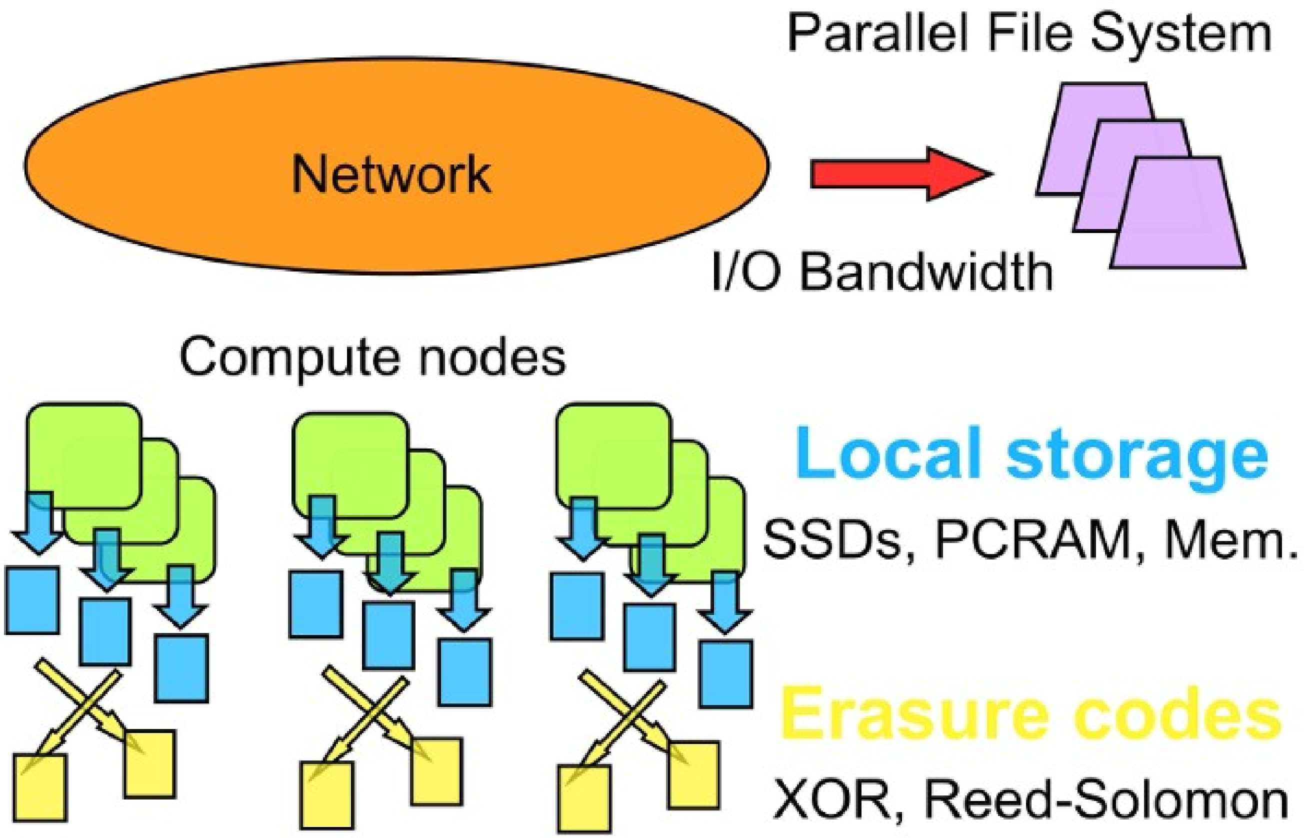
Fast checkpoint restart for sustained petascale computing: Opportunities and directions

Leonardo A. BAUTISTA GOMEZ

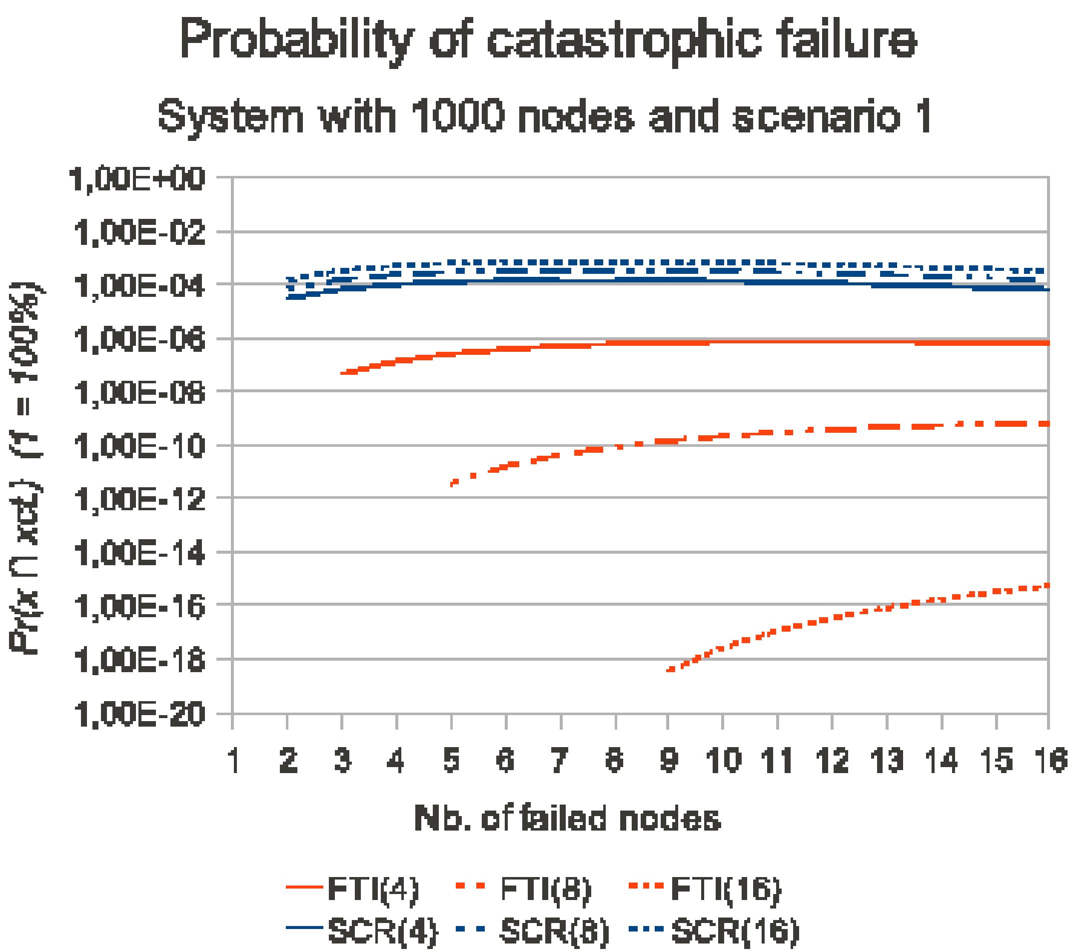
Background


Parallel File System
Network
I/O Bandwidth
Compute nodes
Local storage
SSDs, PCRAM, Mem.
Erasure codes
XOR, Reed-Solomon

Topology-Aware RS encoding

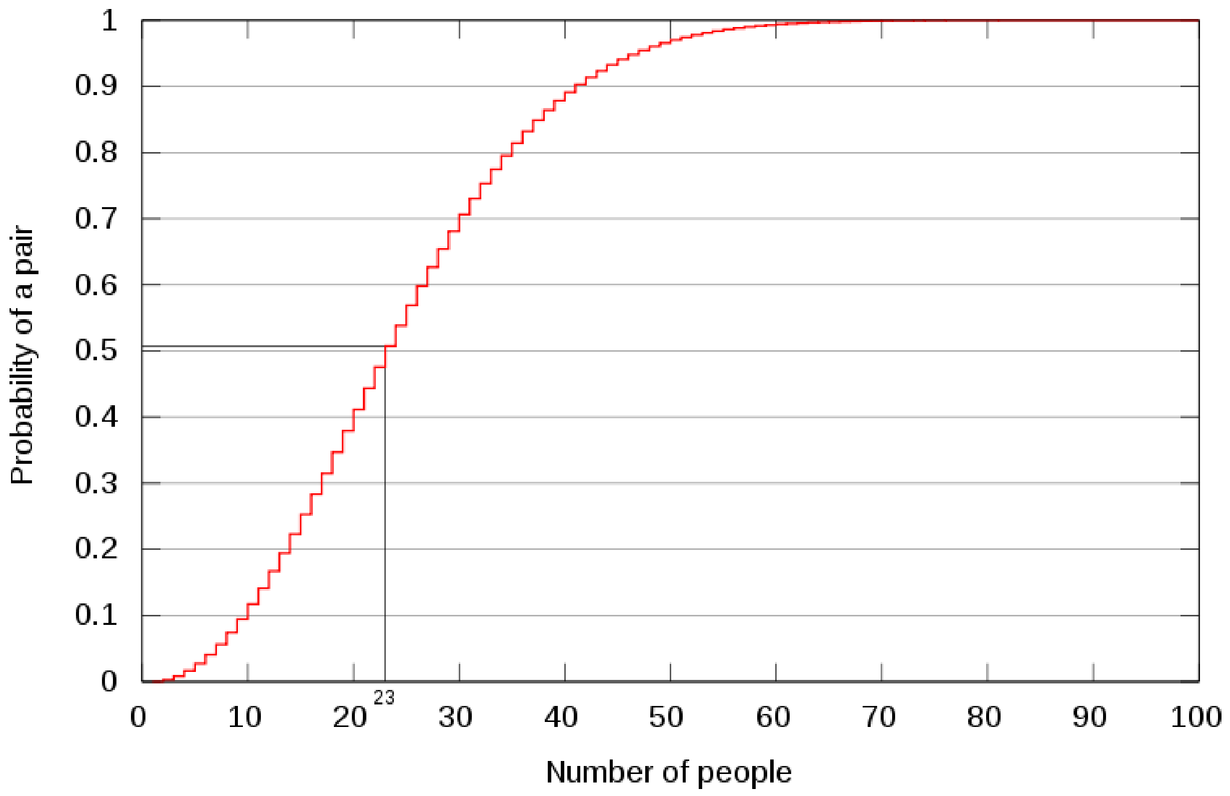
$$\Pr(x_{Ct.} \mid x) = \frac{\binom{g}{1} * \binom{k}{t+1} * \binom{n-(t+1)}{x-(t+1)}}{\binom{n}{x}}$$

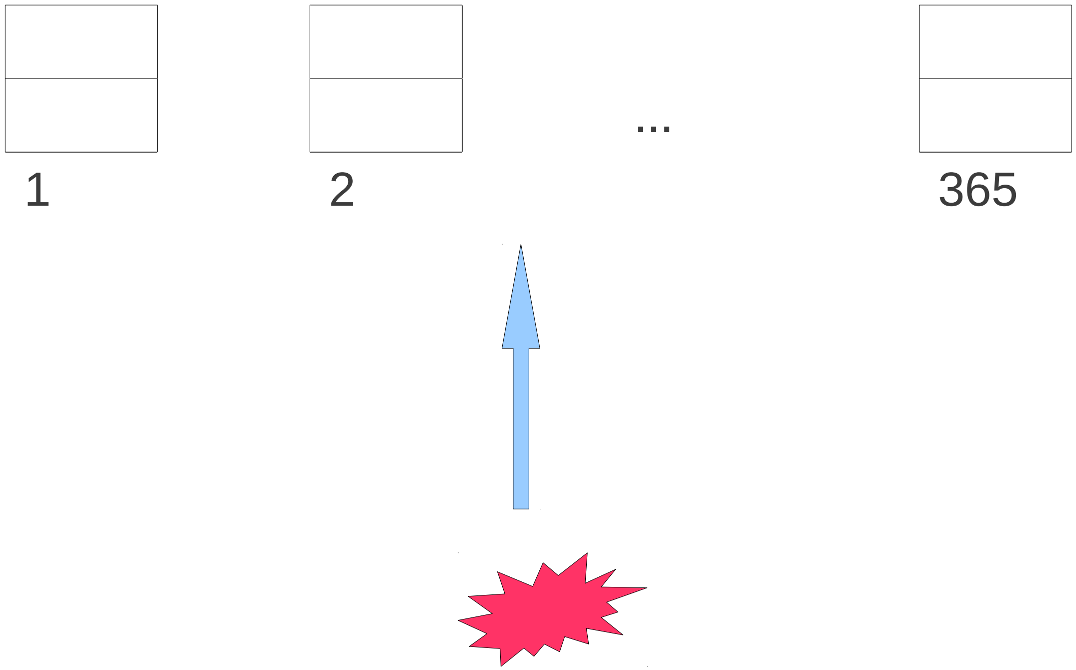
Parameter	Description
n	Total number of nodes in the system $(n = k * g)$
k	Size of the encoding groups
g	Number of encoding groups
t	Number of node failures tolerated per group
x	Number of failed nodes for a given failure


Probability of catastrophic failure
System with 1000 nodes and scenario 1
Pr(x ∩ xct.) (1 = 100%)
1,00E+00
1,00E-02
1,00E-04
1,00E-06
1,00E-08
1,00E-10
1,00E-12
1,00E-14
1,00E-16
1,00E-18
1,00E-20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Nb. of failed nodes
FTI(4)
FTI(8)
FTI(16)
SCR(4)
SCR(8)
SCR(16)

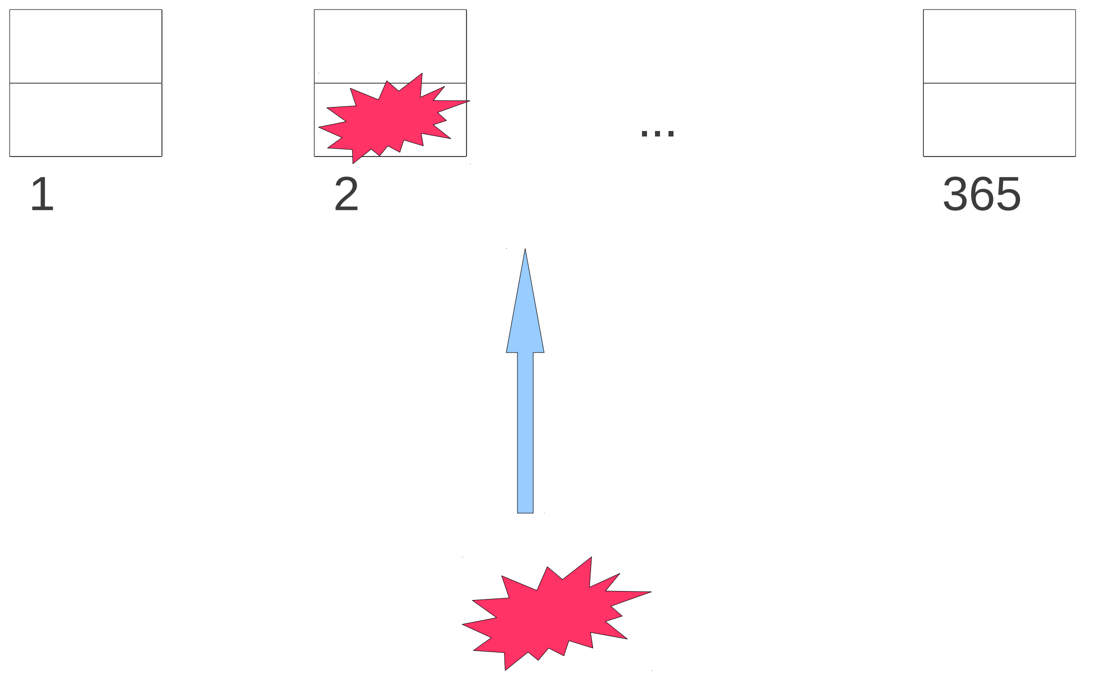
Process replication reliability


HAPPY
BIRTHDAY


Probability of a pair
Number of people
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
23
30
40
50
60
70
80
90
100



365/365 * 364/365 * 363/365 * …



365/365 * 364/365 * 363/365 * …

$$\overline{n} = 1 + Q(M)$$

$$Q(M) = \sum_{k=1}^{M} \frac{M!}{(M-k)!M^k}.$$

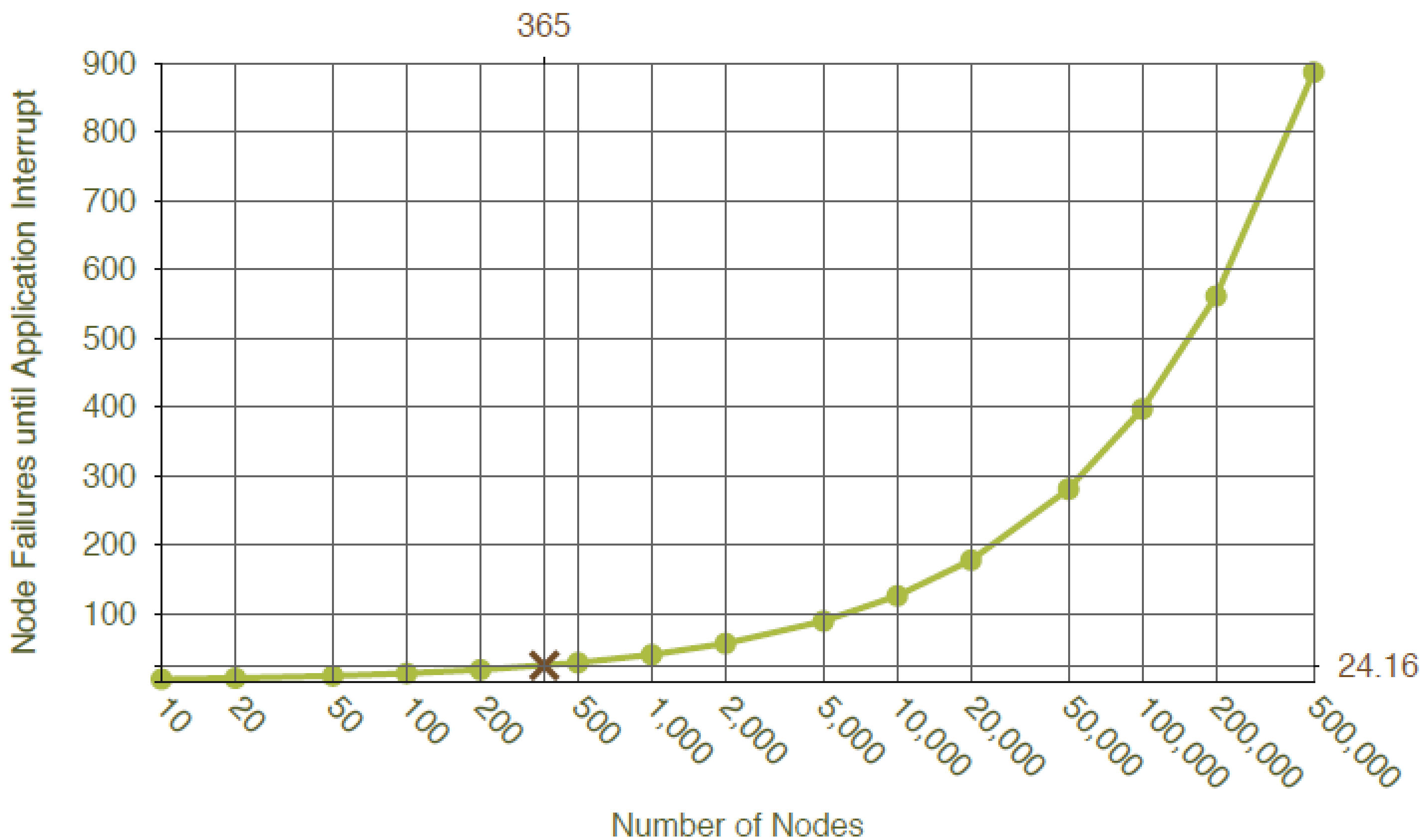


Figure 1. Expected number of node failures before an application interrupt in a system with redundant nodes. Numbers are calculated using the birthday problem Equation 2.

[My first analysis of an algorithm, originally done during summer 1962 in Madison.]

8354

NOTES ON "OPEN" ADDRESSING. D. Knuth 7/22/63

1. Introduction and Definitions. Open addressing is a widely-used technique for keeping "symbol tables." The method was first used in 1954 by Samuel, Amdahl, and Boehme in an assembly program for the IBM 701. An extensive discussion of the method was given by Peterson in 1957 [1], and frequent references have been made to it ever since (e.g. Schay and Spruth [2], Iverson [3]). However, the timing characteristics have apparently never been exactly established, and indeed the author has heard reports of several reputable mathematicians who failed to find the solution after some trial. Therefore it is the purpose of this note to indicate one way by which the solution can be obtained.

We will use the following abstract model to describe the method: N is a positive integer, and we have an array of N variables $x_1, x_2, \ldots, x_N$. At the beginning, $x_i = 0$, for $1 \leqq i \leqq N$.

To "enter the k-th item in the table," we mean that an integer a_k is calculated, $1 \leqq a_k \leqq N$, depending only on the item, and the following process is carried out:

1. Set $j = a_k$.
2. "The comparison step." If $x_j = 0$, set $x_j = 1$ and stop; we say "the k-th item has fallen into position x_j."
3. If $j = N$, go to step 5.
4. Increase j by 1 and return to step 2.
5. "The overflow step." If this step is entered twice, the table is full, i.e. $x_i = 1$ for $1 \leqq i \leqq N$. Otherwise set j to 1 and return to step 2.

Observe the cyclic character of this algorithm.

We are concerned with the statistics of this method, with respect to the number of times the comparison step must be executed. More precisely, we consider all of the N^k possible sequences $a_1, a_2, \ldots a_k$ to be equally probable, and we ask, "What is the probability that the comparison step is used precisely m times when the k-th item is placed?"

2. Non-overflow (self-contained) sequences.

Let $\genfrac{[}{]}{0pt}{}{n}{k}$ denote the number of sequences $a_1, a_2, \ldots a_k$ $(1 \leqq a_i \leqq n)$ in which no overflow step occurs during the entire process of placing k items, if the algorithm is used for $N = n$. (By convention, we set

$$\genfrac{[}{]}{0pt}{}{n}{0} = 1.)$$

Lemma 1: If $0 \leq k \leq n+1$, then $\genfrac{[}{]}{0pt}{}{n}{k} = (n+1)^k - k(n+1)^{k-1}$.

Proof: This proof is based on the fact that $\genfrac{[}{]}{0pt}{}{n}{k}$ is precisely the number of sequences $b_1, b_2, \ldots b_k$ $(1 \leqq b_i \leqq n+1)$ in which, if the algorithm is carried out for $N = n+1$, then $x_{n+1} = 0$ at the end of the operation. This follows because every sequence of the former type is one of the latter, and conversely the condition implies in particular that $1 \leqq b_i \leqq n$, and that no overflow step occurs.

But sequences of the latter type are easily enumerated, because the algorithm has circular symmetry; of the $(n+1)^k$ possible sequences $b_1, b_2, \ldots b_k$, exactly $k/(n+1)$ of these leave $x_{n+1} \neq 0$. This shows that

$$\genfrac{[}{]}{0pt}{}{n}{k} = (n+1)^k \left(1 - \frac{k}{n+1}\right).$$

[2]

3. Sequential pile-up.

How many sequences $a_1, \ldots a_{k-1}$ $(1 \leqq a_i \leqq N)$ leave

$$x_{N-t-1} = 0,\ x_{N-t} = \cdots = x_{N-1} = 1,\ x_N = 0?$$

Let this number be denoted by $Q(N,k,t)$.

Lemma 2: If $0 < k \leqq N$, $0 \leqq t < N-1$, then $Q(N,k,t) = \binom{k-1}{t} \genfrac{[}{]}{0pt}{}{N-t-2}{k-t-1} \genfrac{[}{]}{0pt}{}{t}{t}$.

Proof: In order to construct such a sequence, we have a subsequence of t items which fall into the range x_{N-t} through x_{N-1}; there are $\genfrac{[}{]}{0pt}{}{t}{t}$ such sequences.

The remaining terms form a subsequence of $k-t-1$ items which all land in the range x_1 through x_{N-t-2}; there are $\genfrac{[}{]}{0pt}{}{N-t-2}{k-t-1}$ such sequences. Finally, there are $\binom{k-1}{t}$ ways to put these two subsequences together. This completes the proof.

Notice that the stated formula for $Q(N,k,t)$ is valid also for the excluded case $t = N-1$, if we adopt the convention that

$$\genfrac{[}{]}{0pt}{}{-1}{0} = 1.$$

4. The probability $P(N,k,m)$.

Let $P(N,k,m)$ be the probability that m comparison steps are required to place the k-th item, i.e. step 2 of the algorithm is entered m times.

Lemma 3: The number of sequences $a_1, \ldots a_k$ $(1 \leqq a_i \leqq N)$ in which the k-th item falls in position x_N after precisely m comparisons, is

$$\sum_{i=m}^{N} Q(N,k,i-1) = \genfrac{[}{]}{0pt}{}{N-1}{k-1} - \sum_{i=1}^{m-1} Q(N,k,i-1), \quad \text{for } 1 \leqq k \leqq N.$$

Proof: We must have $a_k = N-m+1$; and after the first $k-1$ steps, we must have $x_i = 1$ for $N-m+1 \leq i < N$, and also $x_N = 0$. Therefore by Lemma 2, the stated formula is obvious, in lieu of the fact that

$$\sum_{i=1}^{N} Q(N,k,i-1) = \genfrac{[}{]}{0pt}{}{N-1}{k-1}.$$

(the number of sequences which leave $x_N = 0$).

Lemma 4: $P(N,k,m) = \frac{1}{N^{k-1}} \sum_{i=m}^{N} Q(N,k,i-1) = 1 - \frac{k-1}{N} - \frac{1}{N^{k-1}} \sum_{i=1}^{m-1} Q(N,k,i-1)$.

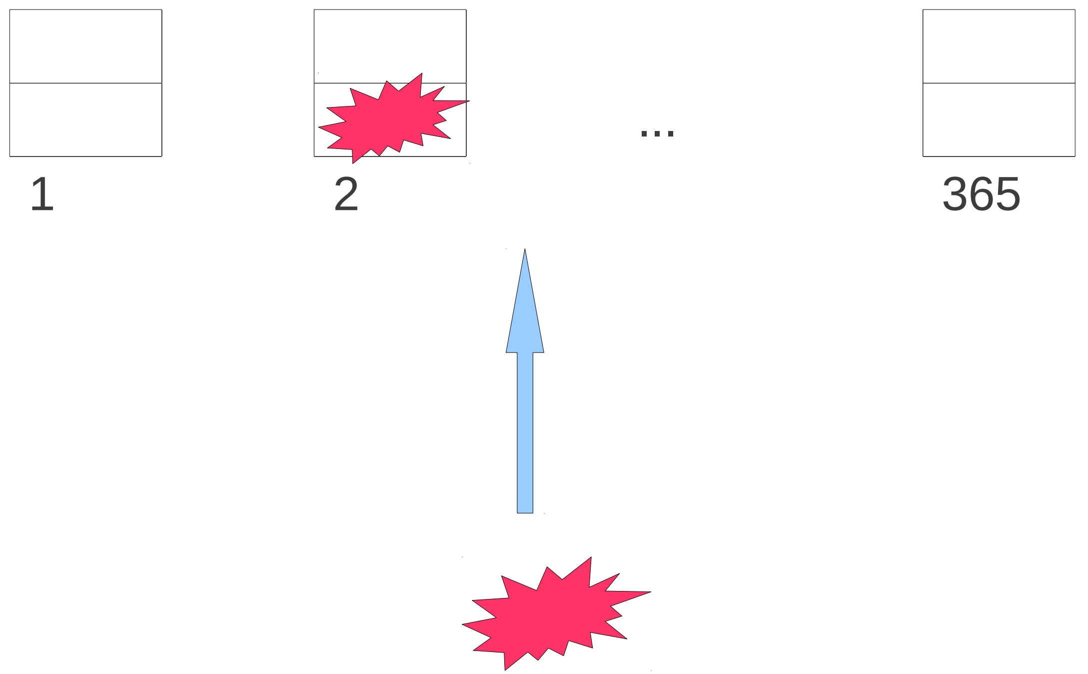
Proof: The position x_N in lemma 3 can be changed to x_i for any other i without affecting the result, by symmetry. There are N^k possible sequences $a_1, \ldots a_k$ $(1 \leqq a_i \leqq N)$, each assumed to be equally probable; hence $P(N,k,m)$ is the appropriate fraction of these sequences, and the result is immediate from Lemma 3.

Now we let $R(N,k,t) = Q(N,k,t-1)/N^{k-3}(N-k)$. By lemmas 1 and 2, we find that for $1 \leqq t < N$, $1 \leq k < N$,

$$R(N,k,t) = \binom{k-1}{t-1}(N-t)^{k-t-1}(N-k)\, t^{t-2} \big/ N^{k-3}(N-k)$$

so that

$$R(N,k,t) = \binom{k-1}{t-1}\left(\frac{t}{N}\right)^{t-2}\left(1-\frac{t}{N}\right)^{k-t-1}.$$



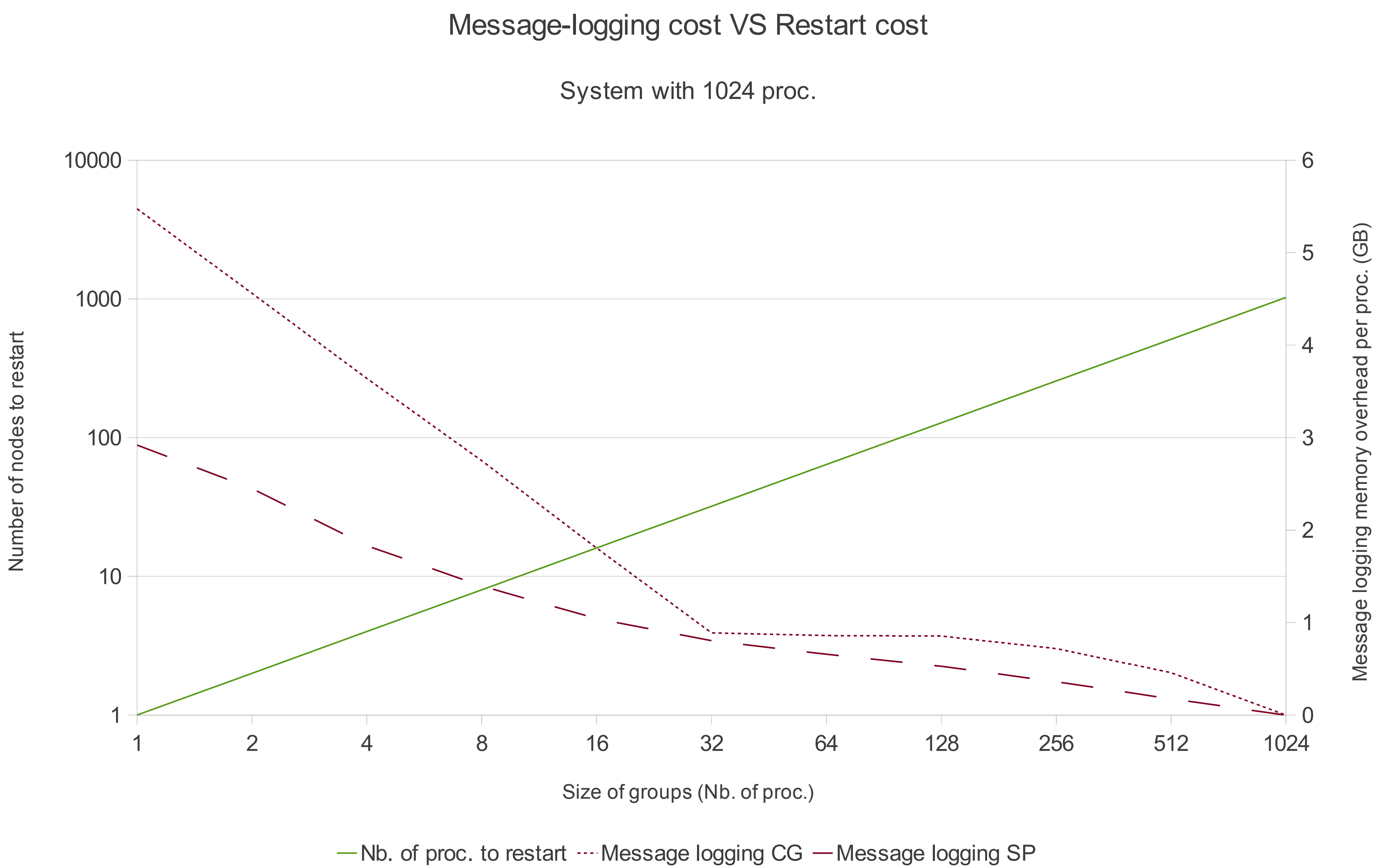
$$365/365 * 364/365 * 363/365 * \ldots$$

$$730/730 * 728/729 * 726/728 * \ldots$$

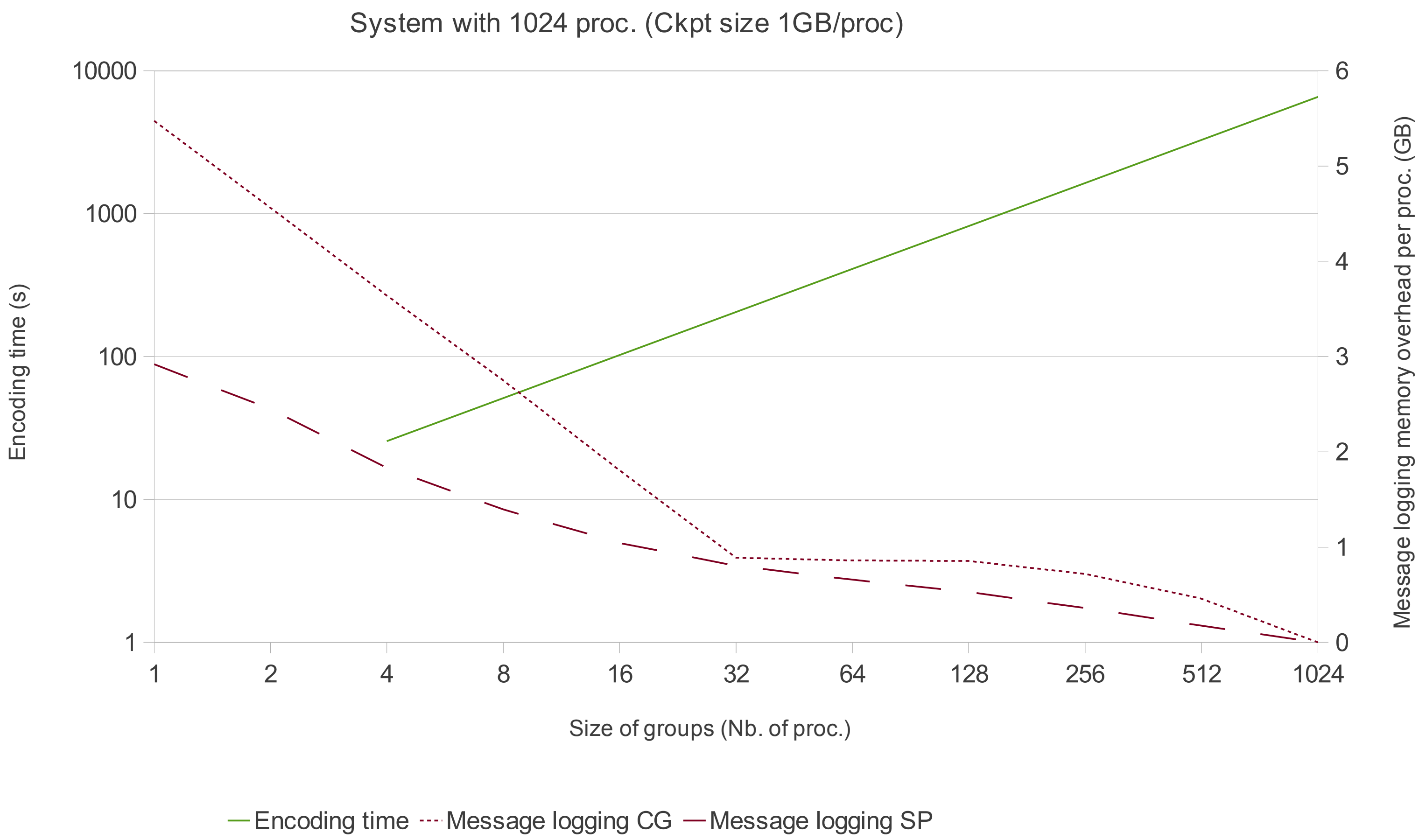
Erasure codes

VS

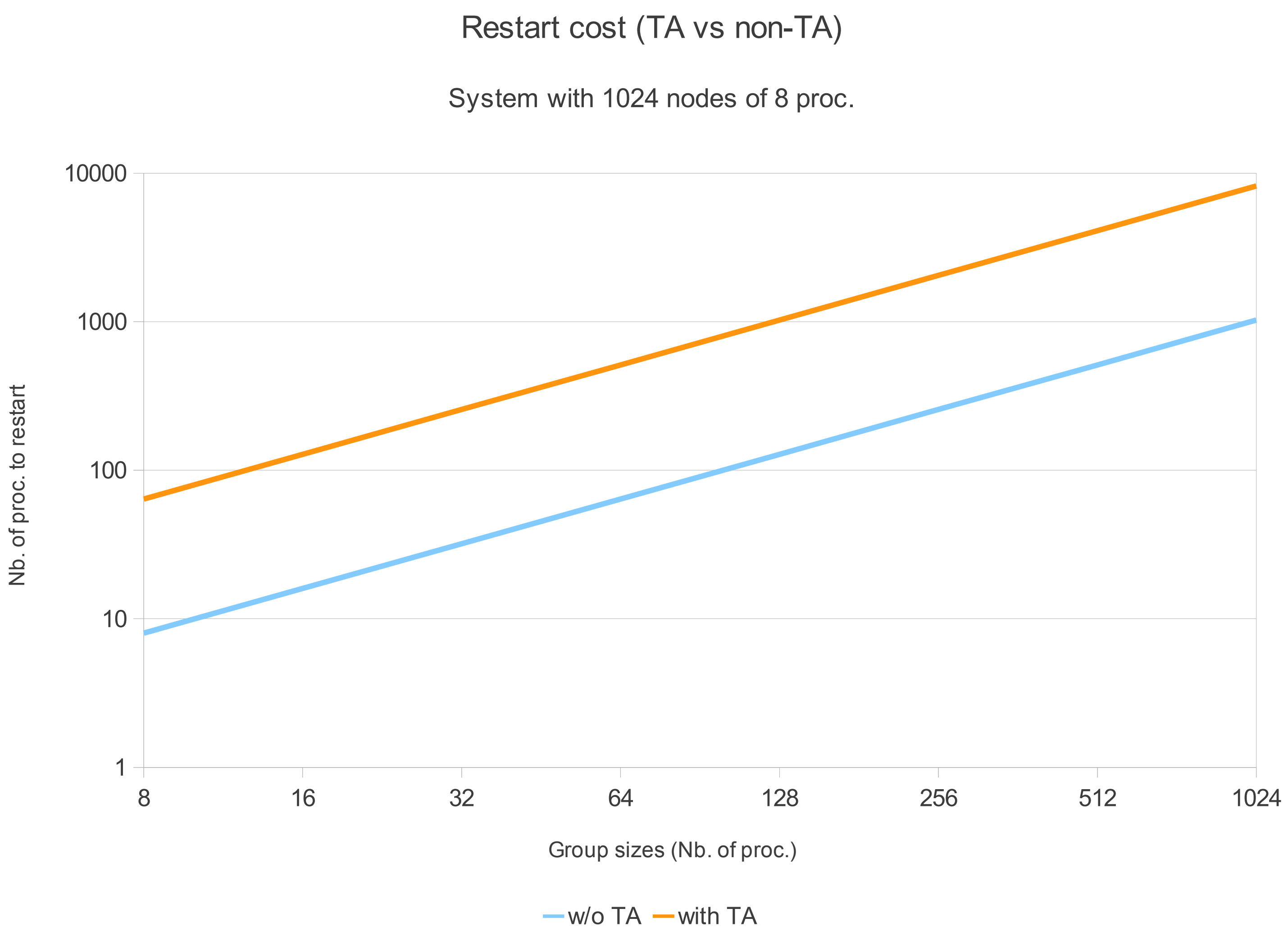
Partial restart


Message-logging cost VS Restart cost
System with 1024 proc.
Number of nodes to restart
10000
1000
100
10
1
Message logging memory overhead per proc. (GB)
6
5
4
3
2
1
0
1
2
4
8
16
32
64
128
256
512
1024
Size of groups (Nb. of proc.)
Nb. of proc. to restart
Message logging CG
Message logging SP

Message-logging VS encoding time



Topology-Aware RS encoding

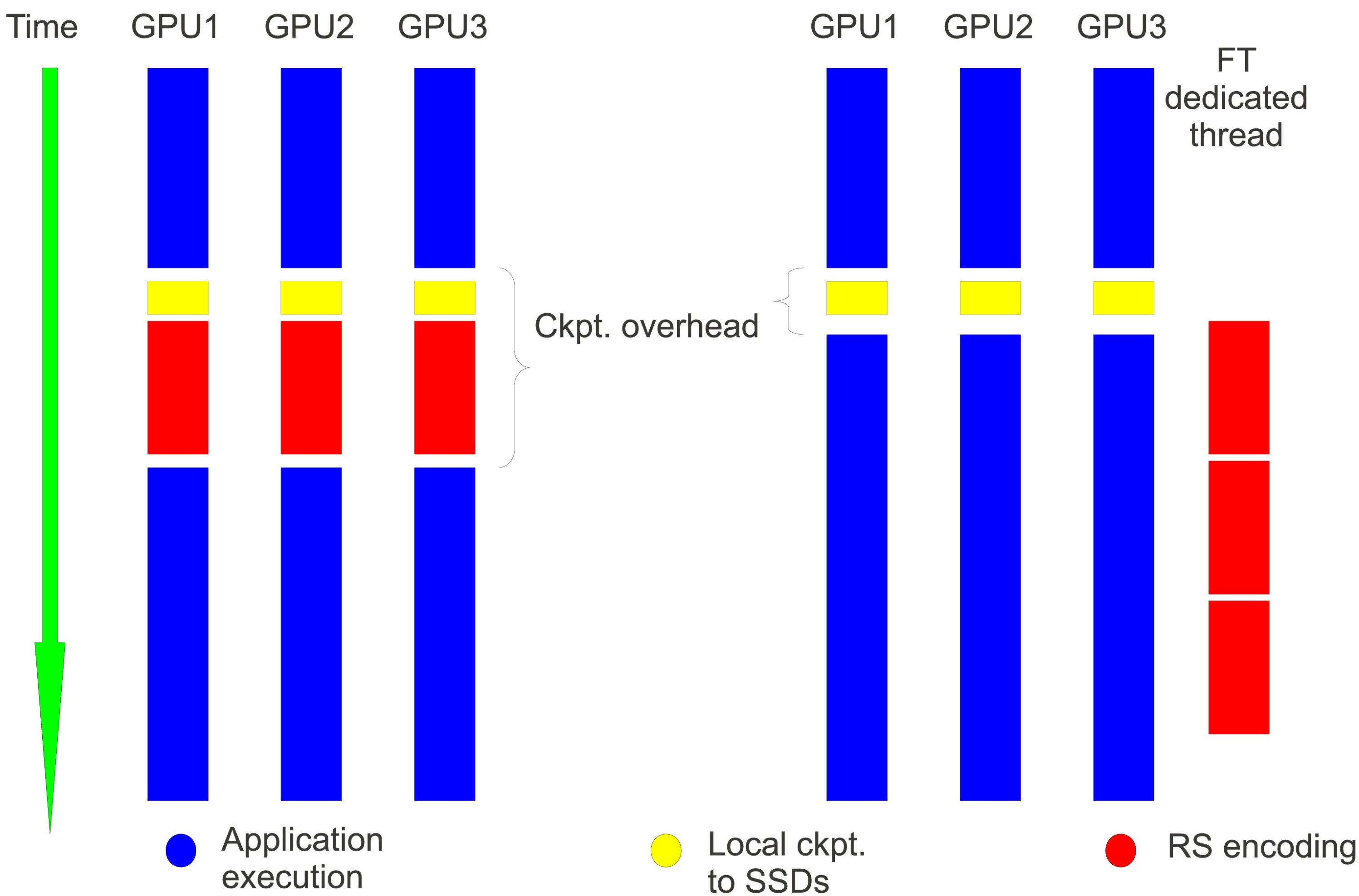

Restart cost (TA vs non-TA)
System with 1024 nodes of 8 proc.
10000
1000
100
10
1
Nb. of proc. to restart
8
16
32
64
128
256
512
1024
Group sizes (Nb. of proc.)
w/o TA
with TA

Topology-Aware RS encoding

FT-dedicated threads

&

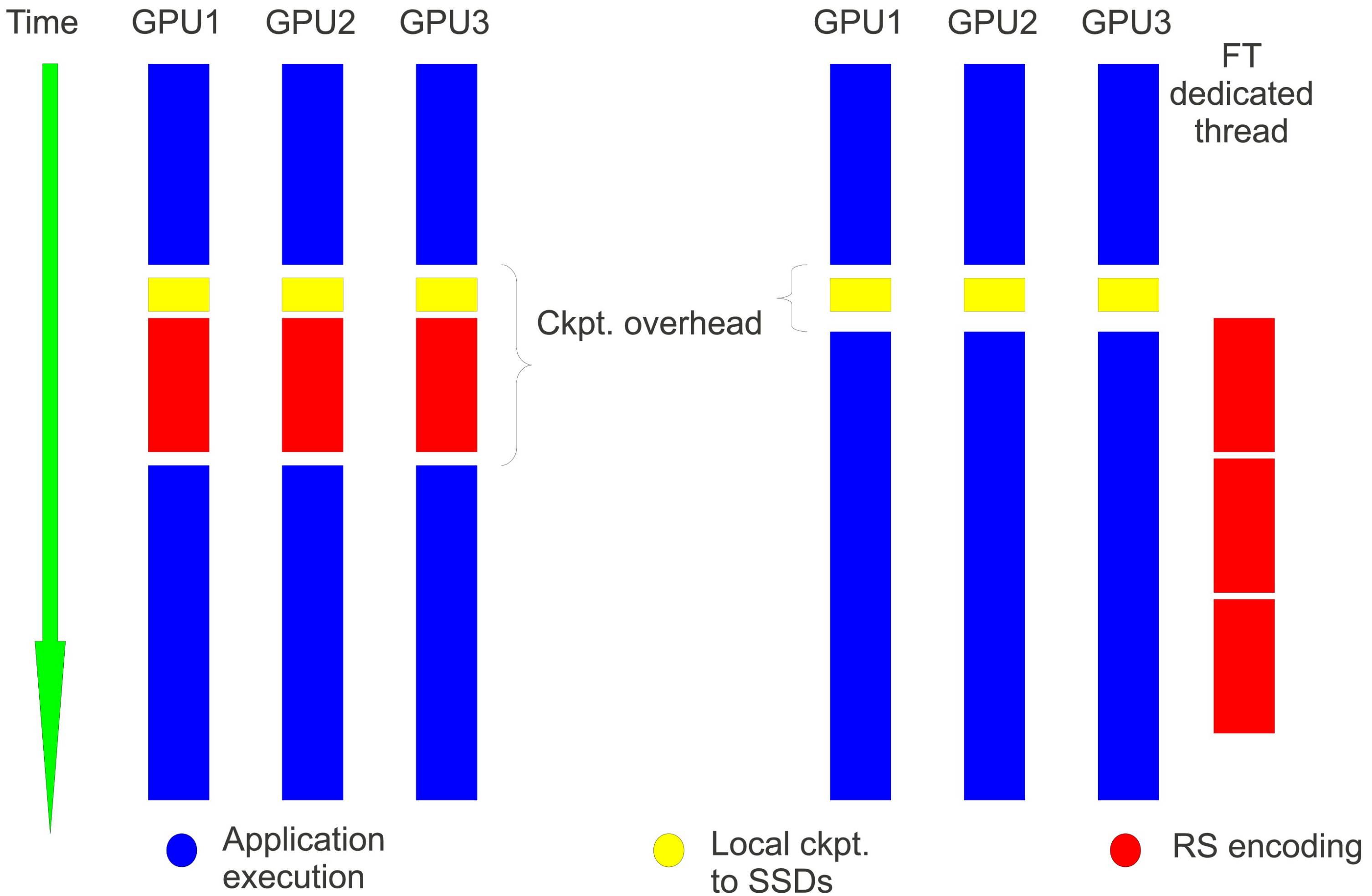
Fault predictions


Time
GPU1
GPU2
GPU3
Ckpt. overhead
GPU1
GPU2
GPU3
FT dedicated thread
Application execution
Local ckpt. to SSDs
RS encoding

FT-dedicated threads

&

Silent errors


Time
GPU1
GPU2
GPU3
Ckpt. overhead
GPU1
GPU2
GPU3
FT dedicated thread
Application execution
Local ckpt. to SSDs
RS encoding

Thank you

Questions?