.11)	(0.63, 0.63)
105	144	(1.25, 0.13)	(0.69, 0.69)

restarts need to be done in few minutes).

Also note that if long-running applications are executed while the platform already accumulated execution time, it seems very difficult to use checkpoint/restart especially when the checkpoint/restart costs are not similar (*e.g.*, $R = 1/10$). Also note that when the checkpoint and restart costs are the same, based on our tables, checkpoint/restart seems a doable solution; however, in practice we already now this is rarely the case since the checkpoint/restart costs vary in function of the policy involved (typically a synchronization has to be done during checkpoints or restarts, leading to a cost different between the two mechanisms).

IV. PROACTIVE-ONLY FAULT TOLERANCE

The proactive-only fault tolerance policy tries to migrate the running application away from faulty nodes when failures are predicted. Of course, not all failures can be predicted and some failure predictions are actually false predictions (false alarms). It means that everytime an unpredicted failure occurs, the application has to be restarted from scratch. It also means that false alarms create an unnecessary overhead since the application is migrated away from a healthy node.

Now if we assume the number of unpredicted failures and false alarms are proportional to the total number of failure, we have $n_p = P * n_f$, $n_{-p} = P' * n_f$, and $n_{fa} = F_A * n_p$.

For simplification we assume a spare node is always available for migration.

In this context, Equation 3 shows the cost of the proactive-only FT policy, where t_i is the time wasted before the unpredicted failure i .

$$total\ FT\ cost = (n_p + n_{fa}) * c_m + \sum_{i=1}^{n-p} t_i \quad (3)$$

Moreover the prediction accuracy is calculated as follow:

$$Prediction\ Accuracy = \frac{n_p}{n_p + n_{-p}}$$

The false alarm rate is calculated as follow:

$$False\ alarm\ rate = \frac{n_{fa}}{n_p + n_{fa}}$$

In this case we select the following parameters: the prediction accuracy is 70%, and the false alarm rate is 30%. Please note that, we picked these parameters as they seem realistic based on our knowledge of various proactive frameworks. However, it should also be noted that the following analysis can be performed using other parameters. Based on these choices, Table IX presents the number of predicted, unpredicted and false alarms for the different cases. The table also presents the number of migration and checkpoint/restart that implies.

Tables X, XI, XII, and XIII presents the repartition of the FT in the different cases selected for this study. Represented data does not really allow us to conclude: the application has to restart from scratch everytime an unpredicted failure occurs. All the time spent to execute the application between failures is therefore lost. In order to conclude, our first step

TABLE IX
FAILURE REPARTITION BASED ON PREDICTION ACCURACY AND FALSE ALARM RATE FOR THE PROACTIVE-ONLY POLICY

Estimated number of failures	Number of predicted failures	Number of unpredicted failures	Number of false alarms	FT overhead
1	1	0	0	c_m
2	1	1	1	$2 * c_m + t_1$
3	2	1	1	$3 * c_m + t_1$
4	3	1	1	$4 * c_m + t_1$
5	4	1	1	$5 * c_m + t_1$
9	6	3	3	$9 * c_m + \sum_{i=3}^3 t_i$
10	7	3	3	$10 * c_m + \sum_{i=1}^4 t_i$
12	8	4	4	$12 * c_m + \sum_{i=1}^4 t_i$
14	10	4	4	$14 * c_m + \sum_{i=1}^5 t_i$
16	11	5	5	$16 * c_m + \sum_{i=1}^8 t_i$
26	18	8	8	$26 * c_m + \sum_{i=1}^8 t_i$
28	20	8	8	$28 * c_m + \sum_{i=1}^9 t_i$
29	20	9	9	$29 * c_m + \sum_{i=2}^9 t_i$
41	29	12	12	$41 * c_m + \sum_{i=1}^9 t_i$
46	32	14	14	$46 * c_m + \sum_{i=1}^9 t_i$
49	34	15	15	$49 * c_m + \sum_{i=1}^9 t_i$
55	39	16	16	$55 * c_m + \sum_{i=1}^9 t_i$
56	39	17	17	$56 * c_m + \sum_{i=1}^9 t_i$
61	42	18	18	$60 * c_m + \sum_{i=1}^9 t_i$
63	44	19	19	$63 * c_m + \sum_{i=1}^9 t_i$
65	46	19	19	$65 * c_m + \sum_{i=1}^9 t_i$
68	48	20	20	$68 * c_m + \sum_{i=1}^9 t_i$
80	56	24	24	$80 * c_m + \sum_{i=1}^9 t_i$
105	74	31	31	$105 * c_m + \sum_{i=1}^9 t_i$
113	79	34	34	$113 * c_m + \sum_{i=1}^9 t_i$
114	80	34	34	$114 * c_m + \sum_{i=1}^9 t_i$
117	82	35	35	$117 * c_m + \sum_{i=1}^9 t_i$

TABLE X
ANALYSIS OF THE FT OVERHEAD W/ A PROACTIVE POLICY, STARTING AT t_0 OF THE EXECUTION PLATFORM LIFETIME

FT overhead (in hours)	FT cost
12	$2 * c_m + t_1$
24	$3 * c_m + t_1$
36	$5 * c_m + t_1$
48	$12 * c_m + \sum_{i=1}^4 t_i$
60	$12 * c_m + \sum_{i=1}^4 t_i$
72	$28 * c_m + \sum_{i=1}^8 t_i$
84	$41 * c_m + \sum_{i=1}^8 t_i$
96	$60 * c_m + \sum_{i=1}^8 t_i$
108	$68 * c_m + \sum_{i=1}^9 t_i$
120	$80 * c_m + \sum_{i=1}^9 t_i$
132	$117 * c_m + \sum_{i=1}^9 t_i$
144	$117 * c_m + \sum_{i=1}^9 t_i$

TABLE XI
ANALYSIS OF THE FT OVERHEAD W/ A PROACTIVE POLICY, STARTING AT
 $t = 192\text{hours}$ OF THE PLATFORM LIFETIME

FT overhead (in hours)	FT cost
12	n/a
24	$10 * c_m + \sum_{i=1}^3 t_i$
36	$26 * c_m + \sum_{i=1}^4 t_i$
48	$46 * c_m + \sum_{i=1}^7 t_i$
60	$56 * c_m + \sum_{i=1}^9 t_i$
72	$63 * c_m + \sum_{i=1}^{11} t_i$
84	$105 * c_m + \sum_{i=1}^{15} t_i$
96	$105 * c_m + \sum_{i=1}^{15} t_i$
108	$105 * c_m + \sum_{i=1}^{15} t_i$
120	$105 * c_m + \sum_{i=1}^{15} t_i$
132	$105 * c_m + \sum_{i=1}^{15} t_i$
144	$105 * c_m + \sum_{i=1}^{15} t_i$

TABLE XII
ANALYSIS OF THE FT OVERHEAD W/ A PROACTIVE-ONLY POLICY,
STARTING AT $t = 504\text{hours}$ OF THE PLATFORM LIFETIME

FT overhead (in hours)	FT cost
12	c_m
24	$4 * c_m + t_1$
36	$9 * c_m + \sum_{i=1}^3 t_i$
48	$14 * c_m + \sum_{i=1}^4 t_i$
60	$28 * c_m + \sum_{i=1}^8 t_i$
72	$55 * c_m + \sum_{i=1}^{16} t_i$
84	$65 * c_m + \sum_{i=1}^{19} t_i$
96	$68 * c_m + \sum_{i=1}^{20} t_i$
108	$113 * c_m + \sum_{i=1}^{34} t_i$
120	$114 * c_m + \sum_{i=1}^{34} t_i$
132	$114 * c_m + \sum_{i=1}^{34} t_i$
144	$114 * c_m + \sum_{i=1}^{34} t_i$

TABLE XIII
ANALYSIS OF THE FT OVERHEAD W/ A PROACTIVE-ONLY POLICY,
STARTING AT $t = 660\text{hours}$ OF THE PLATFORM LIFETIME

FT overhead (in hours)	FT cost
12	$16 * c_m + 5 * \sum_{i=1}^5 t_i$
24	$29 * c_m + 9 * \sum_{i=1}^9 t_i$
36	$49 * c_m + 15 * \sum_{i=1}^{15} t_i$
48	$56 * c_m + 17 * \sum_{i=1}^{17} t_i$
60	$68 * c_m + 20 * \sum_{i=1}^{20} t_i$
72	$105 * c_m + 31 * \sum_{i=1}^{31} t_i$
84	$105 * c_m + 31 * \sum_{i=1}^{31} t_i$
96	$105 * c_m + 31 * \sum_{i=1}^{31} t_i$
108	$105 * c_m + 31 * \sum_{i=1}^{31} t_i$
120	$105 * c_m + 31 * \sum_{i=1}^{31} t_i$
132	$105 * c_m + 31 * \sum_{i=1}^{31} t_i$
144	$105 * c_m + 31 * \sum_{i=1}^{31} t_i$

should be to evaluate the failure distribution during application time in order to calculate the actual time wasted between failures and check if the application actually has the time to terminate within the accepted 10% overhead. These kind of study is particularly interesting since data in Tables X, XI, XII, and XIII tend to show that a proactive-only policy is interesting for long-running application because the proportion of restart regarding the actual time allowed for FT is bigger. We therefore have more chance to be able to restart the application within the accepted maximum overhead.

Furthermore the proactive-only policy is efficient for short running application when they are allocated at the beginning of the platform lifetime. In that case, only few failures occur and these failures are most of the time correctly predicted. Such failures can be handled by migration.

V. HYBRID FAULT TOLERANCE

The proactive-only fault tolerance policy tries to migrate the running application away from faulty nodes when failures are predicted, and restarts application from checkpoints when unpredicted failures occur. If we assume the checkpoints are done just before failures (best case), An implication of this best case is that there would be no rollback time. Equation 4 shows the cost of the hybrid FT policy,

$$\text{total FT cost} = (n_p + n_{fa}) * c_m + (c_c + c_r) * n_{-p} \quad (4)$$

Now if we assume the number of unpredicted failures and false alarms are proportional to the total number of failure, we have $n_p = P * n_f$, $n_{-p} = P' * n_f$, and $n_{fa} = F_A * n_f$. Therefore, Equation 4 becomes:

$$\text{total FT cost} = (P * n_f + F_A * n_f) * c_m + (c_c + c_r) * P' * n_f \quad (5)$$

Moreover the prediction accuracy is calculated as follow:

$$\text{Prediction Accuracy} = \frac{n_p}{n_p + n_{-p}}$$

The false alarm rate is calculated as follow:

$$\text{False alarm rate} = \frac{n_{fa}}{n_p + n_{fa}}$$

In the following, we use the same parameters for the proactive part as we used in Section IV.

- prediction accuracy = 70%,
- false alarm rate = 30%.

Based on these choices, Table XIV presents the number of predicted, unpredicted and false alarms for the different cases. The table also presents the number of migration and checkpoint/restart that implies.

Tables XV, XVI, XVII, and XVIII presents the repartition of the FT in the different cases selected for this study. In these tables, we do not try to actually give a value to the migration, checkpoint, and restart costs. These values may differ a lot depending on the platform, the FT mechanisms, and also the application. Therefore, the tables only show the FT actions that need to be done during the maximum period of time tolerated for FT (10% of the total execution time).

TABLE XIV
FAILURE REPARTITION BASED ON PREDICTION ACCURACY AND FALSE ALARM RATE

Estimated number of failures	Number of pre-dicted failures	Number of unpre-dicted failures	Number of false alarms	FT overhead
1	1	0	0	c_m
2	1	1	1	$2 * c_m + c_c + c_r$
3	2	1	1	$3 * c_m + c_c + c_r$
4	3	1	1	$4 * c_m + c_c + c_r$
5	4	1	1	$5 * c_m + c_c + c_r$
9	6	3	3	$9 * c_m + 3 * (c_c + c_r)$
10	7	3	3	$10 * c_m + 3 * (c_c + c_r)$
12	8	4	4	$12 * c_m + 4 * (c_c + c_r)$
14	10	4	4	$14 * c_m + 4 * (c_c + c_r)$
16	11	5	5	$16 * c_m + 5 * (c_c + c_r)$
26	18	8	8	$26 * c_m + 8 * (c_c + c_r)$
28	20	8	8	$28 * c_m + 8 * (c_c + c_r)$
29	20	9	9	$29 * c_m + 9 * (c_c + c_r)$
41	29	12	12	$41 * c_m + 12 * (c_c + c_r)$
46	32	14	14	$46 * c_m + 14 * (c_c + c_r)$
49	34	15	15	$49 * c_m + 15 * (c_c + c_r)$
55	39	16	16	$55 * c_m + 16 * (c_c + c_r)$
56	39	17	17	$56 * c_m + 17 * (c_c + c_r)$
61	42	18	18	$60 * c_m + 18 * (c_c + c_r)$
63	44	19	19	$63 * c_m + 19 * (c_c + c_r)$
65	46	19	19	$65 * c_m + 19 * (c_c + c_r)$
68	48	20	20	$68 * c_m + 20 * (c_c + c_r)$
80	56	24	24	$80 * c_m + 24 * (c_c + c_r)$
105	74	31	31	$105 * c_m + 31 * (c_c + c_r)$
113	79	34	34	$113 * c_m + 34 * (c_c + c_r)$
114	80	34	34	$114 * c_m + 34 * (c_c + c_r)$
117	82	35	35	$117 * c_m + 35 * (c_c + c_r)$

TABLE XV
ANALYSIS OF THE FT OVERHEAD W/ AN HYBRID POLICY, STARTING AT t_0 OF THE EXECUTION PLATFORM LIFETIME

FT overhead (in hours)	FT cost
12	$2 * c_m + c_c + c_r$
24	$3 * c_m + c_c + c_r$
36	$5 * c_m + c_c + c_r$
48	$12 * c_m + 4 * (c_c + c_r)$
60	$12 * c_m + 4 * (c_c + c_r)$
72	$28 * c_m + 8 * (c_c + c_r)$
84	$41 * c_m + 12 * (c_c + c_r)$
96	$60 * c_m + 18 * (c_c + c_r)$
108	$68 * c_m + 20 * (c_c + c_r)$
120	$80 * c_m + 24 * (c_c + c_r)$
132	$117 * c_m + 35 * (c_c + c_r)$
144	$117 * c_m + 35 * (c_c + c_r)$

We highlight (text in red) cases where the configuration may lead to issue, due to the fact that an important number of migrations and checkpoints/restarts have to be performed in a limited period of time. It is interesting to notice that such cases happen more often for 720–1080 hours applications when not running at the beginning of the platform life time. This may be due to the failure distribution (which is not linear over the platform lifetime).

TABLE XVI
ANALYSIS OF THE FT OVERHEAD W/ AN HYBRID POLICY, STARTING AT $t = 192$ hours OF THE PLATFORM LIFETIME

FT overhead (in hours)	FT cost
12	n/a
24	$10 * c_m + 3 * (c_c + c_r)$
36	$26 * c_m + 8 * (c_c + c_r)$
48	$46 * c_m + 14 * (c_c + c_r)$
60	$56 * c_m + 17 * (c_c + c_r)$
72	$63 * c_m + 19 * (c_c + c_r)$
84	$105 * c_m + 31 * (c_c + c_r)$
96	$105 * c_m + 31 * (c_c + c_r)$
108	$105 * c_m + 31 * (c_c + c_r)$
120	$105 * c_m + 31 * (c_c + c_r)$
132	$105 * c_m + 31 * (c_c + c_r)$
144	$105 * c_m + 31 * (c_c + c_r)$

TABLE XVII
ANALYSIS OF THE FT OVERHEAD W/ AN HYBRID POLICY, STARTING AT $t = 504$ hours OF THE PLATFORM LIFETIME

FT overhead (in hours)	FT cost
12	c_m
24	$4 * c_m + c_c + c_r$
36	$9 * c_m + 3 * (c_c + c_r)$
48	$14 * c_m + 4 * (c_c + c_r)$
60	$28 * c_m + 8 * (c_c + c_r)$
72	$55 * c_m + 16 * (c_c + c_r)$
84	$65 * c_m + 19 * (c_c + c_r)$
96	$68 * c_m + 20 * (c_c + c_r)$
108	$113 * c_m + 34 * (c_c + c_r)$
120	$114 * c_m + 34 * (c_c + c_r)$
132	$114 * c_m + 34 * (c_c + c_r)$
144	$114 * c_m + 34 * (c_c + c_r)$

VI. CONCLUSION

This document presents a study of the impact and feasibility of different fault tolerance policies on the execution of different applications, based on system logs of a 512 nodes cluster. Since the system did not include any FT mechanisms, we fixed some parameters that “seem to be realistic” (based on the literature), but it could be easily extended to other cases, modifying parameters such as the checkpoint cost, the restart cost, the prediction accuracy and so on.

TABLE XVIII
ANALYSIS OF THE FT OVERHEAD W/ AN HYBRID POLICY, STARTING AT $t = 660$ hours OF THE PLATFORM LIFETIME

FT overhead (in hours)	FT cost
12	$16 * c_m + 5 * (c_c + c_r)$
24	$29 * c_m + 9 * (c_c + c_r)$
36	$49 * c_m + 15 * (c_c + c_r)$
48	$56 * c_m + 17 * (c_c + c_r)$
60	$68 * c_m + 20 * (c_c + c_r)$
72	$105 * c_m + 31 * (c_c + c_r)$
84	$105 * c_m + 31 * (c_c + c_r)$
96	$105 * c_m + 31 * (c_c + c_r)$
108	$105 * c_m + 31 * (c_c + c_r)$
120	$105 * c_m + 31 * (c_c + c_r)$
132	$105 * c_m + 31 * (c_c + c_r)$
144	$105 * c_m + 31 * (c_c + c_r)$

The results show that there is no “one-fit-all” solution: based on the platform (typically the failure distribution and the overhead created by deployed FT mechanisms), the time slot allocated for application execution, and application execution time, different fault tolerance policies offer the best results.

Another future task may be to use different system logs for further analysis and analyze if the results are coherent between platforms having different sizes and different configurations.

REFERENCES

- [1] S. Chakravorty, C. Mendes, and L. Kale, “Proactive fault tolerance in large systems,” *HPCRI: 1st Workshop on High Performance Computing Reliability Issues*, in *Proceedings of the 11th International Symposium on High Performance Computer Architecture (HPCA-11)*. IEEE Computer Society, 2005.
- [2] Y. Li and Z. Lan, “Exploit failure prediction for adaptive fault-tolerance in cluster computing,” *CCGrid*, vol. 0, pp. 531–538, 2006.
- [3] A. Tikotekar, G. Vallée, T. Naughton, S. L. Scott, and C. Leangsuksun, “Evaluation of fault-tolerant policies using simulation,” in *Proceedings of IEEE Cluster 2007*. Austin, Texas, USA: IEEE Computer Society, Sep. 17-20, 2007.
- [4] N. Raju, Gottumukkala, Y. Liu, C. B. Leangsuksun, R. Nassar, and S. Scott, “Reliability analysis in hpc clusters,” in *HAPCW'06: High Availability and Performance Computing Workshop*. Santa Fe, New Mexico, USA: Held in conjunction with LACSI 2006, Oct. 2006.