Du concept d'entr'hébergement à Garage, solution technique pour le stockage réparti

Alex Auvolat, Tom Goldoin, Adrien Luxey-Bitri

Du concept d'entr'hébergement à Garage, solution technique pour le stockage réparti

Alex Auvolat [1], Tom Goldoin[1], Adrien Luxey-Bitri[1, 2]

Deuxfleurs est une jeune association proposant des services numériques libres, membre des CHATONS [3], qui défend ardemment l'idée d'un « Internet convivial ». Nous sommes convaincus de la nécessité d'imaginer de nouveaux futurs plus désirables — que nous incarnons petit à petit, au gré de nos expérimentations. Récemment, notre association a fait parler d'elle avec Garage, un logiciel de stockage géodistribué que nous développons. Ce dernier a reçu le soutien financier de l'Union européenne et a attiré l'attention des cercles techniques. Souvent sommés de nous justifier sur son développement d'un point de vue purement technique, nous pensons qu'il est plus pertinent d'abord de mettre en avant ses conditions d'émergence pour mieux comprendre son essence : de quel futur Garage est-il l'outil ? Nous présenterons d'abord la démographie de l'association, ce qui nous anime, notre action, et enfin notre logiciel, Garage.

Sur le chemin reliant leur campus aux lointains restaurants universitaires, un groupe de jeunes chercheurs de l'équipe WIDE (Irisa, Rennes) avait pour habitude de refaire quotidiennement le monde. Parmi eux, trois doctorants (Adrien, Alex et Quentin) s'adonnaient chacun de leur côté à l'auto-hébergement sur leur temps de

1. Association Deuxfleurs.

2. Laboratoire CRIStAL, université de Lille, Inria, GDS ÉcoInfo, CNRS.

3. Collectif des hébergeurs alternatifs, transparents, ouverts, neutres et solidaires (CHATONS, `https://chatons.org`) est une initiative de l'association Framasoft qui fait suite au succès déraisonnable de leurs services FramaDate, FramaPad, FramaCalc... L'Internet français s'étant relativement bien dégoogliné, il fallait désormais le déframasoftiser [6].

DOI:10.48556/SIF.1024.22.171 **1024** – Bulletin de la SIF, numéro 22, novembre 2023, p. 171–183

sommeil — parce qu'ils aimaient l'informatique, et garder leurs données personnelles à l'œil. Quand l'un d'entre eux revint émerveillé de la 36e édition du *Chaos Communication Congress* (36C3) en janvier 2020, ils surent que le moment avait sonné de la mise en commun. Réunissant leurs amis militants et bientôt d'autres forces vives, ils fondèrent l'association Deuxfleurs.

Un assemblage d'internautes

Une part importante d'entre nous est sortie d'un cursus d'informatique il y a moins de 10 ans : INSA Rennes (et sa très formatrice InsaLan), université/ÉNS, Epitech Lille... Une autre part passe ou est passée par un doctorat (dont un en mathématiques) ; ou encore une conceptrice graphique, une mainteneuse du réseau Tor, l'ex-propriétaire d'une des premières épiceries bio de France, un professeur de lycée, une géologue, une pharmacienne de formation... Ce qui nous lie, c'est notre intérêt commun pour l'informatique — les libertés, joies et potentialités qu'elle recèle. De la variété de nos parcours découle le défi que tout le monde se parle, se comprenne, et arrive à dégrossir un idéal commun. Cet idéal ne peut se construire autour de la technique ; il a donc fallu trouver autre chose : nous avons pour ambition de réaliser ensemble avec Internet et nos vieux ordinateurs, un endroit loufoque et convivial, original et agréable, avant tout humain.

Des idées communes

L'intérêt que nous partageons nous rend sensibles et critiques vis-à-vis des pratiques de l'informatique que l'on observe dans nos domaines d'expertise respectifs et dans la société civile. De nos discussions, il ressort une intersection claire de nos valeurs, qui nous permet d'avancer ensemble dans notre diversité.

Constructivisme sur fond de révolte

Nous adressant à nos professeurs, nous avons encore en tête les cours magistraux qu'ils nous donnaient. En informatique, on nous introduisit aux idéaux hippies du partage des ressources, de la liberté par la cryptographie. On nous initia à la méthode scientifique. Partout on nous enseigna l'éthique. Il n'est alors pas surprenant que nous souhaitions maintenant mettre ces enseignements en pratique : par la création d'une communauté libre, souveraine sur son infrastructure numérique, décidant collectivement de son usage ; par la co-construction et le partage de nos outils ; par la remise en question de nos rapports au monde et aux autres.

Hélas, la découverte des réalités du monde du travail est d'une grande violence. Ses structures nous apparaissent aux antipodes des saines valeurs qui nous furent proposées, et que nous désirons incarner. En entreprise, le *diktat* de l'actionnariat contre l'activité économique réelle. En recherche, *publish or perish*, ou comment se tuer à la tâche, en publiant des incréments qui n'amélioreront jamais que des centres

de données déjà trop gros. Beaucoup de pression, des objectifs soit absurdes soit délétères, aucune agentivité (capacité d'action). Ô professeurs, nous et nos camarades savons, pour l'avoir récemment vécu, le traumatisme qui attend vos étudiants à la sortie de votre giron : doute, révolte, dégoût, et souvent résignation — cynique.

Assumer notre radicalité

Craignant le désespoir, cherchant notre stabilité, les membres de Deuxfleurs s'activent. Certains luttent au sein des structures honnies. D'autres les quittent pour leur préférer la précarité. Toutes et tous vivons pleinement notre radicalité au sein de l'association. Radicalité, pas au sens d'« extrême », mais bien de « racine » : dans l'idée d'adresser les causes, et non seulement de limiter les conséquences des externalités néfastes du numérique. De dépasser l'optimisation, « le consommateur averti » [3], et plus généralement toute action qui tend à un ajustement marginal — à rendre soutenable ce système abhorré.

Parmi les causes de ces externalités, citons l'axiome de la croissance capitaliste (qui conduit à l'extraction/production/consommation coûte que coûte) et le mythe techno-solutionniste (qui fatalise le « progrès », invisibilise le choix que nous faisons de le perpétuer). De ceux-là découle l'idéologie productiviste (accélérer la cadence à mesure que la situation empire), qui enfante l'épuisement lié au travail. Conséquemment, nous manquons de temps de cerveau disponible pour nous cultiver, nous réunir, repenser notre condition. La consommation devient notre seul plaisir. En sous-texte enfin, le désastre environnemental. Il ne s'agit là ni de simples croyances ni de partis pris, mais bien des faits documentés par les sciences humaines : économie, sociologie, histoire, psychologie... Il nous appartient de redonner aux dites sciences leur rôle de baromètre et de guide — qui n'affranchira jamais les sociétés humaines de la nécessité de la réflexion et du choix.

Mettre à distance la technique

Nos sociétés et nos formations sont infusées de pensée ingénieure [1]. C'est notamment pourquoi la tentation est toujours grande de plonger dans des considérations techniques dès que l'on nous présente un problème :
– « Tel collectif aurait besoin de communiquer. »
– « Il pourrait utiliser tel outil ? Quoique porté sur notre infrastructure ... ça ne serait pas compatible. Il faut développer un nouveau logiciel ! »

Ce réflexe fétichiste est malsain, car il élude une grande partie de la réflexion : la définition du besoin dudit collectif en concertation avec les moyens de Deuxfleurs, par exemple. La violence se cache, même, dans le fait de réduire les besoins des gens à un simple cas d'usage de tel ou tel outil. La réalité est plus complexe, évidemment, et c'est ainsi que les usagers finissent par devoir se contorsionner pour rentrer dans le cas d'usage. Le message ici n'est pas qu'il faille en finir avec les outils—mais que les outils n'ont pas réponse à tout, et qu'il est important de régulièrement remettre

en question le bien-fondé et les hypothèses sous-tendant les outils que l'on utilise ou développe.

C'est pourquoi, chez Deuxfleurs, il ne se passe pas deux semaines sans que l'on radote sur « La Convivialité » d'Ivan Illich [9] [4]. Un outil convivial est malléable et s'adapte aux besoins de son usager. Deux exemples typiques sont le marteau et le vélo, qui ne fixent aucune limite aux constructions (resp. déplacements) de leur usager, et ne lui imposent aucun pacte faustien [13]. Comparativement, en achetant une voiture, on se rend complice de l'extractivisme pétrolier, on devient acteur de la pollution de l'air, etc. Illich parle de « monopole radical » quand un outil non convivial en vient à imposer sa vision du monde même aux personnes qui lui résistent. Et en effet, même si l'on refuse individuellement d'acheter une voiture, notre médecin de campagne va quand même finir par déménager à 50 km de là, du fait des modifications de politiques territoriales conséquentes à l'hégémonie automobile. De même en informatique : quand on ne peut plus remplir ses fiches d'imposition sans se passer d'Internet, ou transférer de l'argent sans vérifier son identité avec son intelliphone... Il est de notre responsabilité d'informaticiens d'identifier, d'éviter de reproduire, et de combattre ces mauvaises pratiques [2].

Rester humbles et en mouvement

Le lectorat constatera sans mal le caractère effusif et foisonnant de nos idées, qui peuvent leur donner un rendu imprécis, brouillon, voire contradictoire. Nous assumons ce trait. Nous ne sommes qu'une petite association avec la responsabilité de quelques internautes bienveillants sur les bras. Un endroit parfait pour la discussion, en somme. C'est pourquoi, humblement, nous essayons de remettre profondément en question nos procédés, nos systèmes, nos certitudes, nos évidences... C'est un travail en cours, mouvant, et il est naturel que tout n'y soit pas proprement aligné. Toute critique constructive sera reçue avec joie, et suivie d'une discussion sans doute fascinante pour tout le monde. Quant à la rigueur scientifique — chers collègues — c'est elle qui nous arme en arguments pour convaincre (l'art nous armant en imaginaire pour persuader).

Fabriquer nos propres services numériques

C'est donc avec ces idées que nous entreprenons de « fabriquer » nos propres services numériques, services les plus « émancipateurs » possibles — ce qui fait tout l'intérêt d'Internet à nos yeux. Citons notamment : services pour s'organiser

4. Les démonstrations d'Illich — essayiste plutôt que philosophe — sont quelquefois partiales et venimeuses. Pour ne pas s'écœurer de ce qui constitue tout de même une bonne lecture, on recommande la fiche suivante : `http://attac-gard-rhodanien.e-monsite.com/medias/files/la-convivialite-ivan-illich.pdf`.

(e-mails, agendas, listes de diffusion), pour discuter (conversations instantanées, visioconférences), et pour créer ou publier (suite bureautique collaborative, sites web, blogs fédérés). Si nous parlons de fabriquer, c'est qu'à chaque étape du processus — l'hébergement, la configuration et parfois même le développement du service — nous y voyons une opportunité d'y infuser nos valeurs.

De l'auto- à l'entr'hébergement

En accord avec nos valeurs et notre goût pour l'aventure, nous avons très vite opté pour l'hébergement de nos serveurs à la maison en auto-hébergement. Cette façon de faire est semée d'embûches. En premier chef la persistance des fournisseurs d'accès à Internet (FAIs) à vendre des abonnements dégradés, destinés avant tout à la consommation de contenu et non à des pratiques telles que l'hébergement de services — relevant pourtant des principes fondateurs de l'Internet, qui voulaient que tous les postes soient égaux dans leur capacité à recevoir ou à servir des données. Bien que le passage à la fibre ait largement résolu le souci du faible débit sortant de l'ADSL (A pour *Asymmetric*), qui rendait alors la pratique quasiment impossible, l'essentiel des gros opérateurs Internet persiste à fournir des connexions qui sont bridées d'une autre manière : pas d'adresse IP pleinement attribuée (CGNAT), pas d'IPv6, blocage de ports ne pouvant pas être levé, etc. En conséquence, l'hébergement de services numériques à domicile peut se révéler soit impossible (pas de connexion entrante possible sur les ports standard), soit fortement compromis (e.g. par des changements de configuration intempestifs).

On peut espérer se libérer un jour de ces entraves institutionnelles, par exemple via le mouvement des fournisseurs d'accès à Internet associatifs (surtout porté en France par la FFDN [5]). Néanmoins, on se heurtera encore à des contraintes essentiellement techniques : comment assurer la qualité et la disponibilité de nos services, alors que les coupures d'électricité ou d'accès Internet sont courantes au domicile ? Nous pourrions nous satisfaire d'interruptions plus ou moins fréquentes, sous prétexte qu'il faut payer le prix de ses idéaux, et que c'est de toute façon ce qui fait le charme des solutions artisanales. Il est vrai que la conquête de notre liberté numérique passe par des compromis d'usage [6]. Mais ne condamnons pas le fait-maison à l'incompétence. Qui plus est, chaque interruption de service implique de réveiller la personne administratrice (au travail ou au milieu de la nuit) pour qu'il s'empresse de comprendre et de résoudre la panne.

C'est donc à la fois pour rendre désirable notre vision d'Internet, mais aussi pour ne pas nous épuiser à la tâche, qu'il est nécessaire de maximiser la résilience de notre infrastructure. Si nous n'avons pas les moyens de mettre en œuvre des solutions industrielles, qui impliquent de la redondance à tous les niveaux (arrivées électriques,

5. `https://www.ffdn.org`.

6. Et notamment, la réduction de notre dépendance à l'inénarrable facilité d'utilisation des services de la Silicon Valley.

connexions réseau...), nous disposons par contre d'un solide réseau d'amitiés, et ainsi de plusieurs domiciles largement indépendants où nous pourrions placer différentes machines (cf. figure 1), afin que celles-ci puissent prendre le relai les unes des autres en cas de panne (temporaire ou permanente) dans l'un des lieux d'hébergement.

FIGURE 1. Ancienne grappe de production de Lyon : seul le sur-arrosage peut nous couler. Pour plus d'information sur notre infrastructure : `https://guide.deuxfleurs.fr/infrastructures`

.

Ainsi, le CHATON TeDomum [7] a-t-il défini les contours d'une nouvelle discipline que nous adoptons : l'entr'hébergement. Dans le cadre d'un réseau de confiance forte (e.g. à l'échelle d'une association ou d'un groupe d'amis), l'auto-hébergement est complété par la notion de redondance — géographique, mais aussi humaine, en formant plusieurs personnes à disposer des connaissances techniques pour réparer l'infrastructure en cas de panne.

Stockage distribué sous contraintes

La décision étant prise de relier les machines de plusieurs domiciles pour en faire une grappe de serveurs, il ne reste à résoudre qu'un problème fondamental : celui de la synchronisation de leur état global [11]. Malgré la pléthore de solutions techniques existantes pour « unifier » le stockage de plusieurs machines, nous allons voir qu'aucune ne convenait à notre scénario d'entr'hébergement, caractérisé par la tendance aux pannes, des serveurs bas de gamme, et la piètre qualité des liens les reliant (les communications devant traverser Internet à l'échelle d'un pays voire d'un continent). C'est face à ce manque d'outil technique approprié à notre cas d'usage que nous avons finalement conçu le nôtre : Garage.

7. `https://tedomum.net`.

Lors de notre première tentative de distribution de stockage, nous avons d'abord utilisé GlusterFS [8] — distribué sur trois machines dans un réseau local. C'est un système de fichiers distribué, qui apparaît aux clients directement sous la forme hiérarchique habituelle de fichiers et dossiers présents sur leur machine, comme tout système de fichiers en réseau. Il permet donc de faire fonctionner sans modification des applications habituées à fonctionner en local, en substituant de manière transparente à leur dossier de stockage local un dossier stocké de manière répliquée sur plusieurs machines. Nous avons vite aperçu les limites d'un tel système. GlusterFS est conçu pour fonctionner en centre de données, sur une grappe de machines à haute performance connectées par des liens ultra-rapides. Dans notre cas, nous n'avions que d'anciens PC de bureau peu puissants convertis en serveurs, et cela se faisait grandement ressentir sur les performances de GlusterFS, avec des temps d'accès pouvant atteindre la dizaine de secondes pour la lecture d'un simple fichier.

Il est devenu évident que GlusterFS ne se plaçait pas dans le même univers que nous, et que certains de ses choix de conceptions rendaient quasiment nul l'espoir de le faire fonctionner d'une manière satisfaisante avec les ressources matérielles dont nous disposions. Aux premières loges, nous avons identifié l'abstraction Système de fichiers comme étant problématique en elle-même : en effet, celle-ci dispose d'un grand nombre de fonctionnalités (hiérarchie de dossiers, modification et accès concurrent à des fichiers par plusieurs processus, verrous...) qui rendent son implémentation d'une nécessaire complexité. Les différentes opérations devant apparaître de manière atomique aux clients — et dans certains cas dans un ordre déterminé — afin de garantir la cohérence du système pour les applications, cela implique des coûts de synchronisation importants entre les nœuds de la grappe. En d'autre termes, la moindre opération, même élémentaire, peut nécessiter plusieurs allers-retours de communication entre machines. Ces coûts sont marginaux lorsqu'on dispose du matériel d'avant-garde d'un centre de données, mais à l'inverse, sur du matériel standard comme le nôtre, la latence explose. Rappelons qu'à ce stade, notre déploiement GlusterFS n'était que sur un seul réseau local ; nous n'avons pas envisagé une seule seconde de déplacer des nœuds dans d'autres villes pour assurer une redondance géographique des données. GlusterFS était donc un frein à nos projets d'entr'hébergement.

Garage : se donner les moyens de nos ambitions

Pour réduire lesdits coûts, nous avons abandonné l'abstraction Système de fichiers au profit du Stockage objet : abstraction élémentaire de stockage clé/valeur, où les clés sont des noms de fichiers arbitraires (sans hiérarchie), et les valeurs des fichiers immutables, qui ne peuvent qu'être remplacés entièrement et de manière atomique. Comparé à un système fortement cohérent comme les systèmes de fichiers ou les

8. https://www.gluster.org.

bases de données relationnelles, cette simplicité réduit grandement les garanties et donc, dans un contexte réparti, le coût de la synchronisation entre nœuds. Une des premières entreprises qui popularisa ce concept de stockage d'objets auprès des développeurs n'est autre qu'Amazon en 2006, avec son service *Simple Storage Service* (S3). Un an plus tard, Amazon publiait l'article Dynamo [4], qui présente l'architecture d'un système de base de données clé/valeur réparti basé sur une simplification des systèmes de DHT [9] existants. Les similarités entre les propriétés du service S3 d'Amazon et le système décrit dans l'article Dynamo sont frappantes, au point que deux implémentations libres valideront l'hypothèse qu'un service S3 peut être facilement construit sur l'architecture de Dynamo : Riak CS de Basho [10] et Pithos d'Exoscale [11]. Malheureusement, ces logiciels ont été abandonnés par leurs développeurs et leur dette technique n'a pas permis à d'autres de continuer le développement.

En pratique, ces logiciels démontrent qu'une fois le système de clé/valeur mis en place, il ne reste alors qu'à rajouter un système de fragmentation : les fichiers ne peuvent pas être directement stockés comme valeur au risque de déséquilibrer la grappe à cause de fichiers de tailles disparates. À la place, chaque fichier est découpé en fragments de taille fixe identifiés pas une empreinte (*hash*). C'est la liste des empreintes qui est stockée comme valeur du fichier original, tandis que ses fragments sont répartis sur la grappe.

Une fois le modèle de données mis en place, on peut proposer des primitives de lecture et d'écriture reposant également sur un minimum de coordination, afin de pouvoir les exécuter le plus rapidement possible à l'échelle d'une grande grappe, possiblement géo-distribuée. Ces choix de conception de Dynamo nous ont grandement inspirés lorsque nous avons commencé à planifier le développement de Garage.

Il existe aujourd'hui nombre d'autres solutions de stockage objet activement développées. Citons Ceph [12] et MinIO [13]. L'API de S3 étant devenue un standard *de facto* du fait de la popularité du service, de nombreuses solutions de stockage supportent cette interface. C'est le cas de Ceph et MinIO. Pourtant, leurs développeurs n'ont pas choisi de construire sur le modèle de Dynamo : MinIO a fait le choix d'un système de verrouillage [14], et, Ceph, de par son historique de système de fichier distribué, a un système à base de cohérence forte. Ces alternatives activement développées passent selon nous à côté des possibilités d'optimisation radicales du design à coordination faible proposé par Dynamo. Elles font fondamentalement les

9. *Distributed Hash Table* : un système de partage d'informations populaire en réseaux pair-à-pair, notamment instancié au sein de BitTorrent [10].

10. `https://docs.riak.com/riak/cs/2.1.1/index.html`.

11. `https://github.com/exoscale/pithos`.

12. `https://ceph.io`.

13. `https://min.io`.

14. `https://github.com/minio/dsync`.

mêmes hypothèses que GlusterFS et al. —environnement maîtrisé, à très haute performance, en centre de données —et se dégradent vite en présence de latence, ou lorsqu'elles fonctionnent sur des machines peu puissantes. Par exemple, avec 50 ms de latence entre nœuds, ces logiciels mettent une seconde ou plus à répondre pour des opérations basiques [7]. Puisque c'est cette observation qui a fini de nous décider à produire Garage, penchons-nous dessus en comparant la coordination faible à la notion habituelle de consensus.

Coordination faible versus consensus

Le paysage des classes de calculabilité en systèmes distribués étant d'une grande variété, nous proposerons ici une définition réductrice qui se focalisera sur deux classes de calculabilité disjointes : celle du consensus réparti, et celle du passage de message asynchrone sans consensus, que nous appellerons coordination faible. Quand on parle ici de classe de calculabilité, il s'agit bien d'une notion théorique : si l'on vous donne un « objet » distribué quelconque qui implémente le consensus, vous disposez de propriétés et de garanties qui vous permettent de presque tout faire, y compris implémenter des objets à coordination faible. Mais inversement, si l'on ne vous donne que des primitives à base de coordination faible, il vous est impossible, d'un point de vue théorique, de construire un objet qui implémente le consensus — c'est le fameux théorème d'impossibilité dit « FLP » [5]. En d'autres termes, il est plus difficile de concevoir un système distribué si l'on ne dispose que de la coordination faible, car on est moins bien armé qu'avec le consensus.

Le consensus nous permet d'exprimer n'importe quel système distribué à partir d'une machine à états répliquée : on définit la sémantique du système comme étant l'exécution d'une machine à états avec les opérations réalisées par les nœuds, celles-ci s'appliquant dans un ordre déterminé, ce qui permet de ramener le système au cas d'un algorithme séquentiel qui s'exécuterait sur une seule machine. Plusieurs machines sont responsables de l'exécution de cette machine à états, et sont donc garantes de la persistance de cet état et de la continuité de l'exécution si une ou plusieurs machines tombaient en panne. *A contrario*, les systèmes à coordination faible imposent d'abandonner totalement la vision séquentielle, pour se tourner vers des conceptions où les nœuds peuvent effectuer des transitions d'état de manière indépendante, pas toujours dans le même ordre — avec beaucoup moins de concertation préalable. La complexité réside ici dans le fait que ces transitions peuvent amener les nœuds dans des états divergents : il faut donc définir une sémantique précise des objets distribués en question qui leur permettra de converger à terme vers des valeurs identiques. Une façon rigoureuse de mettre en place cette sémantique passe par les types de données répliqués sans conflits (*Conflict-free Replicated Data Types* ou CRDT) proposé par Marc Shapiro, et leur théorie basée sur les treillis [12]. Cette nouvelle manière de concevoir les systèmes répartis commence à faire son chemin dans les esprits, mais reste encore marginalement implémentée en pratique.

Les coûts du consensus

Cette distinction théorique entre consensus et coordination faible a pour conséquence pratique que les systèmes à coordination faible peuvent être implémentés de manière bien plus efficace : moins de synchronisation, donc moins d'échanges réseau au sein de la grappe par opération, donc moins de latence.

Expliquons maintenant pourquoi les systèmes utilisant le consensus sont sujets à des coûts plus importants. Le théorème d'impossibilité FLP, qui se place purement dans le domaine de la théorie, se base bien sur un modèle de système distribué proche des conditions observables en pratique. Dans ce modèle, les messages émis d'une machine peuvent mettre un temps arbitrairement long à arriver à leur destination, et les machines peuvent tomber en panne à tout moment. Du point de vue d'un des nœuds du réseau, il est donc impossible de faire la différence entre un scénario où un nœud ne répond pas car il est tombé en panne, et un scénario où il a bien répondu mais où les réponses sont retardées arbitrairement par le réseau. Par rapport au problème du consensus, qui consiste fondamentalement à mettre tous les nœuds d'accord sur une seule valeur parmi un ensemble de propositions, on voit facilement comment cela invalide des stratégies simples comme, par exemple, prendre la plus petite des valeurs reçues : en effet, on n'est pas capable de savoir quand l'ensemble des valeurs proposées par les nœuds n'ayant pas connu de panne a bien été reçu. On a par la suite montré que, si le problème du consensus est insoluble dans ce modèle, il est solvable si l'on dispose d'outils supplémentaires, que l'on appelle généralement des oracles : soit un détecteur de faute, qui aide les nœuds à trouver un pair qui n'a pas subi de panne pour lui conférer un rôle spécial de coordination, soit un générateur aléatoire d'une qualité suffisante pour combattre en espérance les aléas du réseau et avoir un espoir de converger vers une valeur commune après un nombre plus ou moins grand de tentatives.

Dans la pratique, les algorithmes qui implémentent le consensus existent et fonctionnent largement bien. Ils sont utilisés dans de nombreux systèmes distribués pour la facilité de coordination qu'ils procurent aux couches applicatives supérieures. Néanmoins, la nécessité théorique d'avoir un détecteur de faute ou de s'appuyer sur de l'aléatoire se traduit en pratique par des solutions qui élisent un *leader* (i.e. les algorithmes Raft et PAXOS). Le rôle du *leader* sera de coordonner toutes les opérations ; son élection pourra prendre un temps arbitrairement long en cas d'instabilité du réseau, et elle devra être recommencée à chaque cas de panne du *leader* en place. Cela implique que tout système conçu en se basant sur la primitive du consensus devra payer les coûts correspondants : élection d'un *leader*, et passage obligé par le *leader* pour toutes les opérations. Dans le cas où le *leader* serait, par exemple, sujet à une latence du réseau particulièrement importante, cela rendrait tout le système extrêmement lent. À l'inverse, les systèmes à coordination faible n'ont pas ce point de centralisation du *leader*, les nœuds pouvant simplement s'échanger les opérations

directement entre eux sans se soucier de l'ordre dans lequel celles-ci sont reçues et traitées aux différents nœuds.

Et Garage dans tout ça ?

Au vu de tous les éléments ci-dessus, nous avons fait un choix radical dans la conception de Garage : il n'y aura pas d'algorithme de consensus dans Garage. Puisque le domaine d'expressivité des systèmes en coordination faible est plus limité, cela implique que nous devrons nous limiter à l'ensemble des primitives de type CRDT que l'on peut implémenter dans ce modèle. Cela ne signifie pas que nous abandonnons entièrement les garanties de cohérence ; simplement que nous exploitons au mieux les largesses permises par l'API S3, où la seule garantie nécessaire est de lire ce qui a été dernièrement écrit — la garantie *read-after-write*, qui peut être facilement implémentée en coordination faible avec des quorums lors des opérations de lecture et d'écriture. Plus précisément, S3 n'est pas une base de données transactionnelle de type ACID, comme le serait une base SQL : il n'y a pas de moyen d'appliquer de manière atomique des opérations sur plusieurs objets en même temps, et il n'y a pas de garantie d'ordre sur la visibilité des modifications sur des objets différents. Il nous a donc semblé que S3 était le candidat idéal pour une implémentation en coordination faible, et que celle-ci serait parfaitement adaptée à notre scénario d'entr'hébergement réparti géographiquement sur des machines de faible puissance. Sa réalisation en pratique nous l'a largement confirmé : le système Garage est en place chez Deuxfleurs depuis plus de deux ans et fonctionne sans faille, il est compatible avec de nombreuses applications que nous hébergeons (en particulier Synapse, notre serveur de messagerie instantanée) Sa tolérance aux pannes a été maintes fois mise à l'épreuve : déconnexions régulières de machines à cause de pannes temporaires d'électricité, disque dur qui rend l'âme, emportant avec lui l'intégralité de son contenu... Tout cela ne sont que des incidents banals sans réelle gravité, parfaitement prévus dans le modèle de Garage et qui n'ont eu quasiment aucun impact sur le service fourni. Par ailleurs, la qualité de service en terme de réactivité des applications est incomparablement meilleure qu'avec GlusterFS, que nous avons pu décommissionner de manière définitive début 2022.

Vers l'infini et au-delà

Nous avons vu que le modèle de la coordination faible pouvait parfaitement s'appliquer au stockage objet, et ce avantageusement ; mais nous ne comptons pas nous arrêter là. En particulier, dans le cadre de notre financement européen NGI [8], nous avons commencé à explorer la possibilité de stocker des boîtes mail dans Garage, ce qui nous a conduit à concevoir puis implémenter une seconde API de stockage, *Garage K2V*, plus appropriée au stockage de courtes entrées telles que des en-têtes de messages ou des journaux d'opérations. Notre serveur de stockage de mail est encore à l'état de prototype, mais nous comptons l'étoffer à l'avenir avec l'espoir de le faire

fonctionner en production prochainement. Par ailleurs, nous avons conçu l'API K2V pour être généralisable : elle expose de manière native les propriétés de coordination faible de Garage, et est donc appropriée pour la conception de toute application pouvant être exprimée dans le domaine des CRDT ou de la cohérence à terme. Nous avons aussi conçu cette API pour qu'elle supporte le stockage de contenu chiffré, ce qui nous a paru être particulièrement important dans le cas des e-mails, où l'on souhaite garantir la confidentialité des données des utilisateurs. Ces multiples propriétés de K2V, ainsi que l'usage d'un stockage objet tel que S3 à la place du système de fichiers, dessinent les contours d'une nouvelle façon de concevoir des applications dans des systèmes distribués, que nous souhaitons continuer à explorer et à appliquer dans différents domaines.

Conclusion

Nous qui enjoignions tout à l'heure à se distancier de la technique, voilà que nous avons présenté en long, en large et en travers un outil de notre cru. Ingénieurs irrécupérables, alors ? Pas si sûr. Car c'est en voulant construire les services numériques du futur des personnes, que l'on en vient à réaliser l'absence d'outils appropriés. Ce futur que l'on imagine, il est issu de nos observations, réflexions et lectures, de l'Histoire, de nos espoirs. Les contraintes que l'on pose à notre activité numérique sont humaines, pas techniques : c'est à titre personnel que nous honnissons l'achat de matériel neuf. Puisque l'incessant ballet construction/destruction du matériel de pointe est la voie la plus certaine pour vider nos mines de leur dernier filon de cuivre et faire de l'informatique une idée du passé, y compris dans sa dimension émancipatrice. Alors des outils, nous en proposerons peut-être d'autres ! Car nous aimons les espaces de libre expression et d'accès aux savoirs, qui sont comme chacun sait un préalable à la démocratie. Si vous ne voyez pas là une utopie déjà déchue, alors nous vous donnons rendez-vous dans ces cabanes expérimentales des communs qui foisonnent sur Internet — et que nous investissons, occupons, et faisons vivre fort joyeusement chaque jour.

Références

[1] Aurelien Barrau. Aurelien Barrau à CentraleSupelec : A-t-on encore besoin d'ingénieurs ?, November 2022.

[2] Guillaume Carnino and Clément Marquet. Les datacenters enfoncent le cloud : enjeux politiques et impacts environnementaux d'internet. *Zilsel*, 3(1) :19–62, 2018.

[3] Jean-Baptiste Comby. L'individualisation des problèmes collectifs : une dépolitisation politiquement située. *Savoir/Agir*, 28(2) :45–50, December 2014.

[4] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo : Amazon's highly available key-value store. *ACM SIGOPS operating systems review*, 41(6) :205–220, 2007.

[5] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2) :374–382, 1985.

[6] Framasoft. Déframasoftisons Internet !, September 2019.

[7] Garage team. Confronting theoretical design with observed performances, September 2022.

[8] Tom Goldoin. NGI Pointer subventionne Deuxfleurs, September 2021.

[9] Ivan Illich. *La Convivialité*. POINTS, Paris, August 2014. Première parution : 1973.

[10] Andrew Loewenstern and Arvid Norberg. DHT Protocol. BitTorrent Enhancement Proposal BEP5, BitTorrent, January 2008.

[11] Michel Raynal. *Fault-Tolerant Message-Passing Distributed Systems : An Algorithmic Approach*. Springer International Publishing, Cham, 2018.

[12] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. Conflict-Free Replicated Data Types. In Xavier Défago, Franck Petit, and Vincent Villain, editors, *Stabilization, Safety, and Security of Distributed Systems*, volume 6976, pages 386–400. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[13] Yuval Noah Harari. *Sapiens. Une brève histoire de l'humanité*. Albin michel edition, 2015.