Design technique

Options de traitement

Plusieurs architectures techniques de traitement sont possibles pour réaliser un archiveur de cours massif entre deux plates-formes :

- A - Une architecture monolithique séquentielle asymétrique : Un seul traitement exécute séquentiellement la masse d'actions de sauvegardes et d'envoi dans l'environnement d'archivage. L'action d'archivage est terminée quand le traitement a traité tous les cours à transmettre. Le traitement est asymétrique en ce sens que la plate-forme émettrice des cours à archiver est pilote de TOUT le traitement.
- B - Une architecture asynchrone parallèle asymétrique : plusieurs processus distribués (tâches) sont planifiées dans le temps pour effectuer un déplacement unitaire d'une unité de sauvegarde. L'action d'archivage est terminée lorsque tous les processus planifiés ont abouti. Le traitement est asymétrique en ce sens que la plate-forme émettrice des cours à archiver est pilote de TOUT le traitement.
- C - Une architecture symétrique : La plate-forme émettrice prépare la partie “sauvegarde” (potentiellement sur la base des sauvegardes automatiques), et génère une liste de cours à réintégrer avec les métadonnées nécessaires (catégorie d'arrivée, posttraitements éventuels). La plate-forme réceptrice est responsable de son côté de la réintégration.

Discussion

- A : Ce type d'architecture pose le problème de la résistance à la masse et sa fiabilité sur de très grands corpus. Le procédé étant monolithique et séquentiel, une erreur fatale arrête tout le traitement en cours. Il est nécessaire d'assurer des points de contrôle à chaque unité, et reprendre de manière idempotente un traitement interrompu en sautant les items terminés. De plus, il n'est pas commode de suspendre un processus monolithique pour, par exemple, laisser une plage de fonctionnement “sans charge induite” sur la journée pour reprendre le soir le traitement interrompu. Une telle fonctionnalité suppose précisément que le traitement n'est pas conçu de manière monolithique, mais au contraire, “granulaire et incrémental”. Le traitement séquentiel monolithique suppose également un temps de traitement long, voire très long pour des grosses volumétries de cours. Si la distribution en “taille et complexité” de ces cours est très variable, la durée d'une unité de traitement, à un moment donné, n'est pas prévisible. Les débordements de temps sur une plage horaire à protéger sont donc possibles.
- B : En distribuant le traitement sur un archivage unitaire d'un seul cours, le risque d'une erreur globale interrompant tout le processus d'archivage est moindre. Surtout si les tâches d'archivage sont lancée de manière effectivement indépendantes les unes des autres (Ceci n'est pas garanti par le séquencement de tâches ad-hoc dans le même “tour de cron” d'ailleurs). De ce fait, on peut avoir plus confiance en un traitement plus complet de “tout ce qui a pu passer”, et par contre des reliquats ponctuels présentant une anomalie de sauvegarde ou de récupération, ou des non traitements suivant une anomalie du groupe de tâches ad-hoc traitées. Avec un traitement distribué par unités, il est plus facile de suspendre la programmation des tâches pendant certaines heures. Il reste cependant difficile de prévoir si une tâche particulièrement volumineuse, lancée en fin de plage autorisée, ne viendrait pas déborder en charge sur la plage de fonctionnement à garantir. Dans ce cas de figure, la plus grande difficulté est celle d'organiser un “monitoring” fiable de l'état en cours, et d'offrir

suffisamment d'outils de manoeuvre pour intervenir sur le processus, l'arrêter, nettoyer les tâches en attente, etc.
- C - Dans le cas d'un traitement symétrique, il n'est pas raisonnable de penser à deux traitements monolithiques successifs. Gérer l'information de progression sur deux traitements coordonnés de chaque côté relève d'une complexité algorithmique trop lourde, et d'une logique de récupération de faute inabordable. Un traitement symétrique est donc nécessairement distribué par unité de traitement, et événementiel pour coordonner les deux systèmes de tâches qui se font face dans la plate-forme émettrice et la plate-forme réceptrice. Là encore, se baser sur un système de déclencheurs événementiels amène une complexité de surveillance et de monitoring de l'état en cours qui peut se révéler non opérationnel sur le terrain.

Corollaires à la discussion

Dans tous les cas de figure, et dans tous les scénarios, on voit que le monitoring de la situation supposera un suivi des opérations cours pas cours à partir d'un registre de suivi d'archivage. Ce registre sera initialisé par une liste de cours à traiter, identifiés pour être la cible de l'opération d'archivage. L'archivage de chaque cours devra y être suivi pour pouvoir fournir une information d'ensemble sur le processus. La séquence d'archivage générale est sous sa forme la plus générale :

1. Un prétraitement (si nécessaire)
2. Une sauvegarde aux conditions données par le contexte
3. Un transport de l'archive
4. Une restauration conforme
5. Un post traitement (si nécessaire)
6. Une destruction du volume initial (sera souvent le cas, mais pourrait être optionnelle)

Dans la mise en oeuvre générale il faudrait pouvoir se passer de l'étape de pré-traitement. En effet cette étape modifierait le cours d'origine et pourrait rendre difficile la réentrance du processus en cas d'erreur ultérieure dans la séquence.

Etats généraux du traitement

Le registre d'état doit pouvoir précisément marquer la phase d'action atteinte par un traitement unitaire d'archivage. Les états devraient suivre peu ou prou le processus ci-avant.

- Etats globaux de session
  - % d'exécution
  - Session terminée
  - Nombre d'erreurs
  - Nombre d'unités archivées
- Etat pour une unité de traitement
  - Préparation achevée
  - Sauvegarde achevée
  - Transport achevé
  - Redéploiement achevé
  - Post-traitement achevé
  - Nettoyage achevé (process achevé)

Mécanismes de transport

Il existe plusieurs mécanismes de transport d'une archive de cours pour son déplacement :

- A - Transport par chemin absolu dans le serveur : L'émetteur et le récepteur partagent le même environnement général de fichiers (même montage) auquel ils ont accès (l'émetteur en écriture et le récepteur en lecture).
- B - Transport par services MNET : Certains plugins (block_publishflow) ont organisé un transport de cours par des services MNET. MNET étant déprécié, il ne convient plus de baser des nouveaux développements sur cette base.
- C - Transport par Web Services Moodle (standard) : Moodle prévoit en standard des webservices d'upload de fichiers à déplacer ensuite dans une “aire de fichiers moodle” d'usage. Il n'existe pas par contre de fonctione WS permettant une restauration COMPLETE
- D - Transport par Web service dédié : Les composants écrits pour l'archivage présentent leurs propre webservices de chargement.

Sélection des cours d'une session d'archivage

Dans le premier jet, un archivage vise une catégorie de cours de moodle, en général millésimée. Une disponibilité récurrente de la fonction d'archivage suppose une bonne cohérence entre le pointage de la zone à archiver et les dates envisagées pour exécuter l'archivage. Dans l'idéal nous avons un mécanisme “glissant” qui pour chaque session sait ce qu'il a à faire et comment le faire.

Dans une deuxième approche plus lointaine on devrait pouvoir archiver un ensemble discontinu de cours réparti dans sous des racines multiples.

Transposition et relocalisation dans la cible

Dans le premier jet, le déplacement vers une plate-forme externe pourrait s'accommoder d'une réimplantation “à l'identique”, c'est à dire, en respectant le chemin de classement de l'origine. Cette hypothèse de “non-transposition” ne peut fonctionner si la cible et la source sont les mêmes.

De toutes façons, dans ce dernier cas le protocole d'archivage :

1. serait fonctionnellement dégradé à un simple “déplacement du cours” avec post-traitement
2. ne conduirait pas à une grande pertinence puisque l'objectif de l'archivage est l'allègement de la masse de données stockée dans la plate-forme opérationnelle.

Revenir à l'index du plugin - Revenir à l'index des plugins - Revenir au catalogue