AWS Livre blanc

Disponibilité et au-delà : comprendre et améliorer la résilience des systèmes distribués sur AWS

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Disponibilité et au-delà : comprendre et améliorer la résilience des systèmes distribués sur AWS: AWS Livre blanc

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques et la présentation commerciale d’Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n’est pas d’Amazon, d’une manière susceptible de créer une confusion parmi les clients, ou d’une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d’Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Résumé et introduction i
Introduction 1
Comprendre la disponibilité 2
Disponibilité du système distribué 4
Types de défaillances dans les systèmes distribués 6
Disponibilité avec dépendances 7
Disponibilité avec redondance 9
Théorème CAP 13
Tolérance aux pannes et isolation des pannes 14
Mesurer la disponibilité 17
Taux de réussite des demandes côté serveur et côté client 17
Indisponibilité annuelle 20
Latence 21
Conception de systèmes distribués à haute disponibilité sur AWS 22
Réduire MTTD 22
Réduire MTTR 23
Parcours pour contourner l'échec 24
Retour à un état connu en bon état 26
Diagnostic des défaillances 28
Runbooks et automatisation 28
En augmentation MTBF 28
Élargir le système distribué MTBF 28
Dépendance croissante MTBF 31
Réduire les sources d'impact communes 32
Conclusion 36
Annexe 1 — Métriques critiques du MTTD et du MTTR 38
Collaborateurs 39
Suggestions de lecture 40
Historique de document 41
Avis 42
Glossaire AWS 43
..... xliv

Disponibilité et au-delà : comprendre et améliorer la résilience des systèmes distribués sur AWS

Date de publication : 12 novembre 2021 (Historique de document)

Aujourd'hui, les entreprises exploitent des systèmes complexes et distribués à la fois dans le cloud et sur site. Ils souhaitent que ces charges de travail soient résilientes afin de servir leurs clients et d'atteindre leurs objectifs commerciaux. Ce document décrit une compréhension commune de la disponibilité en tant que mesure de la résilience, établit des règles pour créer des charges de travail hautement disponibles et propose des conseils sur la manière d'améliorer la disponibilité des charges de travail.

Introduction

Qu'est-ce que cela signifie de créer une charge de travail hautement disponible ? Comment mesurez-vous la disponibilité ? Que puis-je faire pour augmenter la disponibilité de ma charge de travail ? Ce document vous aidera à répondre à ce type de questions. Il est divisé en trois sections principales. La première section, intitulée Comprendre la disponibilité, est essentiellement théorique. Il établit une compréhension commune de la définition de la disponibilité et des facteurs qui influent sur celle-ci. La deuxième section, Mesurer la disponibilité, fournit des conseils pour mesurer de manière empirique la disponibilité de votre charge de travail. La troisième section, Concevoir des systèmes distribués à haut niveau de disponibilité sur, AWS est une application pratique des idées présentées dans la première section. De plus, dans ces sections, ce document définira les règles permettant de créer des charges de travail résilientes. Ce document est destiné à étayer les directives et les meilleures pratiques présentées dans le pilier de fiabilité AWS bien architecturé.

Tout au long de cet article, vous rencontrerez de nombreuses mathématiques algébriques. Les principaux points à retenir sont les concepts que ces mathématiques soutiennent, et non les mathématiques elles-mêmes. Cela dit, le présent document vise également à présenter un défi. Lorsque vous gérez des charges de travail à haut niveau de disponibilité, vous devez être en mesure de prouver mathématiquement que ce que vous avez créé atteint vos objectifs. Même les meilleures conceptions fondées sur de bonnes intentions peuvent ne pas toujours atteindre le résultat souhaité. Cela signifie que vous avez besoin de mécanismes qui mesurent l'efficacité de la solution. Par conséquent, un certain niveau de calcul est nécessaire pour créer et exploiter des systèmes distribués résilients et hautement disponibles.

Comprendre la disponibilité

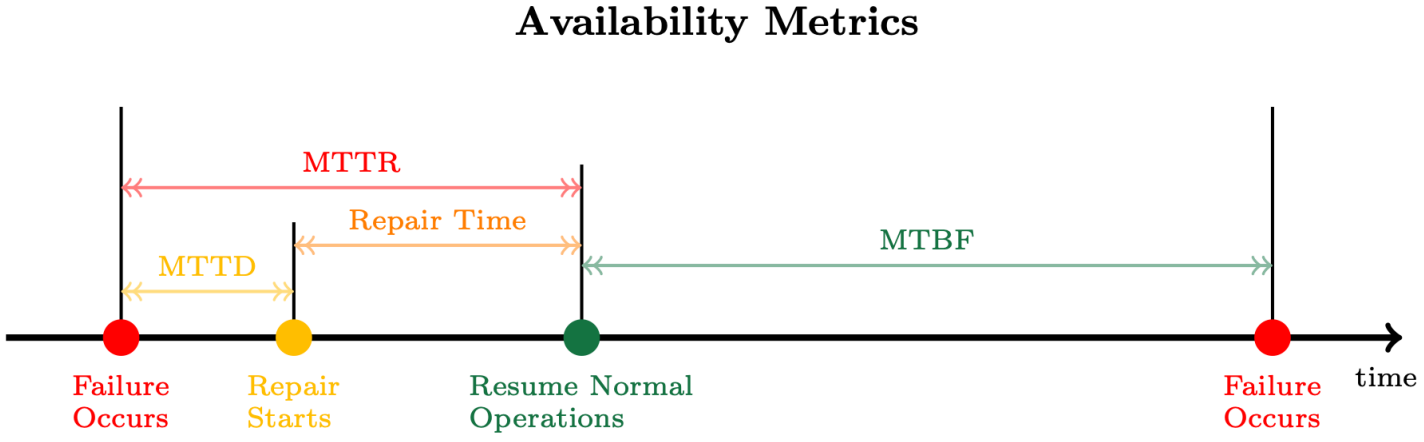
La disponibilité est l'un des principaux moyens de mesurer quantitativement la résilience. Nous définissons la disponibilité, A, comme le pourcentage de temps pendant lequel une charge de travail est disponible pour être utilisée. Il s'agit d'un ratio entre son « temps de disponibilité » attendu (disponibilité) et le temps total mesuré (le « temps de disponibilité » attendu plus le « temps d'arrêt » attendu).

$$A = \frac{uptime}{uptime + downtime}$$

Équation 1 - Disponibilité

Pour mieux comprendre cette formule, nous verrons comment mesurer le temps de disponibilité et les temps d'arrêt. Tout d'abord, nous voulons savoir combien de temps durera la charge de travail sans échec. C'est ce que nous appelons le temps moyen entre les défaillances (MTBF), le temps moyen entre le début du fonctionnement normal d'une charge de travail et sa prochaine défaillance. Ensuite, nous voulons savoir combien de temps il faudra pour récupérer après une panne.

Nous appelons ce délai moyen de réparation (ou de restauration) (MTTR), une période pendant laquelle la charge de travail n'est pas disponible pendant que le sous-système défaillant est réparé ou remis en service. Une période importante du MTTR est le délai moyen de détection (MTTD), c'est-à-dire le laps de temps entre la survenue d'une panne et le début des opérations de réparation. Le schéma suivant montre comment toutes ces mesures sont liées.



La relation entre MTTD, MTTR et MTBF

Nous pouvons ainsi exprimer la disponibilité, A, en utilisant le MTBF, le temps pendant lequel la charge de travail est en hausse, et MTTR, le temps pendant lequel la charge de travail est réduite.

$$A = \frac{MTBF}{MTBF + MTTR}$$

Équation 2 - Relation entre le MTBF et le MTTR

Et la probabilité que la charge de travail soit « réduite » (c'est-à-dire non disponible) est la probabilité d'échec, F.

$$F = 1 - A$$

Équation 3 - Probabilité de défaillance

La fiabilité est la capacité d'une charge de travail à faire ce qu'il faut, lorsque cela est demandé, dans le délai de réponse spécifié. C'est ce que mesure la disponibilité. Le fait qu'une charge de travail tombe en panne moins fréquemment (MTBF plus long) ou que le temps de réparation soit plus court (MTTR plus court) améliore sa disponibilité.

> Règle 1
>
> Des défaillances moins fréquentes (MTBF plus long), des temps de détection des défaillances plus courts (MTTD plus court) et des temps de réparation plus courts (MTTR plus court) sont les trois facteurs utilisés pour améliorer la disponibilité dans les systèmes distribués.

Rubriques

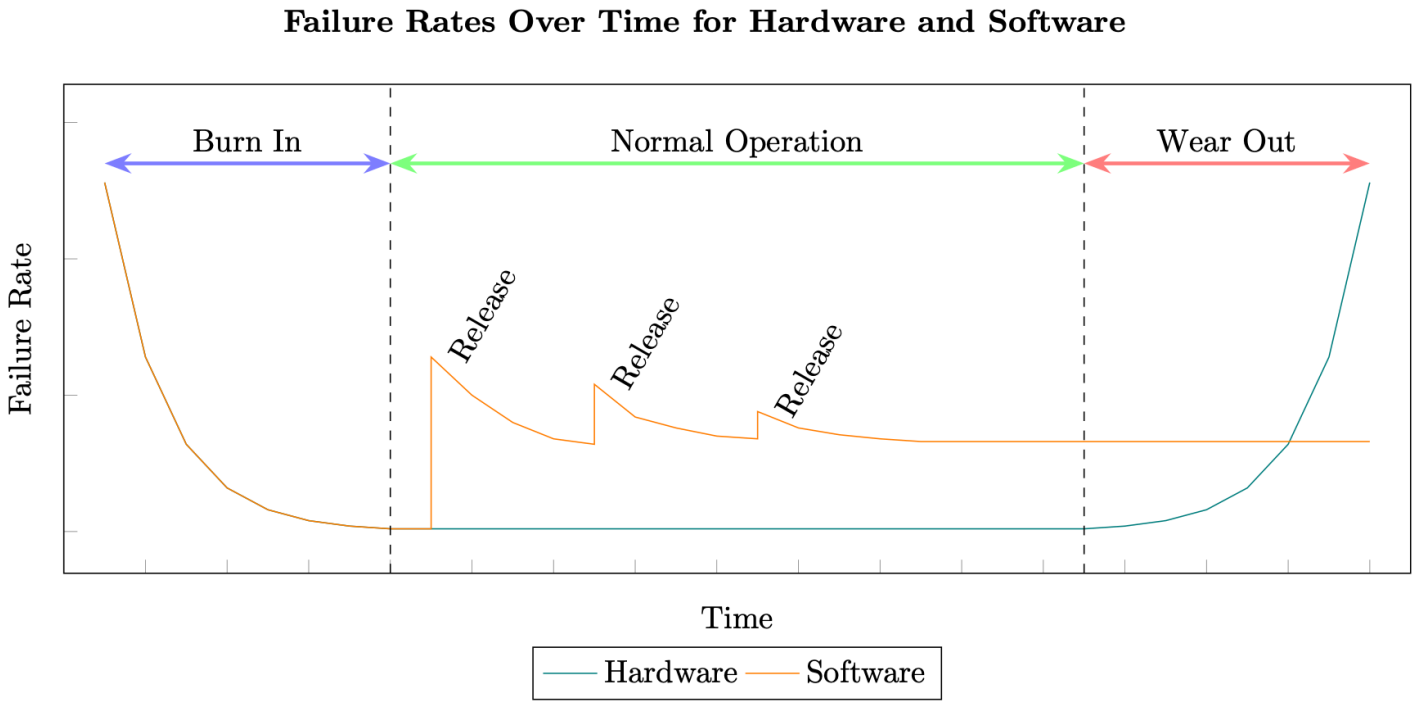
- Disponibilité du système distribué
- Disponibilité avec dépendances
- Disponibilité avec redondance
- Théorème CAP

- Tolérance aux pannes et isolation des pannes

Disponibilité du système distribué

Les systèmes distribués sont composés à la fois de composants logiciels et de composants matériels. Certains composants logiciels peuvent eux-mêmes être un autre système distribué. La disponibilité des composants matériels et logiciels sous-jacents influe sur la disponibilité de votre charge de travail qui en résulte.

Le calcul de la disponibilité à l'aide du MTBF et du MTTR trouve ses racines dans les systèmes matériels. Cependant, les systèmes distribués échouent pour des raisons très différentes de celles d'un élément matériel. Lorsqu'un fabricant peut calculer régulièrement le délai moyen avant qu'un composant matériel ne s'use, les mêmes tests ne peuvent pas être appliqués aux composants logiciels d'un système distribué. Le matériel suit généralement la courbe « baignoire » du taux de défaillance, tandis que le logiciel suit une courbe échelonnée produite par des défauts supplémentaires introduits à chaque nouvelle version (voir Fiabilité des logiciels).



Taux de défaillance du matériel et des logiciels

En outre, les logiciels des systèmes distribués évoluent généralement à un rythme exponentiellement supérieur à celui du matériel. Par exemple, un disque dur magnétique standard peut avoir un taux de défaillance annualisé (AFR) moyen de 0,93 %, ce qui, dans la pratique, peut se traduire par une

durée de vie d'au moins 3 à 5 ans avant qu'il n'atteigne la période d'usure, voire plus longtemps (voir Backblaze Hard Drive Data and Stats, 2020). Le disque dur ne change pas de manière significative au cours de cette durée de vie, où, dans 3 à 5 ans, par exemple, Amazon pourrait déployer plus de 450 à 750 millions de modifications dans ses systèmes logiciels. (Voir Amazon Builders' Library — Automatiser des déploiements sécurisés et sans intervention directe.)

Le matériel est également soumis au concept d'obsolescence planifiée, c'est-à-dire qu'il a une durée de vie intrinsèque et qu'il devra être remplacé après un certain temps. (Voir The Great Lightbulb Conspiracy.) En théorie, le logiciel n'est pas soumis à cette contrainte, il n'a pas de période d'usure et peut être utilisé indéfiniment.

Tout cela signifie que les mêmes modèles de test et de prédiction utilisés pour le matériel pour générer les numéros MTBF et MTTR ne s'appliquent pas aux logiciels. Des centaines de tentatives ont été faites pour créer des modèles pour résoudre ce problème depuis les années 1970, mais elles se répartissent généralement en deux catégories : la modélisation prédictive et la modélisation d'estimation. (Voir la liste des modèles de fiabilité des logiciels.)

Ainsi, le calcul d'un MTBF et d'un MTTR prospectifs pour les systèmes distribués, et donc d'une disponibilité prospective, sera toujours dérivé d'un certain type de prédiction ou de prévision. Ils peuvent être générés par le biais d'une modélisation prédictive, d'une simulation stochastique, d'une analyse historique ou de tests rigoureux, mais ces calculs ne constituent pas une garantie de disponibilité ou d'indisponibilité.

Les raisons pour lesquelles un système distribué a échoué dans le passé peuvent ne jamais se reproduire. Les raisons pour lesquelles il échouera à l'avenir seront probablement différentes et peut-être inconnaissables. Les mécanismes de restauration requis peuvent également être différents de ceux utilisés dans le passé pour les défaillances futures et prendre des durées très différentes.

De plus, le MTBF et le MTTR sont des moyennes. Il y aura un certain écart entre la valeur moyenne et les valeurs réelles observées (l'écart type, σ, mesure cette variation). Ainsi, les charges de travail peuvent s'écouler plus ou moins longtemps entre les défaillances et les temps de restauration dans le cadre d'une utilisation réelle en production.

Cela dit, la disponibilité des composants logiciels qui constituent un système distribué est toujours importante. Les logiciels peuvent échouer pour de nombreuses raisons (abordées plus en détail dans la section suivante) et ont un impact sur la disponibilité de la charge de travail. Ainsi, pour les systèmes distribués à haute disponibilité, il convient d'accorder autant d'importance au calcul, à la mesure et à l'amélioration de la disponibilité des composants logiciels qu'aux sous-systèmes matériels et logiciels externes.

Règle 2

La disponibilité du logiciel dans votre charge de travail est un facteur important de la disponibilité globale de votre charge de travail et doit bénéficier de la même attention que les autres composants.

Il est important de noter que même si le MTBF et le MTTR sont difficiles à prévoir pour les systèmes distribués, ils fournissent tout de même des informations clés sur la manière d'améliorer la disponibilité. La réduction de la fréquence des défaillances (MTBF plus élevé) et la diminution du temps de rétablissement après une défaillance (MTTR plus faible) permettront toutes deux d'améliorer la disponibilité empirique.

Types de défaillances dans les systèmes distribués

Il existe généralement deux catégories de bogues dans les systèmes distribués qui affectent la disponibilité, appelées affectueusement Bohrbug et Heisenbug (voir « A Conversation with Bruce Lindsay », ACM Queue vol. 2, n° 8 — novembre 2004).

Un bohrbug est un problème logiciel fonctionnel récurrent. Avec la même entrée, le bogue produira systématiquement la même sortie incorrecte (comme le modèle atomique déterministe de Bohr, qui est solide et facile à détecter). Ces types de bogues sont rares au moment où une charge de travail passe en production.

Un Heisenbug est un bogue transitoire, c'est-à-dire qu'il ne se produit que dans des conditions spécifiques et peu communes. Ces conditions sont généralement liées à des éléments tels que le matériel (par exemple, une défaillance passagère du périphérique ou des spécificités de l'implémentation matérielle, comme la taille du registre), les optimisations du compilateur et l'implémentation du langage, les conditions limites (par exemple, un manque de stockage temporaire) ou les conditions de course (par exemple, ne pas utiliser de sémaphore pour les opérations multithread).

Les Heisenbugs constituent la majorité des bogues en production et sont difficiles à détecter car ils sont insaisissables et semblent changer de comportement ou disparaître lorsque vous essayez de les observer ou de les corriger. Toutefois, si vous redémarrez le programme, l'opération échouée sera probablement couronnée de succès car l'environnement d'exploitation est légèrement différent, éliminant ainsi les conditions à l'origine du Heisenbug.

Ainsi, la plupart des défaillances de production sont transitoires et il est peu probable qu'elle échoue à nouveau lorsque l'opération est tentée à nouveau. Pour être résilients, les systèmes distribués doivent être tolérants aux défaillances face aux Heisenbugs. Nous verrons comment y parvenir dans la section Augmenter le MTBF des systèmes distribués.

Disponibilité avec dépendances

Dans la section précédente, nous avons indiqué que le matériel, les logiciels et éventuellement d'autres systèmes distribués font tous partie de votre charge de travail. Nous appelons ces composants des dépendances, c'est-à-dire les éléments dont dépend votre charge de travail pour fournir ses fonctionnalités. Il existe des dépendances matérielles, c'est-à-dire des éléments sans lesquels votre charge de travail ne peut fonctionner, et des dépendances souples dont l'indisponibilité peut passer inaperçue ou être tolérée pendant un certain temps. Les dépendances strictes ont un impact direct sur la disponibilité de votre charge de travail.

Nous pouvons essayer de calculer la disponibilité maximale théorique d'une charge de travail. C'est le produit de la disponibilité de toutes les dépendances, y compris le logiciel lui-même (α_n est la disponibilité d'un seul sous-système) car chacune d'entre elles doit être opérationnelle.

$$A = \alpha_1 \times \alpha_2 \times \ldots \times \alpha_n$$

Équation 4 - Disponibilité maximale théorique

Les chiffres de disponibilité utilisés dans ces calculs sont généralement associés à des éléments tels que les SLA ou les objectifs de niveau de service (SLO). Les SLA définissent le niveau de service attendu des clients, les indicateurs permettant de mesurer le service, ainsi que les mesures correctives ou les pénalités (généralement pécuniaires) en cas de non-atteinte des niveaux de service.

En utilisant la formule ci-dessus, nous pouvons conclure que, d'un point de vue purement mathématique, une charge de travail ne peut être plus disponible que n'importe laquelle de ses dépendances. Mais en réalité, ce que nous constatons généralement, c'est que ce n'est pas le cas. Une charge de travail créée à l'aide de deux ou trois dépendances avec des SLA de disponibilité de 99,99 % peut toujours atteindre elle-même une disponibilité de 99,99 %, voire plus.

En effet, comme nous l'avons indiqué dans la section précédente, ces chiffres de disponibilité sont des estimations. Ils estiment ou prédisent la fréquence d'une panne et la rapidité avec laquelle elle

peut être réparée. Ils ne constituent pas une garantie de temps d'arrêt. Les dépendances dépassent souvent leur niveau de disponibilité (SLA ou SLO) déclaré.

Les dépendances peuvent également avoir des objectifs de disponibilité interne plus élevés, par rapport auxquels elles ciblent les performances, que les chiffres fournis dans les SLA publics. Cela permet d'atténuer les risques liés au respect des SLA lorsque l'inconnu ou l'inconnaissable se produit.

Enfin, votre charge de travail peut comporter des dépendances dont les SLA ne peuvent pas être connus ou ne proposent pas de SLA ou de SLO. Par exemple, le routage Internet mondial est une dépendance courante pour de nombreuses charges de travail, mais il est difficile de savoir à quel (s) fournisseur (s) de services Internet votre trafic mondial fait appel, s'ils ont des SLA et dans quelle mesure ils sont cohérents entre les fournisseurs.

Tout cela nous indique que le calcul d'une disponibilité théorique maximale n'est susceptible de produire qu'un calcul approximatif de l'ordre de grandeur, mais qu'il n'est probablement pas en soi précis ou ne fournit pas d'informations significatives. Ce que les calculs nous indiquent, c'est que moins votre charge de travail repose sur des éléments sur lesquels repose votre charge de travail, plus le risque global de défaillance est réduit. Moins il y a de nombres inférieurs à un multipliés, plus le résultat est élevé.

> **Règle 3**
>
> La réduction des dépendances peut avoir un impact positif sur la disponibilité.

Les calculs aident également à éclairer le processus de sélection des dépendances. Le processus de sélection influe sur la manière dont vous concevez votre charge de travail, sur la manière dont vous tirez parti de la redondance des dépendances pour améliorer leur disponibilité, et sur le fait de savoir si vous considérez ces dépendances comme souples ou strictes. Les dépendances susceptibles d'avoir un impact sur votre charge de travail doivent être choisies avec soin. La règle suivante fournit des indications sur la manière de procéder.

> **Règle 4**
>
> En général, sélectionnez les dépendances dont les objectifs de disponibilité sont égaux ou supérieurs aux objectifs de votre charge de travail.

Disponibilité avec redondance

Lorsqu'une charge de travail utilise plusieurs sous-systèmes indépendants et redondants, elle peut atteindre un niveau de disponibilité théorique plus élevé qu'en utilisant un seul sous-système. Prenons l'exemple d'une charge de travail composée de deux sous-systèmes identiques. Il peut être complètement opérationnel si le sous-système 1 ou le sous-système 2 est opérationnel. Pour que l'ensemble du système soit hors service, les deux sous-systèmes doivent être hors service en même temps.

Si la probabilité de défaillance d'un sous-système est de 1 − α, la probabilité que deux sous-systèmes redondants soient en panne en même temps est le produit de la probabilité de défaillance de chaque sous-système, $F = (1- \alpha_1) \times (1- \alpha)._2$ Pour une charge de travail comportant deux sous-systèmes redondants, l'équation (3) permet d'obtenir une disponibilité définie comme suit :

$$A = 1 - F$$
$$F = (1 - \alpha_1) \times (1 - \alpha_2)$$
$$A = 1 - (1 - \alpha)^2$$

Équation 5

Ainsi, pour deux sous-systèmes dont la disponibilité est de 99 %, la probabilité que l'un tombe en panne est de 1 % et la probabilité que les deux échouent est de (1− 99 %) × (1− 99 %) = 0,01 %. Cela porte la disponibilité à 99,99 % à l'aide de deux sous-systèmes redondants.

Cela peut être généralisé pour intégrer également des pièces de rechange redondantes supplémentaires. Dans l'équation (5), nous n'avons supposé qu'une seule réserve, mais une charge de travail peut en comporter deux, trois ou plus, de sorte qu'elle puisse survivre à la perte simultanée de plusieurs sous-systèmes sans affecter la disponibilité. Si une charge de travail comporte trois sous-systèmes et que deux sont des sous-systèmes de rechange, la probabilité que les trois sous-systèmes tombent en panne en même temps est de $(1- \alpha) \times (1- \alpha) \times (1- \alpha)$ ou $(1- \alpha)^3$. En général, une charge de travail avec s spare échouera uniquement si les sous-systèmes s + 1 tombent en panne.

Pour une charge de travail comportant n sous-systèmes et s pièces de rechange, f est le nombre de modes de défaillance ou la manière dont s + 1 sous-systèmes peuvent tomber en panne sur n.

Il s'agit en fait du théorème binomial, du calcul combinatoire qui consiste à choisir k éléments dans un ensemble de n, ou « n choisit k ». Dans ce cas, k est s + 1.

$$k = s + 1$$

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

$$\binom{n}{s+1} = \frac{n!}{(s+1)!\,\big(n-(s+1)\big)!}$$

$$f = \frac{n!}{(s+1)!\,(n-s-1)!}$$

Équation 6

Nous pouvons ensuite produire une approximation de disponibilité généralisée qui intègre le nombre de modes de défaillance et le nombre de modes d'épargne. (Pour comprendre pourquoi il s'agit d'une approximation, reportez-vous à l'annexe 2 de Highleyman, et al. Briser la barrière de disponibilité.)

$$s = \ Number\ of\ spares$$
$$\alpha = Availability\ of\ subcomponent$$
$$f = Number\ of\ failure\ modes$$
$$A = 1 - F \approx 1 - f(1-\alpha)^{s+1}$$

Équation 7

L'épargne peut être appliquée à toute dépendance fournissant des ressources défaillantes de manière indépendante. Les instances Amazon EC2 situées dans différentes zones de disponibilité ou les compartiments Amazon S3 situés dans des zones différentes en Régions AWS sont des exemples. L'utilisation de pièces de rechange permet à cette dépendance d'atteindre une disponibilité totale plus élevée afin de répondre aux objectifs de disponibilité de la charge de travail.

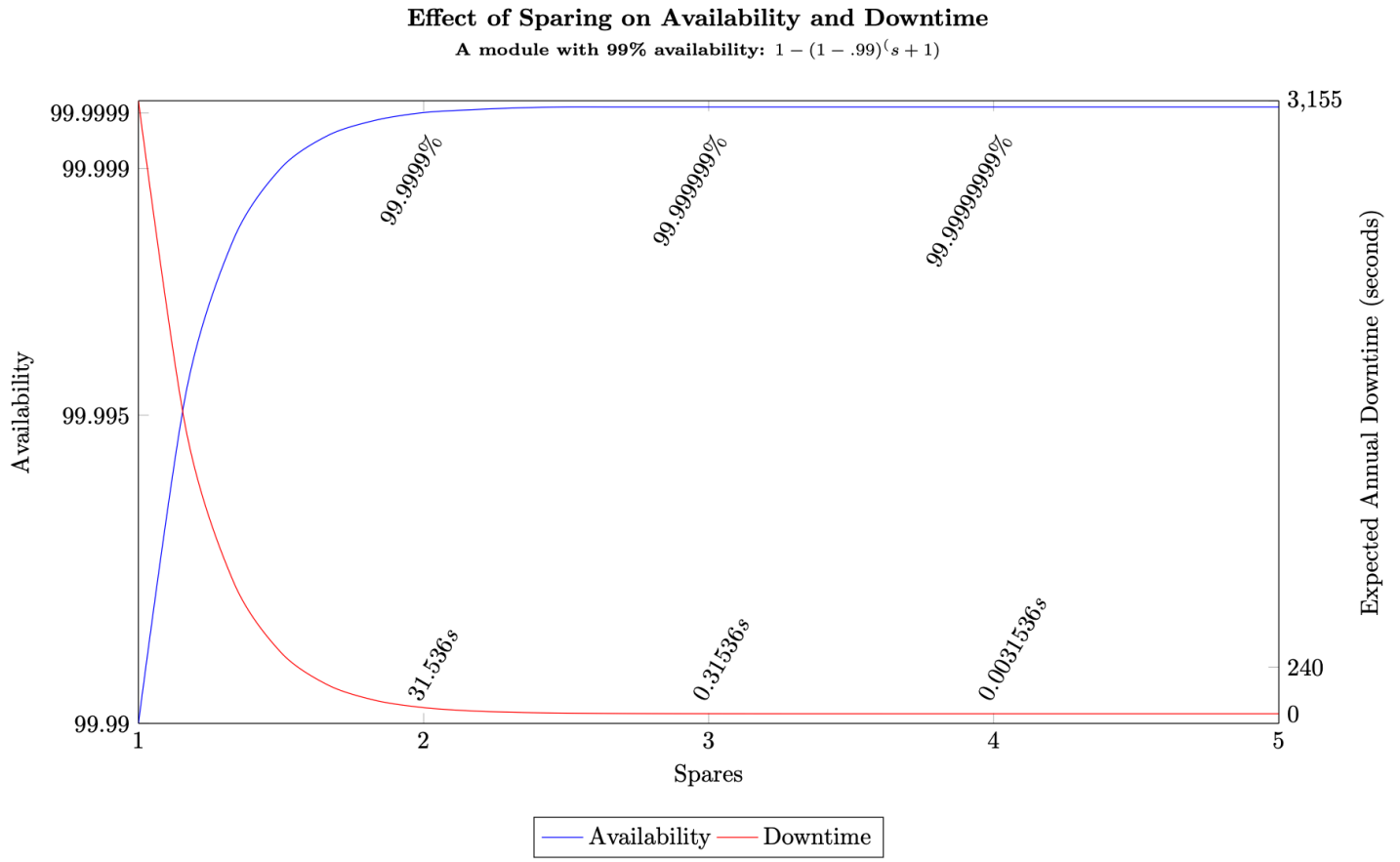
Règle 5

Utilisez l'épargne pour augmenter la disponibilité des dépendances dans une charge de travail.

Cependant, l'épargne a un coût. Chaque pièce de rechange supplémentaire coûte le même prix que le module d'origine, le coût étant au moins linéaire. La création d'une charge de travail pouvant utiliser des pièces de rechange augmente également sa complexité. Il doit savoir comment identifier les défaillances liées à la dépendance, affecter le travail à une ressource saine et gérer la capacité globale de la charge de travail.

La redondance est un problème d'optimisation. Si le nombre de pièces de rechange est insuffisant, la charge de travail risque d'échouer plus fréquemment que prévu, le nombre de pièces de rechange est trop élevé et l'exécution de la charge de travail est trop coûteuse. Il existe un seuil à partir duquel l'ajout de pièces de rechange coûtera plus cher que la disponibilité supplémentaire pour laquelle elles sont garanties.

En utilisant notre formule de disponibilité générale avec des pièces de rechange, équation (7), pour un sous-système dont la disponibilité est de 99,5 %, avec deux pièces de rechange, la disponibilité de la charge de travail est $A \approx 1 - (1)\ (1-0{,}995)^{3} =$ 99,9999875 % (environ 3,94 secondes d'indisponibilité par an), et avec 10 pièces de rechange, nous obtenons $A \approx 1 - (1)\ (1-0{,}995)^{11} = 25{,}5\ 9'$s (environ le temps d'arrêt serait de $1{,}26252 \times 10^{-15}$ m s par an, soit effectivement 0). En comparant ces deux charges de travail, nous avons multiplié par 5 le coût des pièces de rechange, ce qui nous a permis de réduire de quatre secondes les temps d'arrêt par an. Pour la plupart des charges de travail, l'augmentation des coûts ne serait pas justifiée pour cette augmentation de la disponibilité. La figure suivante illustre cette relation.



Rendements décroissants en raison de l'augmentation de l'épargne

À trois pièces de rechange et plus, le résultat est une fraction de seconde d'indisponibilité prévue par an, ce qui signifie qu'après ce point, vous atteignez la zone des rendements décroissants. Il peut y avoir une envie de « simplement en ajouter » pour atteindre des niveaux de disponibilité plus élevés, mais en réalité, le rapport coût-avantage disparaît très rapidement. L'utilisation de plus de trois pièces de rechange n'apporte aucun gain matériel et notable pour presque toutes les charges de travail lorsque le sous-système lui-même est disponible à au moins 99 %.

> **Règle 6**
>
> Il existe une limite supérieure à la rentabilité de l'épargne. Utilisez le moins de pièces de rechange nécessaires pour atteindre la disponibilité requise.

Vous devez tenir compte de l'unité de défaillance lorsque vous sélectionnez le nombre correct de pièces de rechange. Par exemple, examinons une charge de travail qui nécessite 10 instances EC2 pour gérer la capacité maximale et qui sont déployées dans une seule zone de disponibilité.

Les zones Z étant conçues pour être des limites d'isolation des pannes, l'unité de défaillance n'est pas une seule instance EC2, car une instance EC2 complète peut tomber en panne en même temps. Dans ce cas, vous souhaiterez ajouter de la redondance avec un autre AZ, en déployant 10 instances EC2 supplémentaires pour gérer la charge en cas de défaillance d'AZ, pour un total de 20 instances EC2 (suivant le schéma de stabilité statique).

Bien qu'il s'agisse apparemment de 10 instances EC2 de rechange, il ne s'agit en réalité que d'une seule AZ de rechange. Nous n'avons donc pas dépassé le point de baisse des rendements. Cependant, vous pouvez être encore plus rentable tout en augmentant votre disponibilité en utilisant trois AZ et en déployant cinq instances EC2 par AZ.

Cela fournit à une zone de réserve un total de 15 instances EC2 (contre deux zones de 20 instances), tout en fournissant les 10 instances nécessaires pour répondre aux pics de capacité lors d'un événement ayant un impact sur une seule zone de disponibilité. Vous devez donc intégrer l'épargne pour être tolérant aux pannes au-delà de toutes les limites d'isolation des pannes utilisées par la charge de travail (instance, cellule, AZ et région).

Théorème CAP

Une autre façon de penser à la disponibilité est en relation avec le théorème CAP. Le théorème indique qu'un système distribué, composé de plusieurs nœuds stockant des données, ne peut fournir simultanément plus de deux des trois garanties suivantes :

- Cohérence : chaque demande de lecture reçoit l'écriture la plus récente ou une erreur lorsque la cohérence ne peut être garantie.
- Une disponibilité : chaque demande reçoit une réponse sans erreur, même lorsque les nœuds sont en panne ou indisponibles.
- Tolérance de partition : le système continue de fonctionner malgré la perte d'un nombre arbitraire de messages entre les nœuds.

(Pour plus de détails, voir Seth Gilbert et Nancy Lynch, La conjecture de Brewer et la faisabilité de services Web cohérents, disponibles et tolérants aux partitions, ACM SIGACT News, volume 33, numéro 2 (2002), p. 51—59.)

La plupart des systèmes distribués doivent tolérer les défaillances du réseau et, par conséquent, le partitionnement du réseau doit être autorisé. Cela signifie que ces charges de travail doivent choisir entre cohérence et disponibilité lorsqu'une partition réseau se produit. Si la charge de travail choisit la disponibilité, elle renvoie toujours une réponse, mais avec des données potentiellement incohérentes.

S'il choisit la cohérence, il renverra une erreur lors d'une partition réseau, car la charge de travail ne peut pas être sûre de la cohérence des données.

Pour les charges de travail dont l'objectif est de fournir des niveaux de disponibilité plus élevés, elles peuvent choisir Availability and Partition Tolerance (AP) pour éviter de renvoyer des erreurs (indisponibilité) lors d'une partition réseau. Cela entraîne la nécessité d'un modèle de cohérence plus souple, comme une cohérence éventuelle ou une cohérence monotone.

Tolérance aux pannes et isolation des pannes

Ce sont deux concepts importants lorsque l'on pense à la disponibilité. La tolérance aux pannes est la capacité à résister aux défaillances des sous-systèmes et à maintenir la disponibilité (en faisant le bon choix dans le cadre d'un SLA établi). Pour implémenter la tolérance aux pannes, les charges de travail utilisent des sous-systèmes de rechange (ou redondants). Lorsque l'un des sous-systèmes d'un ensemble redondant tombe en panne, un autre reprend le travail, généralement de manière presque fluide. Dans ce cas, les pièces de rechange constituent de véritables capacités inutilisées ; elles sont disponibles pour assumer 100 % du travail lié au sous-système défaillant. Avec de véritables pièces de rechange, plusieurs défaillances de sous-systèmes sont nécessaires pour avoir un impact négatif sur la charge de travail.

L'isolation des défauts minimise l'ampleur de l'impact en cas de panne. Ceci est généralement mis en œuvre avec la modularisation. Les charges de travail sont réparties en petits sous-systèmes qui tombent en panne indépendamment et peuvent être réparés isolément. La défaillance d'un module ne se propage pas au-delà du module. Cette idée s'étend à la fois verticalement, à travers les différentes fonctionnalités d'une charge de travail, et horizontalement, à travers plusieurs sous-systèmes qui fournissent les mêmes fonctionnalités. Ces modules agissent comme des conteneurs de défaillances qui limitent l'étendue de l'impact lors d'un événement.

Les modèles architecturaux des plans de contrôle, des plans de données et de la stabilité statique soutiennent directement la mise en œuvre de la tolérance aux pannes et de l'isolation des pannes. L'article Static stability using Availability Zones de la bibliothèque Amazon Builders fournit de bonnes définitions de ces termes et explique comment ils s'appliquent à la création de charges de travail résilientes et hautement disponibles. Ce livre blanc utilise ces modèles dans la section Conception de systèmes distribués hautement disponibles surAWS, et nous résumons également leurs définitions ici.

- Plan de contrôle : partie de la charge de travail impliquée dans les modifications : ajout de ressources, suppression de ressources, modification des ressources et propagation de ces

modifications là où elles sont nécessaires. Les plans de contrôle sont généralement plus complexes et comportent plus de pièces mobiles que les plans de données. Ils sont donc statistiquement plus susceptibles de tomber en panne et ont une disponibilité moindre.

- Plan de données : partie de la charge de travail qui fournit les fonctionnalités day-to-day métier. Les plans de données ont tendance à être plus simples et à fonctionner à des volumes plus élevés que les plans de contrôle, ce qui se traduit par une plus grande disponibilité.
- Stabilité statique : capacité d'une charge de travail à continuer à fonctionner correctement malgré des troubles de dépendance. L'une des méthodes de mise en œuvre consiste à supprimer les dépendances des plans de contrôle des plans de données. Une autre méthode consiste à coupler de manière souple les dépendances de charge de travail. Peut-être que la charge de travail ne voit aucune information mise à jour (telle que de nouveaux éléments, des éléments supprimés ou des éléments modifiés) que sa dépendance était censée fournir. Cependant, tout ce qu'il faisait avant que la dépendance ne devienne inéluctable continue de fonctionner.

Lorsque nous pensons à la réduction d'une charge de travail, il existe deux approches de haut niveau que nous pouvons envisager pour le rétablissement. La première méthode consiste à réagir à cette déficience une fois qu'elle se produit, par exemple AWS Auto Scaling en ajoutant de nouvelles capacités. La deuxième méthode consiste à se préparer à ces défaillances avant qu'elles ne surviennent, par exemple en surprovisionnant l'infrastructure d'une charge de travail afin qu'elle puisse continuer à fonctionner correctement sans avoir besoin de ressources supplémentaires.

Un système statiquement stable utilise cette dernière approche. Il préapprovisionne la capacité de réserve pour qu'elle soit disponible en cas de panne. Cette méthode évite de créer une dépendance à l'égard d'un plan de contrôle dans le chemin de restauration de la charge de travail afin de fournir une nouvelle capacité de restauration après une panne. En outre, le provisionnement de nouvelles capacités pour diverses ressources prend du temps. En attendant de nouvelles capacités, votre charge de travail peut être surchargée par la demande existante et se dégrader davantage, entraînant une « panne » ou une perte totale de disponibilité. Toutefois, vous devez également tenir compte des implications financières liées à l'utilisation de capacités préprovisionnées par rapport à vos objectifs de disponibilité.

La stabilité statique fournit les deux règles suivantes pour les charges de travail à haute disponibilité.

Règle 7

Ne prenez pas en compte les dépendances des plans de contrôle de votre plan de données, en particulier lors de la restauration.

Règle 8

Couplez les dépendances de manière souple afin que votre charge de travail puisse fonctionner correctement malgré la diminution des dépendances, dans la mesure du possible.

Mesurer la disponibilité

Comme nous l'avons vu précédemment, la création d'un modèle de disponibilité prospectif pour un système distribué est difficile à réaliser et risque de ne pas fournir les informations souhaitées. Ce qui peut s'avérer plus utile, c'est de développer des méthodes cohérentes pour mesurer la disponibilité de votre charge de travail.

La définition de la disponibilité en termes de disponibilité et d'indisponibilité représente l'échec en tant qu'option binaire, que la charge de travail soit élevée ou non.

Toutefois, c'est rarement le cas. Les défaillances ont un certain degré d'impact et se produisent souvent dans certains sous-ensembles de la charge de travail, affectant un pourcentage d'utilisateurs ou de demandes, un pourcentage d'emplacements ou un centile de latence. Ce sont tous des modes de défaillance partielle.

Bien que le MTTR et le MTBF soient utiles pour comprendre ce qui détermine la disponibilité d'un système et, par conséquent, comment l'améliorer, leur utilité n'est pas une mesure empirique de la disponibilité. En outre, les charges de travail sont composées de nombreux composants. Par exemple, une charge de travail telle qu'un système de traitement des paiements est composée de nombreuses interfaces de programmation d'applications (API) et de sous-systèmes. Ainsi, lorsque nous voulons poser une question telle que « quelle est la disponibilité de l'ensemble de la charge de travail ? » , il s'agit en fait d'une question complexe et nuancée.

Dans cette section, nous allons examiner trois manières de mesurer la disponibilité de manière empirique : le taux de réussite des demandes côté serveur, le taux de réussite des demandes côté client et les interruptions annuelles.

Taux de réussite des demandes côté serveur et côté client

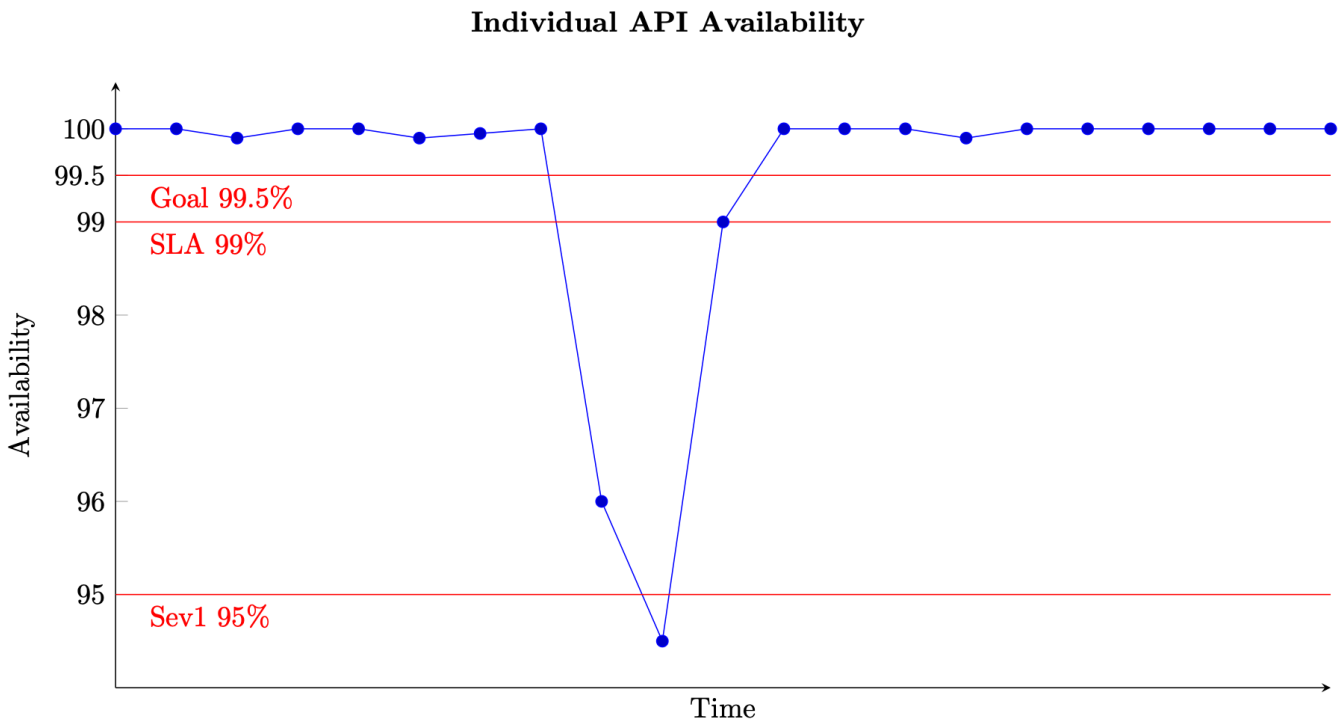
Les deux premières méthodes sont très similaires et ne diffèrent que du point de vue de la mesure. Les métriques côté serveur peuvent être collectées à partir de l'instrumentation du service. Cependant, ils ne sont pas complets. Si les clients ne sont pas en mesure d'accéder au service, vous ne pouvez pas collecter ces statistiques. Pour comprendre l'expérience client, au lieu de se fier aux données télémétriques des clients concernant les demandes ayant échoué, il est plus facile de collecter des statistiques côté client en simulant le trafic client à l'aide de canaries, un logiciel qui analyse régulièrement vos services et enregistre des indicateurs.

Ces deux méthodes calculent la disponibilité comme la fraction du total des unités de travail valides que le service reçoit et de celles qu'il traite avec succès (cela ignore les unités de travail non valides, comme une requête HTTP qui entraîne une erreur 404).

$$A = \frac{Successfully\ Processed\ Units\ of\ Work}{Total\ Valid\ Units\ of\ Work\ Received}$$

Équation 8

Pour un service basé sur des requêtes, l'unité de travail est la requête, comme une requête HTTP. Pour les services basés sur des événements ou des tâches, les unités de travail sont des événements ou des tâches, comme le traitement d'un message à partir d'une file d'attente. Cette mesure de la disponibilité est significative dans des intervalles de temps courts, tels que des fenêtres d'une minute ou de cinq minutes. Il convient également mieux d'un point de vue granulaire, par exemple au niveau de l'API pour un service basé sur des demandes. La figure suivante donne un aperçu de ce à quoi pourrait ressembler la disponibilité au fil du temps lorsqu'elle est calculée de cette manière. Chaque point de données du graphique est calculé à l'aide de l'équation (8) sur une fenêtre de cinq minutes (vous pouvez choisir d'autres dimensions temporelles, comme des intervalles d'une minute ou de dix minutes). Par exemple, le point de données 10 indique une disponibilité de 94,5 %. Cela signifie que pendant les minutes t+45 à t+50, si le service a reçu 1 000 demandes, seules 945 d'entre elles ont été traitées avec succès.



Exemple de mesure de la disponibilité au fil du temps pour une seule API

Le graphique montre également l'objectif de disponibilité de l'API, 99,5 % de disponibilité, le contrat de niveau de service (SLA) qu'elle propose aux clients, 99 % de disponibilité et le seuil d'alerte de gravité élevée, 95 %. Sans le contexte de ces différents seuils, un graphique de disponibilité risque de ne pas fournir d'informations significatives sur le fonctionnement de votre service.

Nous voulons également être en mesure de suivre et de décrire la disponibilité d'un sous-système plus important, tel qu'un plan de contrôle ou un service complet. Pour ce faire, vous pouvez notamment prendre la moyenne de chaque point de données de cinq minutes pour chaque sous-système. Le graphique ressemblera au précédent, mais sera représentatif d'un plus grand ensemble d'entrées. Il accorde également le même poids à tous les sous-systèmes qui constituent votre service. Une autre approche pourrait consister à additionner toutes les demandes reçues et traitées avec succès par toutes les API du service afin de calculer la disponibilité par intervalles de cinq minutes.

Toutefois, cette dernière méthode peut masquer une API individuelle présentant un faible débit et une mauvaise disponibilité. À titre d'exemple simple, considérons un service doté de deux API.

La première API reçoit 1 000 000 de requêtes dans une fenêtre de cinq minutes et traite avec succès 999 000 d'entre elles, soit une disponibilité de 99,9 %. La seconde API reçoit 100 demandes dans cette même fenêtre de cinq minutes et n'en traite que 50 avec succès, soit une disponibilité de 50 %.

Si nous additionnons les requêtes de chaque API, il y a 1 000 100 demandes valides au total et 999 050 d'entre elles sont traitées avec succès, soit une disponibilité de 99,895 % pour l'ensemble du service. Mais si nous faisons la moyenne des disponibilités des deux API, selon la première méthode, nous obtenons une disponibilité de 74,95 %, ce qui est peut-être plus révélateur de l'expérience réelle.

Aucune de ces approches n'est erronée, mais elle montre à quel point il est important de comprendre ce que les indicateurs de disponibilité vous indiquent. Vous pouvez choisir de privilégier la somme des demandes pour tous les sous-systèmes si votre charge de travail reçoit un volume de demandes similaire pour chacun d'entre eux. Cette approche met l'accent sur la « demande » et son succès en tant que mesure de la disponibilité et de l'expérience client. Vous pouvez également choisir de faire la moyenne des disponibilités des sous-systèmes afin de représenter de manière égale leur criticité malgré les différences de volume de demandes. Cette approche met l'accent sur le sous-système et sur la capacité de chacun d'entre eux en tant que proxy de l'expérience client.

Indisponibilité annuelle

La troisième approche consiste à calculer les temps d'arrêt annuels. Cette forme de mesure de disponibilité est plus appropriée à l'établissement et à l'examen d'objectifs à long terme. Cela nécessite de définir ce que les temps d'arrêt signifient pour votre charge de travail. Vous pouvez ensuite mesurer la disponibilité en fonction du nombre de minutes pendant lesquelles la charge de travail n'était pas en « panne » par rapport au nombre total de minutes au cours de la période donnée.

Certaines charges de travail peuvent définir l'indisponibilité comme une baisse en dessous de 95 % de la disponibilité d'une seule API ou d'une fonction de charge de travail pendant un intervalle d'une minute ou cinq minutes (ce qui s'est produit dans le graphique de disponibilité précédent). Vous pouvez également ne prendre en compte les temps d'arrêt que dans la mesure où ils s'appliquent à un sous-ensemble d'opérations critiques du plan de données. Par exemple, le contrat de niveau de service Amazon Messaging (SQS, SNS) relatif à la disponibilité de SQS s'applique à l'API d'envoi, de réception et de suppression de SQS.

Des charges de travail plus importantes et plus complexes peuvent nécessiter la définition de mesures de disponibilité à l'échelle du système. Pour un site de commerce électronique de grande envergure, un indicateur à l'échelle du système peut être quelque chose comme le taux de commandes des clients. Dans ce cas, une baisse de 10 % ou plus des commandes par rapport à la quantité prévue au cours d'une fenêtre de cinq minutes peut définir un temps d'arrêt.

Quelle que soit l'approche choisie, vous pouvez ensuite additionner toutes les périodes de panne pour calculer une disponibilité annuelle. Par exemple, si au cours d'une année civile, il y a eu 27 périodes d'indisponibilité de cinq minutes, définies comme la disponibilité d'une API de plan de données inférieure à 95 %, le temps d'arrêt total était de 135 minutes (certaines périodes de cinq minutes pouvaient être consécutives, d'autres isolées), soit une disponibilité annuelle de 99,97 %.

Cette méthode supplémentaire de mesure de la disponibilité peut fournir des données et des informations absentes des indicateurs côté client et côté serveur. Prenons l'exemple d'une charge de travail altérée et présentant des taux d'erreur très élevés. Les clients soumis à cette charge de travail peuvent cesser complètement d'appeler ses services. Peut-être ont-ils activé un disjoncteur ou suivi leur plan de reprise après sinistre pour utiliser le service dans une autre région. Si nous ne mesurions que les réponses infructueuses, la disponibilité de la charge de travail peut en fait augmenter pendant la panne, non pas parce que la déficience s'améliore ou disparaît, mais parce que les clients cessent tout simplement de l'utiliser.

Latence

Enfin, il est également important de mesurer la latence des unités de traitement du travail au sein de votre charge de travail. La définition de la disponibilité consiste en partie à effectuer le travail dans le cadre d'un SLA établi. Si le retour d'une réponse prend plus de temps que le délai imparti au client, le client a l'impression que la demande a échoué et que la charge de travail n'est pas disponible. Toutefois, côté serveur, la demande peut sembler avoir été traitée avec succès.

La mesure de la latence fournit un autre angle d'évaluation de la disponibilité. L'utilisation de percentiles et d'une moyenne tronquée est une bonne statistique pour cette mesure. Ils sont généralement mesurés au 50e percentile (P50 et TM50) et au 99e percentile (P99 et TM99). La latence doit être mesurée à l'aide de canaries pour représenter l'expérience client ainsi qu'à l'aide de métriques côté serveur. Chaque fois que la moyenne d'un certain centile de latence, comme P99 ou TM99.9, dépasse un SLA cible, vous pouvez prendre en compte ce temps d'arrêt, qui contribue à votre calcul annuel des temps d'arrêt.

Conception de systèmes distribués à haute disponibilité sur AWS

Les sections précédentes ont principalement porté sur la disponibilité théorique des charges de travail et sur ce qu'elles peuvent accomplir. Il s'agit d'un ensemble important de concepts à garder à l'esprit lorsque vous créez des systèmes distribués. Ils vous aideront à définir votre processus de sélection des dépendances et à mettre en œuvre la redondance.

Nous avons également examiné la relation entre MTTDMTTR, et avec MTBF la disponibilité. Cette section présentera des conseils pratiques basés sur la théorie précédente. En bref, les charges de travail d'ingénierie pour la haute disponibilité visent à augmenter MTBF et MTTR à réduire leMTTD.

L'idéal serait d'éliminer toutes les défaillances, mais ce n'est pas réaliste. Dans les grands systèmes distribués où les dépendances sont profondément empilées, des défaillances sont susceptibles de se produire. « Tout échoue tout le temps » (voir Werner Vogels, Amazon.comCTO, 10 leçons tirées de 10 ans d'Amazon Web Services.) et « Vous ne pouvez pas légiférer contre l'échec [alors] concentrez-vous sur une détection et une réponse rapides ». (voir Chris Pinkham, membre fondateur de l'EC2équipe Amazon, ARC335Designing for failure : Architecting Resilient Systems on) AWS

Cela signifie que vous n'avez souvent aucun contrôle sur l'éventualité d'une défaillance. Ce que vous pouvez contrôler, c'est la rapidité avec laquelle vous détectez la panne et prenez des mesures pour y remédier. Ainsi, bien que l'augmentation MTBF reste un élément important de la haute disponibilité, les changements les plus importants que les clients peuvent contrôler sont la réduction MTTD etMTTR.

Rubriques

- Réduire MTTD
- Réduire MTTR
- En augmentation MTBF

Réduire MTTD

Pour réduire le nombre MTTD de défaillances, il faut les découvrir le plus rapidement possible. Le raccourcissement MTTD est basé sur l'observabilité, c'est-à-dire sur la manière dont vous avez instrumenté votre charge de travail pour comprendre son état. Les clients doivent surveiller leurs

indicateurs d'expérience client dans les sous-systèmes critiques de leur charge de travail afin d'identifier de manière proactive le moment où un problème survient (voir l'annexe 1) MTTD et les indicateurs MTTR critiques pour plus d'informations sur ces indicateurs.). Les clients peuvent utiliser Amazon CloudWatch Synthetics pour créer des canaris qui surveillent APIs votre expérience utilisateur et celle de vos consoles afin de mesurer de manière proactive l'expérience utilisateur. Il existe un certain nombre d'autres mécanismes de vérification de l'état qui peuvent être utilisés pour les minimiserMTTD, tels que les contrôles de santé d'Elastic Load Balancing (ELB), les contrôles de santé d'Amazon Route 53, etc. (Voir Amazon Builders' Library — Implementation des bilans de santé.)

Votre surveillance doit également être capable de détecter les défaillances partielles du système dans son ensemble et de vos sous-systèmes individuels. Vos indicateurs de disponibilité, de défaillance et de latence doivent utiliser la dimensionnalité de vos limites d'isolation des pannes comme dimensions CloudWatch métriques. Par exemple, considérez une EC2 instance unique faisant partie d'une architecture basée sur des cellules, dans l'AZ use1-az1, dans la région us-east-1, qui fait partie de la mise à jour de la charge de travail qui fait partie de son sous-système de plan de contrôle. API Lorsque le serveur envoie ses métriques, il peut utiliser son identifiant d'instance, son AZ, sa région, son nom et API le nom du sous-système comme dimensions. Cela vous permet d'avoir de l'observabilité et de définir des alarmes pour chacune de ces dimensions afin de détecter les défaillances.

Réduire MTTR

Après la découverte d'une défaillance, le reste du MTTR temps est consacré à la réparation effective ou à l'atténuation de l'impact. Pour réparer ou atténuer une panne, vous devez savoir ce qui ne va pas. Deux groupes clés de mesures fournissent des informations au cours de cette phase : 1/ les métriques d'évaluation d'impact et 2/ les métriques de santé opérationnelle. Le premier groupe indique l'ampleur de l'impact en cas de panne, en mesurant le nombre ou le pourcentage de clients, de ressources ou de charges de travail affectés. Le deuxième groupe aide à déterminer pourquoi il y a un impact. Une fois le pourquoi découvert, les opérateurs et l'automatisation peuvent réagir et résoudre la panne. Reportez-vous à l'annexe 1 MTTD et MTTR aux mesures critiques pour plus d'informations sur ces mesures.

> Règle 9
>
> L'observabilité et l'instrumentation sont essentielles pour réduire MTTD etMTTR.

Parcours pour contourner l'échec

L'approche la plus rapide pour atténuer l'impact consiste à utiliser des sous-systèmes rapides qui contournent les défaillances. Cette approche utilise la redondance pour réduire la charge de travail en MTTR transférant rapidement le travail d'un sous-système défaillant vers un sous-système de rechange. La redondance peut aller des processus logiciels aux EC2 instances, en passant par plusieurs ou plusieurs AZs régions.

Les sous-systèmes de rechange peuvent les MTTR réduire à presque zéro. Le temps de reprise correspond uniquement à ce qu'il faut pour réacheminer le travail vers le serveur de réserve. Cela se produit souvent avec une latence minimale et permet au travail de se terminer dans les délais définisSLA, tout en maintenant la disponibilité du système. MTTRsCela se traduit par des retards légers, voire imperceptibles, plutôt que par des périodes d'indisponibilité prolongées.

Par exemple, si votre service utilise des EC2 instances situées derrière un Application Load Balancer ALB (), vous pouvez configurer des contrôles de santé à un intervalle de cinq secondes seulement et n'exiger que deux tests de santé ayant échoué avant qu'une cible ne soit marquée comme non saine. Cela signifie qu'en 10 secondes, vous pouvez détecter une panne et arrêter d'envoyer du trafic vers l'hôte défaillant. Dans ce cas, MTTR c'est effectivement le même que le cas MTTD puisque dès que la panne est détectée, elle est également atténuée.

C'est ce que cherchent à atteindre les charges de travail à haute disponibilité ou à disponibilité continue. Nous voulons contourner rapidement les défaillances de la charge de travail en détectant rapidement les sous-systèmes défaillants, en les marquant comme défaillants, en cessant de leur envoyer du trafic et en envoyant le trafic vers un sous-système redondant.

Notez que l'utilisation de ce type de mécanisme de rapidité rend votre charge de travail très sensible aux erreurs transitoires. Dans l'exemple fourni, assurez-vous que les vérifications de l'état de votre équilibreur de charge ne portent que sur des vérifications superficielles ou locales de l'instance, et non sur les dépendances ou les flux de travail (souvent appelées vérifications de santé approfondies). Cela permettra d'éviter le remplacement inutile d'instances lors d'erreurs transitoires affectant la charge de travail.

L'observabilité et la capacité à détecter les défaillances dans les sous-systèmes sont essentielles à la réussite du contournement des défaillances. Vous devez connaître l'ampleur de l'impact afin que les ressources affectées puissent être signalées comme étant défectueuses ou défaillantes et mises hors service afin de pouvoir être réacheminées. Par exemple, si le service d'une seule zone de disponibilité est partiellement perturbé, votre instrumentation devra être en mesure d'identifier

l'existence d'un problème localisé dans la zone AZ pour acheminer toutes les ressources de cette zone jusqu'à ce qu'elle soit rétablie.

La capacité à contourner les défaillances peut également nécessiter un outillage supplémentaire en fonction de l'environnement. En utilisant l'exemple précédent avec des EC2 instances situées derrière unALB, imaginez que les instances d'une zone géographique passent avec succès les tests de santé locaux, mais qu'une déficience isolée de la zone Z les empêche de se connecter à leur base de données dans une autre zone. Dans ce cas, les contrôles de santé de l'équilibrage de charge ne mettront pas ces instances hors service. Un mécanisme automatisé différent serait nécessaire pour supprimer l'AZ de l'équilibreur de charge ou forcer les instances à échouer à leurs tests de santé, ce qui dépend de l'identification du fait que l'étendue de l'impact est une AZ. Pour les charges de travail qui n'utilisent pas d'équilibreur de charge, une méthode similaire serait nécessaire pour empêcher les ressources d'une zone de travail spécifique d'accepter des unités de travail ou de supprimer complètement de la capacité de la zone de gestion.

Dans certains cas, le transfert du travail vers un sous-système redondant ne peut pas être automatisé, comme le basculement d'une base de données principale vers une base de données secondaire lorsque la technologie ne fournit pas sa propre élection de leader. Il s'agit d'un scénario courant pour les architectures AWS multirégionales. Étant donné que ces types de basculement nécessitent un certain temps d'arrêt, ne peuvent pas être annulés immédiatement et laissent la charge de travail sans redondance pendant un certain temps, il est important de faire participer un humain au processus décisionnel.

Les charges de travail qui peuvent adopter un modèle de cohérence moins strict peuvent être réduites MTTRs en utilisant l'automatisation du basculement multirégional pour contourner les défaillances. Des fonctionnalités telles que la réplication entre régions Amazon S3 ou les tables globales Amazon DynamoDB fournissent des fonctionnalités multirégionales grâce à une réplication finalement cohérente. De plus, l'utilisation d'un modèle de cohérence assoupli est bénéfique lorsque l'on considère le CAP théorème. Lors de pannes réseau qui ont un impact sur la connectivité aux sous-systèmes dynamiques, si la charge de travail choisit la disponibilité plutôt que la cohérence, elle peut toujours fournir des réponses sans erreur, ce qui constitue un autre moyen de contourner les défaillances.

Le routage en cas de défaillance peut être mis en œuvre à l'aide de deux stratégies différentes. La première stratégie consiste à implémenter la stabilité statique en préprovisionnant suffisamment de ressources pour gérer la charge complète du sous-système défaillant. Il peut s'agir d'une EC2 instance unique ou d'une capacité totale d'AZ. Tenter de provisionner de nouvelles ressources en cas

de panne augmente le plan de contrôle MTTR et ajoute une dépendance à celui-ci dans votre chemin de restauration. Cependant, cela entraîne des frais supplémentaires.

La deuxième stratégie consiste à acheminer une partie du trafic du sous-système défaillant vers d'autres sous-systèmes et à éliminer le trafic excédentaire qui ne peut pas être traité par la capacité restante. Au cours de cette période de dégradation, vous pouvez augmenter le volume de nouvelles ressources pour remplacer la capacité défaillante. Cette approche est plus longue MTTR et crée une dépendance à l'égard d'un plan de contrôle, mais elle coûte moins cher en réserve et en capacité de réserve.

Retour à un état connu en bon état

Une autre approche courante pour atténuer les risques pendant la réparation consiste à remettre la charge de travail dans un état normal connu auparavant. Si une modification récente est susceptible d'être à l'origine de l'échec, l'annulation de cette modification est un moyen de revenir à l'état précédent.

Dans d'autres cas, des conditions transitoires peuvent être à l'origine de l'échec, auquel cas le redémarrage de la charge de travail peut atténuer l'impact. Examinons ces deux scénarios.

Lors d'un déploiement, la minimisation de la MTTD terre MTTR repose sur l'observabilité et l'automatisation. Votre processus de déploiement doit surveiller en permanence la charge de travail pour détecter toute augmentation des taux d'erreur, une latence accrue ou des anomalies. Une fois ceux-ci reconnus, le processus de déploiement doit être interrompu.

Il existe différentes stratégies de déploiement, telles que les déploiements sur place, les déploiements bleu/vert et les déploiements progressifs. Chacun d'entre eux peut utiliser un mécanisme différent pour revenir à un état dont le fonctionnement a été vérifié. Il peut revenir automatiquement à l'état précédent, ramener le trafic vers l'environnement bleu ou nécessiter une intervention manuelle.

CloudFormation offre la possibilité de revenir en arrière automatiquement dans le cadre de ses opérations de création et de mise à jour de la pile, comme le fait AWS CodeDeploy. CodeDeploy prend également en charge les déploiements bleu/vert et les déploiements progressifs.

Pour tirer parti de ces fonctionnalités et minimiser les vôtresMTTR, envisagez d'automatiser l'ensemble de votre infrastructure et de vos déploiements de code par le biais de ces services. Dans les scénarios où vous ne pouvez pas utiliser ces services, envisagez d'implémenter le modèle Saga AWS Step Functions pour annuler les déploiements ayant échoué.

Lorsque l'on envisage le redémarrage, il existe plusieurs approches différentes. Cela va du redémarrage d'un serveur, la tâche la plus longue, au redémarrage d'un thread, la tâche la plus courte. Voici un tableau qui décrit certaines des approches de redémarrage et les durées approximatives d'exécution (représentatives de la différence d'ordres de grandeur, elles ne sont pas exactes).

Mécanisme de restauration en cas de panne	Estimé MTTR
Lancer et configurer un nouveau serveur virtuel	15 minutes
Redéployez le logiciel	10 minutes
Redémarrer le serveur	5 minutes
Redémarrer ou lancer le conteneur	2 secondes
Invoquer une nouvelle fonction sans serveur	100 millisecondes
Processus de redémarrage	10 millisecondes
Redémarrer le fil	10 µs

MTTREn examinant le tableau, l'utilisation de conteneurs et de fonctions sans serveur (comme AWS Lambda) présente des avantages évidents. Ils MTTR sont bien plus rapides que le redémarrage d'une machine virtuelle ou le lancement d'une nouvelle machine. Cependant, l'utilisation de l'isolation des défauts par le biais de la modularité logicielle est également avantageuse. Si vous pouvez contenir l'échec d'un seul processus ou thread, la reprise après cette défaillance est beaucoup plus rapide que le redémarrage d'un conteneur ou d'un serveur.

Comme approche générale de la restauration, vous pouvez passer de bas en haut : 1/Restart, 2/Reboot, 3/RE-image/Redeploy, 4/Replace. Cependant, une fois que vous avez atteint l'étape de redémarrage, le routage en cas d'échec est généralement une approche plus rapide (cela prend généralement 3 à 4 minutes au maximum). Ainsi, pour atténuer le plus rapidement possible l'impact d'une tentative de redémarrage, contournez la panne puis, en arrière-plan, poursuivez le processus de restauration pour rétablir la capacité de votre charge de travail.

Règle 10

Concentrez-vous sur l'atténuation des impacts et non sur la résolution des problèmes. Prenez le chemin le plus rapide pour revenir à un fonctionnement normal.

Diagnostic des défaillances

Une partie du processus de réparation après la détection est la période de diagnostic. Il s'agit de la période pendant laquelle les opérateurs tentent de déterminer ce qui ne va pas. Ce processus peut impliquer d'interroger les journaux, de passer en revue les indicateurs de santé opérationnelle ou de se connecter aux hôtes pour résoudre les problèmes. Toutes ces actions prennent du temps. La création d'outils et de runbooks pour accélérer ces actions peut également contribuer à MTTR les réduire.

Runbooks et automatisation

De même, une fois que vous avez déterminé ce qui ne va pas et quel plan d'action permettra de réparer la charge de travail, les opérateurs doivent généralement suivre un ensemble d'étapes pour y parvenir. Par exemple, après une panne, le moyen le plus rapide de réparer la charge de travail peut être de la redémarrer, ce qui peut impliquer plusieurs étapes ordonnées. L'utilisation d'un manuel qui automatise ces étapes ou fournit des instructions spécifiques à un opérateur accélérera le processus et contribuera à réduire le risque d'action involontaire.

En augmentation MTBF

Le dernier élément pour améliorer la disponibilité consiste à augmenter leMTBF. Cela peut s'appliquer à la fois au logiciel et aux AWS services utilisés pour l'exécuter.

Élargir le système distribué MTBF

L'un des moyens d'augmenter MTBF est de réduire les défauts du logiciel. Il existe plusieurs méthodes pour le faire. Les clients peuvent utiliser des outils tels qu'Amazon CodeGuru Reviewer pour détecter et corriger les erreurs courantes. Vous devez également effectuer des révisions complètes du code par les pairs, des tests unitaires, des tests d'intégration, des tests de régression et des tests de charge sur le logiciel avant son déploiement en production. L'augmentation de la couverture du code dans les tests permettra de garantir que même les chemins d'exécution de code peu courants sont testés.

Le déploiement de modifications mineures peut également aider à éviter des résultats inattendus en réduisant la complexité des modifications. Chaque activité permet d'identifier et de corriger les défauts avant qu'ils ne puissent être invoqués.

Une autre approche pour prévenir les défaillances consiste à effectuer des tests réguliers. La mise en œuvre d'un programme d'ingénierie du chaos peut vous aider à tester les défaillances de votre charge de travail, à valider les procédures de reprise et à identifier et corriger les modes de défaillance avant qu'ils ne surviennent en production. Les clients peuvent utiliser le simulateur d'injection de AWS défauts dans le cadre de leur ensemble d'outils d'expérimentation d'ingénierie du chaos.

La tolérance aux pannes est un autre moyen de prévenir les défaillances dans un système distribué. Les modules rapides, les nouvelles tentatives avec retard et instabilité exponentiels, les transactions et l'idempotencie sont autant de techniques qui contribuent à rendre les charges de travail tolérantes aux pannes.

Les transactions sont un groupe d'opérations qui respectent les ACID propriétés. Ce sont les suivants :

- Atomicité — Soit toutes les actions se produisent, soit aucune d'entre elles ne se produira.
- Cohérence — Chaque transaction laisse la charge de travail dans un état valide.
- Isolation — Les transactions effectuées simultanément laissent la charge de travail dans le même état que si elles avaient été effectuées de manière séquentielle.
- Durabilité — Une fois qu'une transaction est validée, tous ses effets sont préservés, même en cas de défaillance de la charge de travail.

Les nouvelles tentatives avec retard et instabilité exponentiels vous permettent de surmonter les défaillances transitoires causées par des Heisenbugs, une surcharge ou d'autres conditions. Lorsque les transactions sont idempotentes, elles peuvent être réessayées plusieurs fois sans effets secondaires.

Si nous prenons en compte l'effet d'un Heisenbug sur une configuration matérielle tolérante aux pannes, nous ne nous inquiétons guère, car la probabilité que le Heisenbug apparaisse à la fois sur le sous-système principal et sur le sous-système redondant est infiniment faible. (Voir Jim Gray, « Pourquoi les ordinateurs s'arrêtent-ils et que peut-on faire pour y remédier ? », juin 1985, Rapport technique de Tandem 85.7.) Dans les systèmes distribués, nous voulons obtenir les mêmes résultats avec nos logiciels.

Lorsqu'un Heisenbug est invoqué, il est impératif que le logiciel détecte rapidement l'opération incorrecte et échoue afin de pouvoir réessayer. Ceci est réalisé grâce à une programmation défensive et à la validation des entrées, des résultats intermédiaires et des sorties. De plus, les processus sont isolés et ne partagent aucun état avec les autres processus.

Cette approche modulaire garantit que l'ampleur de l'impact en cas de panne est limitée. Les processus échouent indépendamment. Lorsqu'un processus échoue, le logiciel doit utiliser des « paires de processus » pour recommencer le travail, ce qui signifie qu'un nouveau processus peut assumer le travail d'un processus défaillant. Pour maintenir la fiabilité et l'intégrité de la charge de travail, chaque opération doit être traitée comme une ACID transaction.

Cela permet à un processus d'échouer sans altérer l'état de la charge de travail en annulant la transaction et en annulant les modifications apportées. Cela permet au processus de restauration de réessayer la transaction à partir d'un état dont le fonctionnement a été vérifié et de redémarrer correctement. C'est ainsi que le logiciel peut être tolérant aux pannes face aux Heisenbugs.

Cependant, vous ne devez pas viser à rendre le logiciel tolérant aux bohrbugs. Ces défauts doivent être détectés et corrigés avant que la charge de travail n'entre en production, car aucun niveau de redondance ne permettra d'obtenir un résultat correct. (Voir Jim Gray, « Pourquoi les ordinateurs s'arrêtent-ils et que peut-on faire pour y remédier ? », juin 1985, Rapport technique de Tandem 85.7.)

La dernière méthode d'augmentation MTBF consiste à réduire la portée de l'impact d'une défaillance. L'utilisation de l'isolation des pannes par modularisation pour créer des conteneurs de défaillances est un moyen principal d'y parvenir, comme indiqué précédemment dans Tolérance aux pannes et isolation des pannes. La réduction du taux de défaillance améliore la disponibilité. AWS utilise des techniques telles que la division des services en plans de contrôle et en plans de données, l'indépendance des zones de disponibilité (AZI), l'isolation régionale, les architectures basées sur les cellules et le shuffle-sharding pour isoler les pannes. Ce sont également des modèles qui peuvent également être utilisés par les AWS clients.

Par exemple, examinons un scénario dans lequel une charge de travail plaçait les clients dans différents conteneurs de défaillances de son infrastructure desservant au plus 5 % du total des clients. L'un de ces conteneurs d'erreurs connaît un événement qui augmente la latence au-delà du délai d'expiration du client pour 10 % des demandes. Lors de cet événement, pour 95 % des clients, le service était disponible à 100 %. Pour les 5 % restants, le service semblait être disponible à 90 %. Cela se traduit par une disponibilité de $1 - (5\%\ of\ customers \times 10\%\ of\ their\ requests) = 99{,}5\%$ au lieu de 10 % des demandes échouées pour 100 % des clients (soit une disponibilité de 90 %).

Règle 11

L'isolation des défaillances réduit la portée de l'impact et augmente la charge MTBF de travail en réduisant le taux de défaillance global.

Dépendance croissante MTBF

La première méthode pour augmenter votre AWS dépendance MTBF consiste à utiliser l'isolation des pannes. De nombreux AWS services offrent un certain niveau d'isolation au niveau de l'AZ, ce qui signifie qu'une panne dans un AZ n'affecte pas le service dans un AZ différent.

L'utilisation d'EC2instances redondantes dans plusieurs instances AZs augmente la disponibilité des sous-systèmes. AZIfournit une capacité d'épargne au sein d'une seule région, ce qui vous permet d'augmenter la disponibilité de vos AZI services.

Cependant, tous les AWS services ne fonctionnent pas au niveau AZ. Beaucoup d'autres offrent un isolement régional. Dans ce cas, lorsque la disponibilité prévue du service régional ne prend pas en charge la disponibilité globale requise pour votre charge de travail, vous pouvez envisager une approche multirégionale. Chaque région propose une instanciation isolée du service, ce qui équivaut à du sparing.

Il existe différents services qui facilitent la création d'un service multirégional. Par exemple :

- Base de données mondiale Amazon Aurora
- Tableaux globaux Amazon DynamoDB
- Amazon ElastiCache (RedisOSS) — Banque de données mondiale
- AWS Accélérateur mondial
- Réplication entre régions Amazon S3
- Contrôleur de restauration d'applications Amazon Route 53

Ce document n'aborde pas les stratégies de création de charges de travail multirégionales, mais vous devez évaluer les avantages de disponibilité des architectures multirégionales avec les coûts supplémentaires, la complexité et les pratiques opérationnelles nécessaires pour atteindre les objectifs de disponibilité souhaités.

La méthode suivante pour augmenter la dépendance MTBF consiste à concevoir votre charge de travail de manière à ce qu'elle soit statiquement stable. Par exemple, vous avez une charge de travail

qui fournit des informations sur les produits. Lorsque vos clients font une demande pour un produit, votre service adresse une demande à un service de métadonnées externe pour récupérer les détails du produit. Votre charge de travail renvoie ensuite toutes ces informations à l'utilisateur.

Toutefois, si le service de métadonnées n'est pas disponible, les demandes de vos clients échouent. Au lieu de cela, vous pouvez extraire ou transférer de manière asynchrone les métadonnées localement vers votre service afin de les utiliser pour répondre aux demandes. Cela élimine l'appel synchrone au service de métadonnées depuis votre chemin critique.

En outre, étant donné que votre service est toujours disponible même lorsque le service de métadonnées ne l'est pas, vous pouvez le supprimer en tant que dépendance dans votre calcul de disponibilité. Cet exemple repose sur l'hypothèse selon laquelle les métadonnées ne changent pas fréquemment et qu'il vaut mieux servir des métadonnées périmées que l'échec de la demande. Un autre exemple similaire est celui de Serve-Stale, DNS qui permet de conserver les données dans le cache au-delà de TTL leur expiration et de les utiliser pour les réponses lorsqu'une réponse actualisée n'est pas facilement disponible.

La dernière méthode pour augmenter la dépendance MTBF consiste à réduire la portée de l'impact d'une défaillance. Comme indiqué précédemment, l'échec n'est pas un événement binaire, il existe des degrés de défaillance. C'est l'effet de la modularisation ; l'échec est limité aux demandes ou aux utilisateurs desservis par ce conteneur.

Cela permet de réduire le nombre de défaillances lors d'un événement, ce qui augmente en fin de compte la disponibilité de la charge de travail globale en limitant l'étendue de l'impact.

Réduire les sources d'impact communes

En 1985, Jim Gray a découvert, lors d'une étude menée par Tandem Computers, que les défaillances étaient principalement dues à deux facteurs : le logiciel et les opérations. (Voir Jim Gray, « Pourquoi les ordinateurs s'arrêtent-ils et que peut-on faire pour y remédier ? », juin 1985, Rapport technique de Tandem 85.7.) Même 36 ans plus tard, cela continue d'être vrai. Malgré les avancées technologiques, il n'existe pas de solution facile à ces problèmes, et les principales causes de défaillance n'ont pas changé. La résolution des défaillances logicielles a été abordée au début de cette section. L'accent sera donc mis ici sur les opérations et la réduction de la fréquence des défaillances.

Stabilité par rapport aux fonctionnalités

Si nous nous référons au graphique des taux d'échec du logiciel et du matériel dans la sectionthe section called “Disponibilité du système distribué”, nous pouvons remarquer que des défauts sont

ajoutés dans chaque version logicielle. Cela signifie que toute modification de la charge de travail augmente le risque de défaillance. Ces modifications concernent généralement de nouvelles fonctionnalités, ce qui constitue un corollaire. Des charges de travail à disponibilité plus élevée favoriseront la stabilité par rapport aux nouvelles fonctionnalités. Ainsi, l'un des moyens les plus simples d'améliorer la disponibilité consiste à déployer moins souvent ou à proposer moins de fonctionnalités. Les charges de travail qui sont déployées plus fréquemment seront intrinsèquement moins disponibles que celles qui ne le sont pas. Cependant, les charges de travail qui n'ajoutent pas de fonctionnalités ne répondent pas à la demande des clients et peuvent devenir moins utiles au fil du temps.

Alors, comment continuer à innover et à publier des fonctionnalités en toute sécurité ? La réponse est la standardisation. Quelle est la bonne méthode de déploiement ? Comment commandez-vous des déploiements ? Quelles sont les normes de test ? Combien de temps attendez-vous entre les étapes ? Vos tests unitaires couvrent-ils une partie suffisante du code logiciel ? La normalisation répondra à ces questions et évitera les problèmes causés par des facteurs tels que l'absence de tests de charge, le fait de sauter des étapes de déploiement ou un déploiement trop rapide sur un trop grand nombre d'hôtes.

La façon dont vous implémentez la standardisation passe par l'automatisation. Cela réduit les risques d'erreurs humaines et permet aux ordinateurs de faire ce pour quoi ils sont doués, c'est-à-dire faire toujours la même chose de la même manière. La façon dont vous combinez standardisation et automatisation consiste à définir des objectifs. Des objectifs tels que l'absence de modifications manuelles, l'accès à l'hôte uniquement par le biais de systèmes d'autorisation conditionnelsAPI, l'écriture de tests de charge pour chacun, etc. L'excellence opérationnelle est une norme culturelle qui peut nécessiter des changements substantiels. L'établissement et le suivi des performances par rapport à un objectif contribuent à susciter un changement culturel qui aura un impact important sur la disponibilité de la charge de travail. Le pilier AWS Well-Architected Operational Excellence fournit des meilleures pratiques complètes pour l'excellence opérationnelle.

Sécurité des opérateurs

L'autre facteur majeur qui contribue aux événements opérationnels qui entraînent des défaillances sont les personnes. Les humains commettent des erreurs. Ils peuvent utiliser des informations d'identification incorrectes, saisir la mauvaise commande, appuyer sur Entrée trop tôt ou manquer une étape critique. Toute action manuelle entraîne systématiquement des erreurs, ce qui entraîne un échec.

L'une des principales causes des erreurs des opérateurs réside dans les interfaces utilisateur confuses, peu intuitives ou incohérentes. Jim Gray a également noté dans son étude de 1985 que

« les interfaces qui demandent des informations à l'opérateur ou lui demandent d'exécuter une fonction doivent être simples, cohérentes et tolérantes aux pannes de l'opérateur ». (Voir Jim Gray, « Pourquoi les ordinateurs s'arrêtent-ils et que peut-on faire pour y remédier ? », juin 1985, Rapport technique de Tandem 85.7.) Cette idée est toujours vraie aujourd'hui. Au cours des trente dernières années, il existe de nombreux exemples dans l'industrie où une interface utilisateur confuse ou complexe, l'absence de confirmation ou d'instructions, ou même simplement un langage humain peu convivial ont incité un opérateur à faire le mauvais choix.

> Règle 12
>
> Permettez aux opérateurs de prendre facilement les bonnes décisions.

Prévenir les surcharges

Le dernier contributeur commun à l'impact est constitué par vos clients, les véritables utilisateurs de votre charge de travail. Les charges de travail réussies ont tendance à s'habituer, dans une large mesure, mais cette utilisation dépasse parfois la capacité d'évolution de la charge de travail. De nombreuses choses peuvent se produire : les disques peuvent être saturés, les pools de threads peuvent être épuisés, la bande passante réseau peut être saturée ou les limites de connexion à la base de données peuvent être atteintes.

Il n'existe pas de méthode infaillible pour les éliminer, mais une surveillance proactive de la capacité et de l'utilisation par le biais de métriques Operational Health fournira des alertes précoces lorsque de telles défaillances pourraient survenir. Des techniques telles que le délestage, les disjoncteurs et les nouvelles tentatives avec recul et instabilité exponentiels peuvent aider à minimiser l'impact et à augmenter le taux de réussite, mais ces situations constituent tout de même un échec. La mise à l'échelle automatisée basée sur les indicateurs de santé opérationnelle peut aider à réduire la fréquence des pannes dues à une surcharge, mais risque de ne pas être en mesure de répondre assez rapidement aux changements d'utilisation.

Si vous devez garantir la disponibilité continue de la capacité pour les clients, vous devez faire des compromis en termes de disponibilité et de coût. L'un des moyens de garantir que le manque de capacité n'entraîne pas d'indisponibilité est de fournir un quota à chaque client et de veiller à ce que la capacité de votre charge de travail soit adaptée à 100 % des quotas alloués. Lorsque les clients dépassent leur quota, ils sont limités, ce qui n'est pas un échec et n'est pas pris en compte dans la disponibilité. Vous devrez également suivre de près votre clientèle et prévoir son utilisation future afin

de maintenir une capacité suffisante. Cela garantit que votre charge de travail n'est pas entraînée par des scénarios de défaillance en raison d'une consommation excessive par vos clients.

- Amazon Builders' Library — Utiliser le délestage pour éviter les surcharges
- Amazon Builders' Library — Équité dans les systèmes multi-locataires

Par exemple, examinons une charge de travail qui fournit un service de stockage. Chaque serveur de la charge de travail peut prendre en charge 100 téléchargements par seconde, les clients disposent d'un quota de 200 téléchargements par seconde, et il y a 500 clients. Pour pouvoir prendre en charge ce volume de clients, le service doit fournir une capacité de 100 000 téléchargements par seconde, ce qui nécessite 1 000 serveurs. Si un client dépasse son quota, il est limité, ce qui garantit une capacité suffisante pour tous les autres clients. Il s'agit d'un exemple simple d'une façon d'éviter la surcharge sans rejeter les unités de travail.

Conclusion

Nous avons établi 12 règles de haute disponibilité dans ce document.

- Règle 1 — Des pannes moins fréquentes (MTBF plus long), des temps de détection des pannes plus courts (MTTD plus court) et des temps de réparation plus courts (MTTR plus court) sont les trois facteurs utilisés pour améliorer la disponibilité dans les systèmes distribués.
- Règle 2 — La disponibilité du logiciel dans votre charge de travail est un facteur important de la disponibilité globale de votre charge de travail et doit bénéficier de la même attention que les autres composants.
- Règle 3 — La réduction des dépendances peut avoir un impact positif sur la disponibilité.
- Règle 4 — En général, sélectionnez les dépendances dont les objectifs de disponibilité sont égaux ou supérieurs aux objectifs de votre charge de travail.
- Règle 5 — Utilisez la modération pour augmenter la disponibilité des dépendances dans une charge de travail.
- Règle 6 — Il existe une limite supérieure à la rentabilité de l'épargne. Utilisez le moins de pièces de rechange nécessaires pour atteindre la disponibilité requise.
- Règle 7 — Ne vous fiez pas aux plans de contrôle de votre plan de données, en particulier lors de la restauration.
- Règle 8 — Associez les dépendances de manière souple afin que votre charge de travail puisse fonctionner correctement malgré la déficience de la dépendance, dans la mesure du possible.
- Règle 9 — L'observabilité et l'instrumentation sont essentielles pour réduire le MTTD et le MTTR.
- Règle 10 — Concentrez-vous sur l'atténuation des impacts, pas sur la résolution des problèmes. Prenez le chemin le plus rapide pour revenir à un fonctionnement normal.
- Règle 11 — L'isolation des pannes réduit l'ampleur de l'impact et augmente le MTBF de la charge de travail en réduisant le taux de défaillance global.
- Règle 12 — Permettez aux opérateurs de faire facilement ce qu'il faut.

L'amélioration de la disponibilité de la charge de travail passe par la réduction du MTTD et du MTTR et par l'augmentation du MTBF. En résumé, nous avons discuté des moyens suivants d'améliorer la disponibilité qui couvrent la technologie, les personnes et les processus.

- MTTD
  - Réduisez le MTTD grâce à une surveillance proactive de vos indicateurs d'expérience client.

  - Profitez de contrôles de santé granulaires pour un basculement rapide.
- MTTR
  - Surveillez l'étendue de l'impact et les indicateurs de santé opérationnelle.
  - Réduisez le MTTR en suivant 1/Redémarrer, 2/Redémarrer, 3/Réimage/Redéployer et 4/Remplacer.
  - Contournez la défaillance en comprenant l'ampleur de l'impact.
  - Utilisez des services dont les temps de redémarrage sont plus courts, tels que les conteneurs et les fonctions sans serveur sur des machines virtuelles ou des hôtes physiques.
  - Annulez automatiquement les déploiements ayant échoué lorsque cela est possible.
  - Établissez des runbooks et des outils opérationnels pour les opérations de diagnostic et les procédures de redémarrage.
- MTBF
  - Éliminez les bogues et les défauts des logiciels grâce à des tests rigoureux avant leur mise en production.
  - Mettez en œuvre l'ingénierie du chaos et l'injection de défauts.
  - Utilisez le bon niveau d'économie dans les dépendances pour tolérer les pannes.
  - Minimisez l'ampleur de l'impact en cas de défaillance grâce à des conteneurs de défauts.
  - Mettez en œuvre des normes pour les déploiements et les modifications.
  - Concevez des interfaces opérateurs simples, intuitives, cohérentes et bien documentées.
  - Fixez-vous des objectifs d'excellence opérationnelle.
  - Privilégiez la stabilité par rapport à la publication de nouvelles fonctionnalités lorsque la disponibilité est une dimension critique de votre charge de travail.
  - Implémentez des quotas d'utilisation avec limitation ou délestage, ou les deux, pour éviter les surcharges.

N'oubliez pas que nous ne parviendrons jamais complètement à empêcher l'échec. Concentrez-vous sur des conceptions logicielles offrant la meilleure isolation possible contre les pannes afin de limiter la portée et l'ampleur de l'impact, en maintenant idéalement cet impact en dessous des seuils de « temps d'arrêt » ET investissez dans une détection et une atténuation très rapides et très fiables. Les systèmes distribués modernes doivent encore considérer les défaillances comme une fatalité et être conçus à tous les niveaux pour garantir une haute disponibilité.

Annexe 1 — Métriques critiques du MTTD et du MTTR

Vous trouverez ci-dessous un cadre de standardisation de l'instrumentation et de l'observabilité qui peut aider à réduire le MTTD et le MTTR lors d'un événement.

Métriques de l'expérience client. Ces indicateurs indiquent qu'un service est réactif et disponible pour répondre aux demandes des clients. Par exemple, la latence du plan de contrôle. Ces mesures mesurent le taux d'erreur, la disponibilité, la latence, le volume et le taux d'accélération.

Métriques d'évaluation d'impact. Ces indicateurs fournissent des informations sur l'ampleur de l'impact des événements. Par exemple, le nombre ou le pourcentage de clients concernés par un événement survenu dans un avion de données. Mesure le nombre ou le pourcentage d'éléments concernés.

Métriques de santé opérationnelle. Ces indicateurs indiquent qu'un service est réactif et disponible pour répondre aux demandes des clients, tout en se concentrant sur des sous-systèmes et des ressources d'infrastructure communs. Par exemple, le pourcentage d'utilisation du processeur de votre parc EC2. Ces mesures doivent mesurer l'utilisation, la capacité, le débit, le taux d'erreur, la disponibilité et la latence.

Collaborateurs

Les contributeurs à ce document incluent :

- Michael Haken, architecte de solutions principal, Amazon Web Services

Suggestions de lecture

Pour plus d'informations, consultez :

- Pilier de fiabilité bien conçu
- Pilier d'excellence opérationnelle bien conçu
- Amazon Builders' Library : garantir la sécurité des annulations lors des déploiements
- Amazon Builders' Library — Au-delà des cinq années 9 : leçons tirées de nos plans de données les plus élevés disponibles
- Amazon Builders' Library — Automatiser des déploiements sécurisés et sans intervention
- Amazon Builders' Library : architecture et exploitation de systèmes résilients sans serveur à grande échelle
- Amazon Builders' Library : l'approche d'Amazon en matière de déploiement à haute disponibilité
- Amazon Builders' Library : l'approche d'Amazon pour créer des services résilients
- Amazon Builders' Library : l'approche d'Amazon pour réussir
- AWSCentre d'architecture

Historique du document

Pour être informé des mises à jour de ce livre blanc, abonnez-vous au fil RSS.

Modification	Description	Date
Publication initiale	Le livre blanc a été publié pour la première fois.	12 novembre 2021

> **Note**
>
> Pour vous abonner aux mises à jour RSS, un plug-in RSS doit être activé pour le navigateur que vous utilisez.

Avis

Les clients sont tenus de faire leur propre évaluation indépendante des informations contenues dans ce document. Ce document : (a) est fourni à titre informatif uniquement, (b) représente les offres de AWS produits et les pratiques actuelles, qui sont susceptibles d'être modifiées sans préavis, et (c) ne crée aucun engagement ni aucune garantie de la part AWS de ses filiales, fournisseurs ou concédants de licence. AWSles produits ou services sont fournis « tels quels » sans garantie, représentation ou condition d'aucune sorte, qu'elle soit expresse ou implicite. Les responsabilités et obligations AWS de ses clients sont régies par AWS des accords, et le présent document ne fait partie d'aucun accord entre AWS et ses clients et ne le modifie pas.

© 2021 Amazon Web Services, Inc. ou ses filiales. Tous droits réservés.

Glossaire AWS

Pour connaître la terminologie la plus récente d'AWS, consultez le Glossaire AWS dans la Référence Glossaire AWS.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.