Storage Reliability Calculations

02 December 2016

When designing storage systems, for example comparing different coding schema, it is necessary to calculate reliability numbers. Here I summarize the useful concepts from reliability engineering and their math relations.

Reliability engineering

```
λ = failure rate, expressed in failures per unit of time
MTBF (mean time between failures) = 1/λ    // why? see hazard function and exponential distribution
AFR = 1 - e^(-exposure_time/MTBF) = 1-e^(-exposure_time*λ) = probability of failure occurs within the exposure_
Assuming a small AFR (<5%), we can approximately have AFR ≈ 8760/MTBF
1 - AFR = probability of failure does not occur (survived) within the exposure_time
```

Hazard function

```
T = a random variable of the time until some event of interest happens
f(t) = probability density of event happens at t
F(t) = P(T<=t) = probability of event happens before t

S(t) = survivor function / reliability function := 1 − F(t) = probability of event happens after t, or say even

h(t) = hazard function := lim(h->0+, P[t ≤ T < t + h|T ≥ t]/h) = f(t)/S(t-) = - d ln(1-F(t)) / dt = - d ln(S(t)
h(t)dt = f(t)dt/S(t) ≈ P[fail in [t,t+dt) | survive until t]
H(t) = cumulative hazard function := ∫ (0->t, h(u)du) (t>0) = -ln(1-F(t)) = -ln(S(t))
So, S(t) = e^−H(t), f(t) = h(t)e^−H(t)

Exponential Distribution: T ~ Exp(λ), for t>0.
    f(t) = λe^−λt for λ > 0 (scale parameter)
    F(t) = 1 − e^−λt    // this is the AFR mentioned above
    S(t) = e^−λt    // this is the (1 - AFR) mentioned above
    h(t) = λ    // constant hazard function. this is the failure rate mentioned above
    H(t) = λt
    Characteristics:
        E(T) = 1/λ    // this is the MTBF mentioned above
        "Lack of Memory": P[T > t] = P[T > t + t0|T > t0]. Probability of surviving another t time units does n
```

Reference: [1] (http://web.stanford.edu/~lutian/coursepdf/unit1.pdf)

Calculating MTBF from observation

```
observed AFR = failed device count / total device count
calculated MTBF(hrs) = 8760 / AFR    // 8760 is the number of hours in a year
exposure_time: for storage system with auto repair, the exposure time of second node failure should be the reco
failure possibility in a given time window = F(t) = 1 − e^−λt ≈ λt = t/MTBF = t * AFR / 8760, t is the time win
```

Availability vs durability

```
Availability: the probability of data is accessible by user. The data needs to be on disk, and server nodes hav
Durability: the probability of data is safe on disk. The data may not available to user, but still on disk.
For example: AWS S3 offers 99.999999999% durability and 99.9% availability.
```

Data loss probabilities

```
Basics

    A(n, k) := Permutation(n, k) = n! / (n-k)!
    C(n, k) := Combination(n, k) = n! / ((n-k)! k!)
    Zm := {0, 1, .., m-1}
    Subset(A) := {a | a is subset of A}
    ¬A := overbar A := complementary set of A
    I := the complete set = A + ¬A
    Code schema (D-K, K) := the simplest erasure-coding schema of D-K data fragments and K parity fragments. Wh
    P(event): a probability is operating on event. An event is a set. So set algebra applies to events.
    ∩: event1 AND. This is aligned with set algebra
    ∪: event1 OR. This is aligned with set algebra
    ¬ or overbar: event NOT happen. This is aligned with set algebra

Basic formulas (proof omitted)

    Event breakdown formula: (aux1)
    Multiple event breakdown formula: (aux2)
    If Ai are mutually independent, then ¬Ai are mutually independent: (aux3)
    Conditional probability can reverse: (aux4)
    If Ai are mutually conditional independent given N, then ¬Ai are mutually conditional independent given N:
    Space division, which is useful for node failure probability: (aux6)

Notations

    Cluster setup:

        We have N nodes, each have independent probability p to fail.
        We have M extents (extent is the basic data unit, i.e. object), each is encoded into a D-fragment code.
        We randomly place the fragments on nodes, each node holds at most 1 fragment from a given extent.
        The code schema can be complex, e.g. LRC code.
        The placement may have constraints, e.g. domains, copysets.

    Node failure

        Node set is ZN ("N" in subscript)
        NL := a node failure event. NL is a subset of Zn.
        NLS := the space of NL = Subset(Zn)

    Extent and Extent Loss

        Ei := Extent i, i in ZM ("M" in subscript)
        EL := an extent loss event = {indexes of the lost fragments}    // fragment index is element of ZD ("D"
        ELS := the space of EL = {EL|any EL}, it is the set of which fragment-loss can cause extent loss
        ¬ELS := Subset(ZD) - ELS, it is the set of safe fragment-loss

    Placement

        PM := a placement = [node indexes of the placed fragment, for fragment from idx=0 to idx=D-1]
        PMi := extent i's placement
```

/

Sall ("all" in subscript) := {all possible placements }
S := set of the allowed placements, S is subset of Sall, due to placement constraints
SNL ("NL" in subscript) := {placement that will cause extent loss}, given NL
∩h ("h" in subscript) := set intersect and map node index to fragment index. if (PM ∩h NL) is element o

Data loss

Ei DL := an event of extent i is lost. P(Ei DL) is the probability of lost extent i
Ei DL | NL = extent i loses, given NL. P(Ei DL|NL) is the conditional probability of Ei DL given NL
DL := an event of data loss. P(DL) = P(DL>0) is the probability of losing at least 1 extent in the clus

Assumptions

Extent placements are mutually independent: (asp1)
Given an extent, placements are exclusive: (asp2)
Placement is evenly randomized: (asp3)

Node failure is exclusive: (asp4)
Node failure is independent with extent placement: (asp5)

Implications (proof omitted)

Extent placement sets are independent: (imp1)
Node failure is independent with extent placement sets: (imp2)
Sample space (elementary event space)
An elementary event is: (NL, PM0, PM1, .., PMM-1)   ("0", "1", .., "M-1" in subscript)
The sample space is: NLS × S × S × ... × S   (count of S = M)
Given NL, P(0) := elementary event probability, P(0) formula: (imp3)
Key takeaway: most of the proofs below is conducted by breaking events down to elementary events
probability is essentially counting elementary events

Conclusions (proof omitted)

Calculate P(Ei DL|NL): (thr1)
(Ei DL|NL) are mutually conditional independent: (thr2)
(Ei DL|NL) and ¬(Ei DL|NL) are mutually conditional independent: (thr3)
Calculate P(Ei DL): (thr4)   // To prove the two forms are equal, breakdown them into P(0)
Ei DL are not independent: (thr5)
Alternative assumption, Ei DL are still not independent: Assume extent placements are fixed/known, the samp
Data loss count distribution: (thr6)
Data loss probability: (thr7)
P(Ei DL) vs P(DL): connected by |SNL|/|S| ("NL" in subscript), but not always matched
Counter-case: from (N, p, D, K, M) = (20, 0.16, 4, 1, 50) to (20, 0.16, 12, 2, 50), P(DL) drops, but P(
P(Ei DL) vs P(DL): meaning
P(Ei DL): when a customer asks how easy is his extent to be lost.
P(DL): when the operation asks how easy will they encouter data loss alert, and how much lost data.

Questions

If we have a large number of nodes, and many extents, will the extent loss event become nearly independent?
We can think this way

```
        Since cluster is large, (Ei DL) is less affected by (Ej DL).
        When we take extents E0, E1, ..., Em-1, they are more likely to be not overlapped
            When E0, E1, ..., Em-1 are not overlapped, we can deduce that Ei DL are independent
```

Below are the formulas stated above

(aux1) Given $B_i \cap B_j = \emptyset$, $\forall\, i \neq j \in Z_m$, and $\bigcup_{i\in Z_m} B_i = I$

$P(A) = \sum_{i\in Z_m} P(B_i \cap A) = \sum_{i\in Z_m} P(B_i)P(A|B_i)$

(aux2) Given $A_{i,j}$, $i\in Z_n$, $j\in Z_{m_i}$; $A_{i,j_1} \cap A_{i,j_2} = \emptyset$, $\forall i\in Z_n$, $\forall j_1\neq j_2 \in Z_{m_i}$. We have

$P(\bigcap_{i\in Z_n} \bigcup_{j\in Z_{m_i}} Ai,j) = \sum_{j_0\in Z_{m_0}} \sum_{j_1\in Z_1} \dots \sum_{j_{n-1}\in Z_{m_{n-1}}} P(\bigcap_{i\in Zn} A_{i,j_i})$

If A_{i_1,j_1} and A_{i_2,j_2} are independent, $\forall i_1\neq i_2\in Z_n$, $\forall j_1\in Z_{m_{i_1}}$, $j_2\in Z_{m_{i_2}}$. We have

$P(\bigcap_{i\in zn} \bigcup_{j\in Z_{m_i}} Ai,j) = \sum_{j_0\in Z_{m_0}} \sum_{j_1\in Z_1} \dots \sum_{j_{n-1}\in Z_{m_{n-1}}} \prod_{i\in Z_n} P(A_{i,j_i})$

(aux3) Given $A_0, A_1, \dots, A_{m-1}$ are mutually independent. We have

$P(\overline{A_0} \cap \overline{A_1} \cap \dots \cap \overline{A_{m-1}}) = \prod_{i\in Z_m} P(\overline{A_i})$

(aux4) $P(\overline{A}|N) = 1 - P(A|N)$

(aux5) Given $A_0, A_1, \dots, A_{m-1}$ are mutually conditional independent given N. We have

$P(\overline{A_0} \cap \overline{A_1} \cap \dots \cap \overline{A_{m-1}}|N) = \prod_{i\in Z_m} P(\overline{A_i}|N)$

(aux6) Given $A_i \cap A_j = \emptyset$, $\forall i\neq j\in Z_m$. $\bigcup_{i\in Z_m} A_i = S$. $P(e) = p$, $\forall e\in S$. Event $e\in S$ are mutually independent. We have

$P(e\in\{a_0 \cup a_1 \cup \dots \cup a_{m-1} | a_0\in SA_1, a_1\in SA_2, \dots, a_{m-1}\in SA_{m-1}\}) = \prod_{i\in Z_m} \sum_{a_i\in SA_i} p^{|a_i|}(1-p)^{|SA_i|-|a_i|}$

(asp1) $P(\bigcap_{i\in J} PM_i) = \prod_{i\in J} P(PM_i)$, $\forall J \subseteq Z_M$

(asp2) $P(PM_i = PM1 \cap PM_i = PM2) = 0$, $\forall\, PM1 \neq PM2 \in S$

(asp3) $P(PM_i) = \frac{1}{|S|}$, $\forall i \in Z_M$

(asp4) $P(NL = NL1 \cap NL = NL2) = 0$

(asp5) $P(NL \cap (\bigcap_{i\in J} PMi)) = P(NL)P(\bigcap_{i\in J} PMi)$, $\forall\, NL \in NLS$, $J \subseteq Z_M$

(imp1) $P(\bigcap_{i\in J} PM_i \in S_i) = \prod_{i\in J} P(PM_i \in S_i) = \sum_{e_0\in S_0} \sum_{e_1\in S_1} \dots \sum_{e_{m-1}\in S_{m-1}} \prod_{i\in Z_m} P(PM_i = e_i) = \left(\frac{1}{|S|}\right)^{|J|} \prod_{i\in J} |S_i|$, $\forall J \subseteq Z_M$

(imp2) $P(NL \cap (\bigcap_{i\in J} PM_i \in S_i)) = P(NL)P(\bigcap_{i\in J} PM_i \in S_i)$, $\forall J \subseteq Z_M$

(imp3) $P(\odot) = p^{|NL|}(1-p)^{N-|NL|} \frac{1}{|S|}$

(thr1) $P(E_i\, DL|NL) = \frac{|S_{NL}|}{|S|}$

(thr2) $P(\bigcap_{i\in Z_m} E_i\, DL \mid NL) = \prod_{i\in Z_m} P(E_i\, DL|NL)$

(thr3) $P(\bigcap_{0\le i\le r} E_i\, DL \cap \bigcap_{r+1\le i\le m-1} \overline{E_i\, DL} \mid NL) = \prod_{0\le i\le r} P(E_i\, DL|NL) \prod_{r+1\le i\le m-1} P(\overline{E_i\, DL}|NL)$

(thr4) $P(E_i\, DL) = \sum_{NL\in NLS} P(NL) \frac{|S_{NL}|}{|S|} = \sum_{EL\in ELS} p^{|EL|}(1-p)^{D-|EL|}$

(thr5) $P(\bigcap_{i\in Z_m} E_i\, DL) = \sum_{NL\in NLS} P(NL) \left(\frac{|S_{NL}|}{|S|}\right)^m \neq \left(\sum_{NL\in NLS} P(NL) \frac{|S_{NL}|}{|S|}\right)^m = \prod_{i\in Z_m} P(E_i\, DL)$

(thr6) $P(DL = r) = C_M^r \sum_{NL\in NLS} P(NL) \left(\frac{|S_{NL}|}{|S|}\right)^r \left(1 - \frac{|S_{NL}|}{|S|}\right)^{M-r}$

(thr7) $P(DL) = 1 - P(DL=0) = 1 - \sum_{NL\in NLS} P(NL) \left(1 - \frac{|S_{NL}|}{|S|}\right)^M$

Reliability Limits

I want to explore what limits the reliability of super large storage clusters.

```
Basics

    E(a) := expectation of random_variable
    E(a|case) := conditional expectation of random_varable = sum(v in {a's possible values}, v*P(a=v|case))

Basic formulas

    Conditional expection: (aux7)

Notations

    λ := node failure rate = d(P(node fail in time dt))/dt = a constant
    dt := a very small time window, which is usually used with λ
        λdt = node failure possibility in time window dt

    K := K that, an extent losing l>K fragments is never recoverable, losing l<=K fragments can be recoverable
        usually, K = parity fragment count

    SR := storage overhead = D/(D-K)
    SD := user data size per extent
    TD := total user data size
    DS := fragment size = SD*SR/D = SD/(D-K)

Assumptions

    Breakdown extents by how many fragments lost: (asp6)

        Ml ("l" in subscription) := extent count of losing l fragments
        NB := network bandwidth for data recovery. note that other network bandwidth can be used for customer t
        NBl ("l" in subscription) := network bandwidth for recover extents in Ml
        rvl ("l" in subscription) := given the code schema, in all cases of losing l fragments, ratio of the re

Conclusions

    First, we study the the stable state. I.e. NBl needs to be proper to hold Ml stable.

        Extent count that flows into Mi due to node loss: (thr8)
        Extent count that flows into Mi due to recovery: (thr9)
        Extent count that flows out of Mi due to recovery: (thr10)
        Extnet count that flows out of Mi due to node loss: (thr11)

        By balancing the flow in and flow out, to maintain the stable state, we have: (thr12). Conclusions

            It is not possible to balance. Not possible even M0=K, M1..MK = 0. There is no balance state.
            If we force M0..MK-1 to balance, MK will drop. The process repeat until all MK..M1 drop to 0, survi
            If we force MK..M1 to balance, M0 will drop. The process repeats until all M0..MK-1 drop to 0, surv

            A healthy cluster can only be M0=M: (thr13)
                The other M1..MK should be infinitely small, ignorable count
```

```
Next, lets find out data loss rates

    Extent loss rate λ(ext): (thr14)
    Cluster data loss rate λ(DL): (thr15)
        λ(ext) and λ(DL) are the solo result of Mi states. They has not relation with NBi. While NBi is wha
        When M0=M, λ(ext) and λ(DL) are the best of what they can get from NBi.

    Practical markov for single extent loss probability: (thr16)
        Notations
            Let NBSi := the available network recovery bandwidth for this single extent, when it has lost i
                The bandwidth is affacted by repair priority, TOR and node NIC bandwidth.
            P(EDL Mi) := probability of lose this extent when it is at Mi
                P(EDL M0) is what we try to calculate
        Note that this formula not math strict, because in a absorbing markov chain the extent loss probabi
            https://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/Chapter11.pdf
        But we are here to calculate the "practical" probability
            This formula assumes we follow the repair lifecycle of the given extent
            While the above λ(ext) assumes we take any extent from the cluster, which has M1-MK count is in
        This formula is useful because
            It takes network recovery bandwidth into consideration, very practical
            It follows the entire repair lifecycle of the given extent

Conclusions

    To maintain cluster healthy, recovery bandwidth NB shoud be no less than λ*TD*SR.
        The necessary recovery bandwidth growth with total user data size, code schema storage overhead, bu
    Recovery bandwidth NB limits how much data can be there in a super large storage cluster
        λ and code schema storage overhead multiply it by factors
        Cluster node count has no relation with NB. But user data size does.

    For customer-faced reliability, extent loss rate λ(ext) should be low. It is the direct result λ and co
    For operation ease, cluster data loss λ(DL) should be low. It is the direct result of λ and code schema
        λ(ext) and λ(DL) have no direct relation ship with NB. But NB is necessary to pull extents from M1.
        If M1..MK != 0, the λ(ext)dt and λ(DL)dt raise by factor of 1/(λdt)
```

Below are the formulas stated above

(aux7) Given $B_i \cap B_j = \emptyset, \forall i \neq j \in Z_m$, and $\bigcup_{i\in Z_m} B_i = I$

$E(a) = \sum_{v\in\{a's\ possible\ values\}} v * P(a = v) = \sum_{i\in Z_m} P(B_i)E(a|B_i)$

(asp6) $M = M_0 + M_1 + \cdots + M_l + \cdots + M_D; M_l = 0, \forall l > K$

$NB = NB_0 + NB_1 + \cdots + NB_l + \cdots + NB_D; NB_0 = 0; NB_l = 0, \forall l > K; \forall l$, if $M_l = 0$, then $NB_l = 0$

$rv = [rv_0, rv_1, \dots, rv_l, \dots, rv_D]; rv_0 = 1; rv_l = 0, \forall l > K; rv_{l+1} < rv_l, \forall l$

(thr8) $CntExt\_LeftFlowIn_i = rv_i \sum_{|NL|=0}^{N} \sum_{l=0}^{i-1} (\lambda dt)^{|NL|} (1 - \lambda dt)^{N-|NL|} * M_l * C_{D-l}^{i-l} C_{N-(D-l)}^{|NL|-(i-l)}$

$\approx rv_i * \lambda dt * M_{i-1} * (D - i + 1)$

(thr9) $CntExt\_RightFlowIn_i = \sum_{l=i+1}^{D} M_l * \prod_{j=i+1}^{l} \frac{NB_j}{M_j * DS} * (dt)^{l-i}$

$\approx \frac{NB_{i+1}}{DS} dt$

(thr10) $CntExt\_LeftFlowOut_i = \frac{NB_i}{DS} dt$

(thr11) $CntExt\_RightFlowOut_i = M_i \sum_{|NL|=0}^{N} (\lambda dt)^{|NL|} (1 - \lambda dt)^{N-|NL|} (C_N^{|NL|} - C_{N-(D-i)}^{|NL|})$

$\approx M_i * \lambda dt * (D - i)$

(thr12) $CntExt\_LeftFlowIn_i + CntExt\_RightFlowIn_i = CntExt\_LeftFlowOut_i + CntExt\_RightFlowOut_i$, $\forall i=0..K$

$$\Rightarrow \left.\begin{array}{l} \left\{\begin{array}{l} \frac{NB_k}{DS} * \frac{1}{\lambda D} = rv_k * M_{k-1} * \left(1 - \frac{k-1}{D}\right) - M_k * \left(1 - \frac{K}{D}\right) \\ \frac{NB_i - NB_{i+1}}{DS} * \frac{1}{\lambda D} = rv_i * M_{i-1} * \left(1 - \frac{i-1}{D}\right) - M_i * \left(1 - \frac{i}{D}\right) \end{array}\right. \\ \text{with } \frac{NB_0}{DS} * \frac{1}{\lambda D} = 0 \quad \text{and} \quad \forall l, \text{ if } M_l = 0, \text{ then } NB_l = 0 \end{array}\right\} \Rightarrow$$

Let $b_i = \frac{NB_i}{DS} * \frac{1}{\lambda D}, a_i = M_i * (1 - \frac{i}{D})$

$$\Rightarrow \left\{\begin{array}{l} b_k = rv_k a_{k-1} - a_k \\ b_i - b_{i+1} = rv_i a_{i-1} - a_i \\ b_0 = 0 \\ \forall l, if\ a_l = 0\ then\ b_l = 0 \end{array}\right\} \Rightarrow$$

If start from $b_k = rv_k a_{k-1} - a_k$, we get $b_{i,from\ b_k} = rv_i a_{i-1} - \sum_{l=i+1}^{K} (1 - rv_l) a_{l-1} - a_k$, then

$b_{0,from\ b_k} - b_0 = -\left(\sum_{l=1}^{K} (1 - rv_l) a_{l-1} + a_k\right) \leq 0$

If start from $b_0 = 0$, we get $b_{i,from\ b_0} = \sum_{l=1}^{i-1} (1 - rv_l) a_{l-1} + a_{i-1}$, then

$b_{k,from\ b_0} - b_k = \sum_{l=1}^{K} (1 - rv_l) a_{l-1} + a_k \geq 0$

If we balance the equation group, then we get

$M_l = 0, NB_l = 0, \forall l = 0..K$ (note that l=0 is included)

(thr13) $M_0 = M$

$\frac{NB}{DS} dt = M_0 * \lambda dt * D \Rightarrow NB = \lambda * D * DS * M = \lambda * M * SD * SR = \lambda * TD * SR$

(thr14) $\lambda_{ext}dt = \sum_{i=0}^{D} \frac{M_i}{M} \sum_{l=1}^{D-i} C_{D-i}^{l} (\lambda dt)^l (1-\lambda dt)^{(D-i)-l} (1 - rv_{i+l}) \approx \lambda D * dt * \sum_{i=0}^{D} \frac{M_i}{M} (1 - \frac{i}{D})(1 - rv_{i+1})$

$\lambda_{ext}dt = \sum_{l=1}^{D} C_D^l (\lambda dt)^l (1-\lambda dt)^{D-l} (1 - rv_l), if\ M_0 = M$

(thr15) $\lambda_{DL}dt \approx 1 - \sum_{|NL|=0}^{N} C_N^{|NL|} (\lambda dt)^{|NL|} (1-\lambda dt)^{N-|NL|} \prod_{i=0}^{D} (\sum_{l=1}^{D-i} \frac{C_{D-i}^{l} C_{N-(D-i)}^{|NL|-l}}{C_N^{|NL|}} rv_{i+l} + \frac{C_{D-i}^{0} C_{N-(D-i)}^{|NL|}}{C_N^{|NL|}})^{M_i}$

// The above uses "≈" because I averaged ∏ extent loss probability by rv

$\approx \lambda * N * dt * (1 - \prod_{i=0}^{D} (1 - \frac{D-i}{N} (1 - rv_{i+1}))^{M_i})$

$\lambda_{DL}dt \approx 1 - \sum_{|NL|=0}^{N} C_N^{|NL|} (\lambda dt)^{|NL|} (1-\lambda dt)^{N-|NL|} \left(\sum_{l=0}^{D} \frac{C_D^l C_{N-D}^{|NL|-l}}{C_N^{|NL|}} rv_l\right)^M, if\ M_0 = M$

(thr16) $P(EDL\ M_i) = \sum_{l=i+1}^{D} C_{D-i}^{l-i} \left(\lambda \frac{NBS_i}{DS}\right)^{l-i} \left(1 - \lambda \frac{NBS_i}{DS}\right)^{D-l} \left((1 - rv_l) + rv_l P(EDL\ M_l)\right)$

$P(EDL\ M_K) = 1 - \left(1 - \lambda \frac{NBS_K}{DS}\right)^{D-K}$

Reliability framework

To summarize the technique framework to ensure storage reliability

```
* Data input: ensure the reliability when data is coming in
    * End-to-end verification: ensure what user writes is what is stored
    * Commit safe: when user is ack-ed, ensure the data is guaranteed persistent

* Data storing: ensure the reliability at where data is stored
    * Good data should not be overwriten. In case a bug or at least we can recover data from history
    * Data recovery
        * Detect data loss as fast as you can
            * On-the-fly, when user read data, from node loss, from disk failure, etc. Summarize whatever inf
            * Prediction, from unstable/unhealthy/untrustable nodes that are going to loss, from disks that f
        * Repair action should be prioritized
            * Repair the data at high risk first.
            * Repair actions should be well-scheduled and well-managed.
            * The overall reliability is determined by how fast you can repair.
        * Watch out that your data is more vulnerable when server upgrading, because
            * Some portion of data is unavailable
            * The user traffic becomes larger because of reconstruction read
            * The repair traffic also becomes larger
        * Code schema (e.g. 3-replica, erasure-coding, LRC) is very important for data reliability.
            * It is also important for balancing reliability with overhead of storage, bandwidth, latency, re
            * Policied auto transition between different code schemas may be necessary.
    * Silent data corruption: This is a real problem. Background check and repair can help; but too time/reso
    * Some piece hardware from vendors can have problem. Some firmware can have problem, go wrong, or have bu
    * Old clusters are highly risked: old devices are easy to fail, and in burst, which generates a lot of da

* Data serving
    * Data availability and data durability are different. Good data serving ensures data availability
    * Nodes can be transiently unavailable. Detect transient and permanent before you head to expensive data
    * Node failure can usually be repaired quickly by a reboot. Node can jump online and offline. Be careful
    * Customer impact is different, i.e. even data is lost, if we are in the lucky time window where customer

* Raw disk recovery: when everything is goine, we expect to extract user data from a raw disk
    * Make sure you have a way to know the raw binary format on disk. Extract data from raw disk can be the f
    * Make sure the necessary metadata for raw disk recovery can be obtained and well protected.

* Hardware: hardware failure rate has direct impact on everything about reliability, very straightforward. Be
```

Other references

- DataDomain DIA (http://www.emc.com/collateral/software/white-papers/h7219-data-domain-data-invul-arch-wp.pdf)

storage 22 (/categories.html#storage-ref)

storage 32 (/tags.html#storage-ref) reliability 7 (/tags.html#reliability-ref)

cloud 15 (/tags.html#cloud-ref)

Create an Issue () or comment below