

****

|  |  |
| --- | --- |
| L’intitulé de la mission | * Déployer des outils de mesure de la consommation énergétique des   serveurs/VM   * Proposer une visualisation sous forme de métaphores parlantes des métriques collectées en lieu et place des Watts |
| l'auteur du rapport | Slim Khiari |
| le nom de la formation suivie et l'année | IATIC 4 – 2021/2022 |
| le lieu du stage | La côte basque |
| les dates du stage | 9 Mai 2022 - 9 Aout 2022 |
| Encadré par | * M. Phillipe Roose |

Le lexique

|  |  |
| --- | --- |
| API | Application Programming Interface |
| RAPL | Running average power limit |
| CPU | Central Processing Unit |
| DSL | Domain-Specific Language |
| SDLC | Software Development Life Cycle |
| BBCP | Behavior-Based Consumption Profiles |
| QoS | Quality of Service |
| SCP | Systèmes Cyber-Physiques |
| HTTP | Hypertext Transfer Protocol |
| SD | Service Directory |
| TDP | Thermal Design Power |
| DRAM | Dynamic Random Access Memory |
| ETL | Extract-transform-load |
| KVM | Kernel-based Virtual Machine |

## Introduction

Dans le cadre de ma formation d’ingénieur informatique, intitulée IATIC (Ingénierie des Architectures Technologiques de l'Information et de la Communication) de ISTY (Institut des Sciences et Techniques des Yvelines) de l’UVSQ (Université Saint-Quentin en Yvelines), j’ai été amené à réaliser un stage d’une durée de 3 mois à partir du 9 Mai 2022. Il est effectué au sein de LIUPPA (Laboratoire Informatique de l’Université de Pau et des Pays de l’Adour) sur le site de la Côte Basque.

Le thème général de mon stage est le green IT. Il s’agit d’un ensemble de techniques ayant pour objectif de limiter les conséquences environnementales des technologies de l’information et de la communication. Le green IT est apparu lorsque l’infrastructure informatique est devenue l’une des causes de l’augmentation de l’effet de serre en raison de sa consommation relativement importante de l’énergie électrique.

Mon stage consiste à collecter puis afficher graphiquement les métriques de consommation énergétique, d’un serveur et d’une machine virtuelle en utilisant une API spécifique pour la mesure de la consommation énergétique et un outil de visualisation de données sous forme de graphiques pour la mise en place d’un tableau de bord final résumant les données collectées pour une machine donnée. Comme le répertoire intel:rapl, fournissant les métriques que nous cherchons à obtenir, RAPL, n’est pas disponible sous une machine virtuelle, la principale difficulté de ce stage est donc de chercher une solution permettant de les récupérer sous ce type de machine.

Je commencerai d’abord par présenter le laboratoire et son secteur d’activité. Après la présentation des missions du stage, je détaillerais les travaux effectués. Pour cela, dans un premier temps, j’aborderai la partie de la conception en justifiant le choix des outils utilisés, comme par exemple l’API Scaphandre qui permet d’effectuer les mesures des métriques de consommation énergétique, et en expliquant leurs fonctionnant. Dans un deuxième temps, j’expliquerai la mise en place de l’ensemble des outils utilisés et la présentation des métriques présentes dans le tableau de bord obtenu.

Présentation du laboratoire

Présentation générale

Le laboratoire LIUPPA fait partie de l’université UPPA. Il a participé à la réalisation de plusieurs projets non seulement européens sur les SCP, le génie logiciel, le développement d’une plateforme de gestion de l’énergie, mais aussi internationaux avec l’Algérie, le Mexique, le Pays Basque Espagnol.

Ses axes de recherche sont focalisés principalement sur les besoins et les enjeux d’une société numérique dans laquelle les réseaux ont occupé une place non négligeable dans nos activités de tous les jours. Ce phénomène a poussé les systèmes informatiques à devenir de plus en plus complexes à travers non seulement la masse des données de tout types, fortement délocalisées, qui augmente sans arrêt, mais aussi l’évolution des besoins des différents usagers.

Secteur

LIUPPA cherche des solutions sur deux domaines différents avec des préoccupations liées à la sécurité, au traitement de l’image et du signal, à la visualisation, aux systèmes distribués, et à l’interaction et l’adaptation. Le premier domaine est les sciences et les technologies de l’information autour des traitements de l’information, de la connaissance, et du web. Le second principal domaine est le génie logiciel autour de l’ingénierie des modèles, des services et des architectures logicielles.

Le savoir faire du laboratoire se décompose en 3 parties. La première partie concerne les systèmes d’information à travers l’ingénierie des documents électronique, la sémantique des contenus (l’extraction et indexation d’information spatio-temporelle, la recherche par contenus), les interfaces multimodale, les interfaces intelligentes, l’ingénierie collaborative et le E-learning. La deuxième partie concerne le génie logiciel et les systèmes distribués via la modélisation, la vérification, la validation et le codage de systèmes, la conception et le déploiement de solutions à base de composants et d’agents logiciels, les définition et la mise en place de politique de sécurité pour les réseaux et les bases de données, et enfin le Context Aware Middleware. Concernant la troisième partie du savoir faire, il y a les SCP à travers l’analyse des données et apprentissage automatique, la représentation des connaissances, les réseaux et protocoles, la gestion des évènements et contrôle des ressources dans les réseaux de capteurs et la protection de la vie privée sur les réseaux sociaux.

Le laboratoire positionne son projet scientifique dans un champ applicatif bien précis qui est la gestion des systèmes d’information et des architectures des SCP. Les SCP sont des systèmes connectés dans lesquels chaque élément est en interaction avec tous les autres éléments. Ainsi, chaque élément contribue à constituer la complexité de cet ensemble d’éléments. Ces systèmes SCP, composés des systèmes logiciels, de capteurs et d’actionneurs, permettent de mettre en relation le monde physique au monde du numérique du traitement de l’information.

Organigramme du laboratoire



Service où est effectué le stage

Le stage effectué est réalisé au sein de l’équipe des traitements des informations pour l’adaptation de l’interaction au contexte et à l’utilisateur (T2I). Cette équipe traite plus particulièrement les éléments externes et contextuels d’un ESCP.

Le cadre du stage

Description de différentes équipes composantes le laboratoire LIUPPA

Le laboratoire LIUPPA est composé de 3 équipes différentes. La première est l’équipe spécialisée dans l’architecture des Systèmes Cyber-Physiques. Elle a pour mission d’élaborer des recherches au tour de la gestion de la sémantique des données à l’intérieur d’un système cyber-physique, la conception des architectures systèmes, et la maitrise des échanges entre les équipements. La seconde équipe est l’équipe spécialisée dans les traitements des informations pour l’adaptation de l’interaction au contexte et à l’utilisateur. Elle a pour mission de conception, d’implémentation et de déploiement des applications génériques, interactives et adaptatives, permettant le traitement des différentes données ayant des origines de provenance différentes, qui nécessitent des modèles de représentation et des méthodes d’accès originaux. Le thème général des recherches de cette équipe est la valorisation de l’information et la facilitation des interactions de l’usager. Enfin, la troisième équipe est l’équipe de l’ingénierie dirigée par les modèles. Cette équipe s’intéresse aux langages de spécification et de modélisation semi-formelle pour la conception de logiciels de qualité. Elle mène des recherches sur la conception des logiciels de natures variées.

Le projet général et les missions à réaliser

Le projet sur lequel j’ai eu l’occasion de travailler, ayant pour thème l’informatique responsable, se diviser en 3 parties majeures. D’abord, la partie sur le DSL réalisé par un étudiant-doctorant, Jorge Andrés LARRACOECHEA. En effet, il travaille principalement sur la création d’un DSL dans le but d’effectuer un profilage du comportement des logiciels à partir des phases initiales du SDLC et d’une méthodologie assistée par un outil personnalisé pour rendre la construction et l’évaluation énergétique des profils plus faciles. Son DSL, nommé BBCP, permet aux concepteurs et architectes de logiciels de générer des descriptions du comportement de n’importe quelle unité de logiciel. BBCP permet également de générer des études diachroniques des logiciels, d’estimer une consommation de ressources locale (profil unique) et globale (collections de profils), et de générer une estimation de la consommation d’énergie et un score énergétique (local ou global). La seconde partie est réalisée aussi par un étudiant-doctorant, Hernán Humberto Álvarez Valera. Il étudie les méthodes nécessaires pour déployer et redéployer des composants logiciels, c’est-à-dire des microservices, afin d’économiser de l’énergie. Afin d’évaluer ces méthodes, il a développé un simulateur capable d’effectuer une modélisation des différents scénarios distribués. Ces méthodes prennent en compte la relation entre la consommation d’énergie, l’utilisation des composants matériels, et les différentes définitions de QoS des applications. Enfin, la troisième partie, constituant donc le sujet de mon stage, est la partie sur l’étude des données de la consommation énergétique réelles. Les missions de ce stage se composent en 2 principales parties. D’abord, la 1ère mission consiste à déployer des outils de mesure et de stockage de consommation énergétique (l’API Scaphandre, powerAPI, Prometheus et Influxdb), dans le but donc de collecter, en temps réel, des métriques de consommation énergétique et un ensemble d’informations qui y sont liées, provenant de différentes sources (serveurs, machines virtuelles et applications). Par la suite, à l’aide des outils spécifiques de visualisation (Grafana et Chronograf), un langage de requête intégré conçu pour Prometheus et un autre pour InfluxDB, la 2ème mission consiste à visualiser ces données sous forme de graphiques. Ces graphiques seront présentées sous forme d’un tableau de bord efficace en termes de compréhension et d’analyse. Les graphiques obtenues seront facilement compréhensibles, par les futurs usagers du tableau de bord, grâce à des métaphores parlantes de ces consommations permettant de remplacer l’unité de mesure Watt par une autre unité « du monde réel ».

Aspects techniques – conception et réalisation

Conception

Dans cette sous partie, je vais parler essentiellement parler sur le planning que j’ai mis en place lors de mes premières semaine du stage, ainsi que des différents outils techniques que j’ai utilisées afin d’atteindre mon objectif fixé au départ qui est l’obtention d’un tableau de bord résumant la consommation énergétique des serveurs et d’une machine virtuelle.

Répartition du temps de travail et des tâches

Afin de bien mener mes taches, j’ai d’abord réalisé un planning résumant les objectifs à atteindre pour chaque semaine. Le planning, que j’ai fixé, a évolué tout au long du stage, à travers les mises au point hebdomadaires effectuées avec mon tuteur professionnel M Phillipe Roose. Pendant ces mises au points, je présente ce que j’ai réalisé dans la dernière semaine passée, d’autres axes de recherche et d’améliorations m’ont été proposés comme la recherche d’autres sources de données de consommation énergétique des serveurs dans le but de les comparer avec les sources de données de consommation énergétique déjà obtenues, et la résolution de quelques problèmes techniques rencontrés lors de mon stage comme le fait de ne pas pouvoir accès directement aux hyperviseurs de l’IUT pour des raisons de sécurité. Voici donc un tableau résumant, d’une manière hebdomadaire, mon travail effectué au sein du laboratoire pendant les 8 premières semaines du stage.

|  |  |
| --- | --- |
| La semaine du | Descriptions |
| 9 Mai 2022 | La première semaine du stage est principalement dédiée à la recherche des APIs que seront utilisés pendant le stage. Cela a commencé d’abord par la compréhension du problème que pose une vm afin de mesurer sa consommation énergétique (e raison de l’abscence du répertoire RAPL permettant de fournir les métriques de consommation énergétique), pour enchainer par la suite par la recherche des outils dédiés à ce type de mesure et enfin le choix entre les 2 outils trouvés Scaphandre et powerAPI. |
| 16 Mai 2022 | Une fois les outils de mesure trouvés, j’ai commencé par regarder le fonctionnement et le lancement de powerAPI. Malheureusement, le module de powerAPI permettant de mesurer les métriques de consommation énergétique d’une vm, VirtualWatts, ne fonctionne pas avec une version du kernel récente (5.15.0-33-generic). Après avoir installer un hyperviseur KVM et mettre en place une vm, Je me suis donc tourné vers l’autre API, Scaphandre.  Malgré quelques problème rencontré au départ lié au nombre de CPUs virtuels de la vm, cette API a bien fonctionné et donc les différentes métriques proposées par cette API ont bien récupérées. |
| 23 Mai 2022 | Une fois j’ai obtenu les métriques de consommation énergétiques d’une vm, je me suis directement lancé à chercher comment afficher ces métriques, en temps réel, d’une manière graphique. Pour cela, j’ai cherche les outils de visualisation et de stockage de données permettant d’afficher les données d’une source de données en temps réel.  Grafana et Prometheus ont donc été choisis.  J’ai donc bien réussi à réaliser, à titre d’exemple, un tableau de bord, composé seulement de 2 graphiques, qui sera amélioré en termes de nombre de métriques et d’organisation, dans les semaines à venir.  Afin de concrétiser les étapes choisis, j’ai réalisé un tutoriel détaillé, avec quelques remarques importantes liées à la configuration de la vm, permettant de mettre en place un dashboard de visualisation des métriques de consommation énergétique en utilisant l’api Scaphandre. |
| 30 Mai 2022 | Après avoir rendu le tutoriel de la semaine précédente, j’ai passé la majorité du temps de cette semaine, à chercher les différentes possibilités d’amélioration du dashboard obtenu en terme de conversions de consommation énergétique en métaphore plus parlantes. Pour cela, j’ai réalisé un tableur afin de les résumer. De plus, j’ai commencé à chercher à comprendre le fonctionnement d’InfluxDB. Ensuite, j’ai mis en place une base de données InfluxDB sur la vm du laboratoire (nommée « greenit ») afin de préparer la réception des données de la consommation énergétique depuis les deux serveurs du site de Pau ; kvm-0 et kvm-1. J’ai installé aussi Chronograf afin de visualiser des données de la base de données InfluxDB créée et donc exécuter les requêtes InfluxQL pour filtrer les données reçues. |
| 6 Juin 2022 | La récupération des données depuis la bare-metal du site de Pau (sous InfluxDB) et leurs affichage dans le dashboard avec Chronograf ont bien étés réalisés pendant cette semaine. J’ai reçu 3 métriques primordiales des 2 serveurs; la consommation du DRAM en watts, la consommation globale, et le TDP. |
| 13 Juin 2022, 20 Juin 2022, et 27 Juin 2022 | Les principales mission de ces 3 semaines sont la rédaction du rapport du satge, l’amélioration des dashboards obtenus avec d’autres nouvelles conversion de consommation énergétique en métaphores plus parlantes, la recherche des manières de mesure de la consommation énergétique des applications (de bureau et web), et la préparation des diaporamas afin de présenter, à mon tuteur professionnel M Phillipe Roose, ce que j’ai réalisé pendant ces 8 premières semaines du stage.  De nouvelles métriques, des deux serveus, concernant le CPU et le débit réseau ont reçus pendant cette période. |

Outils utilisés lors de mon stage

Dans cette sous-partie, je vais présenter et expliquer le fonctionnement des différents outils que j’ai utilisés afin d’aboutir à un tableau de bord final permettant d’afficher les métriques de consommation énergétique. Pour cela, je vais d’abord commencer par l’outil permettant uniquement de mesurer ces métriques, par la suite j’aborderai la partie des outils de stockage des données et enfin les outils de visualisation graphique choisis.

L’outil de mesure des métriques de consommation énergétique

Scaphandre

Scaphandre est une API ayant pour objectif de mesurer les métriques de consommation d’énergie des services technologiques. Il permet de fournir l’énergie consommée par un seul processus sur un serveur ou une machine virtuelle.

Afin de comprendre facilement le calcul de la quantité d’énergie consommée par un processus unique exécuté sur une machine (serveur ou machine virtuelle), nous pouvons imaginer un rectangle contenant des traits représentant chacun le temps de calcul allouées à chaque processus. Pour les machines travaillant sur différents processus en même temps (c’est-à-dire travaillant sur un processus pendant un court intervalle de temps, puis un autre intervalle de temps et ainsi de suite), nous appelons ces intervalles les jiffies. Chaque processus conserve un total cumulé du nombre total de jiffies qui lui sont alloués. Ainsi, pour connaitre la quantité de ressources d’une machine utilisées par un processus donné, il suffit de connaitre le nombre total de jiffies utilisés. Afin de connaitre donc la puissance utilisée par un processus donné, nous comptons d’abord les jiffies utilisées par ce processus lorsqu’il est en cours d’exécution. Ensuite, pour chaque jiffy, nous vérifions la quantité d’énergie consommée à ces moments précis. En regroupant toutes les lectures de puissance pour tous les jiffies sur un intervalle de temps bien précis, nous pouvons arriver à un chiffre utilisable pour la quantité d’énergie utilisée en termes de wattheures.

Sans déployer Scaphandre, ou une API du même type, il est bien évidemment possible de savoir la partie la plus importante des ressources d’une machine en termes d’utilisation par un processus donné. Par contre, si nous voulons trouver la quantité d'énergie utilisée par processus, nous devons obligatoirement savoir quelle quantité d'énergie est utilisée par la machine en termes absolus.

Les informations liées à la consommation énergétique sont extraites à l’aide de la technologie RAPL. Il s’agit d’une technologie intégrée dans les processeurs Intel et les processeurs AMD, ayant l’architecture x86, produits après l’année 2012.

Le module powercap, se trouvant entre Scaphandre et ces données de consommation énergétique, écrit la consommation énergétique dans des fichiers, qui seront, par la suite, lus par Scaphandre. Après avoir été lus, Scaphandre stocke ces données dans des tampons, et permet, par conséquent, un traitement supplémentaire à travers les différents exportateurs.

Cette API, Scaphandre, est composé de deux parties principales ; un capteur et un exportateur.

La 1ère partie, le capteur, est destinée donc à obtenir les métriques de consommation d’énergie de l’hôte, et les mettre à la disposition de l’exportateur. Par exemple, PowercapRAPL obtient et transforme les métriques provenant du noyau powercap Linux, qui sert d’interface pour obtenir les données de la fonction RAPL des processeurs x86. Le capteur PowercapRAPL collecte d’abord les mesures de consommation d’énergie, puis il les convertit en mesures de consommation d’énergie. À chaque fois que l’exportateur, comme l’exportateur prometheus, demande une mesure, par exemple à chaque fois qu’une demande arrive, que nous appelons demande i, PowercapRAPL lit les valeurs des compteurs d’énergie de powercap. Ensuite, ce capteur les stocke et effectue le même fonctionnement pour les statistiques d’utilisation du processeur et pour chaque processus en cours d’exécution sur la machine en temps réel. Maintenant, entre 2 demandes de mesures, demande i et demande i+1, nous avons la possibilité d’obtenir le sous-ensemble de consommation d’énergie lié au PID d’un processus. Donc pour savoir ce qu’un service consomme en réalité, il suffit de joindre la consommation de tous les PID associés. Il est important de noter que cette fonctionnalité n’est pas disponible directement pour les machines virtuelles. Dans ce cas, avec l’exportateur QEMU, il faudra d’abord exécuter Scaphandre sur l’hyperviseur (bare-metal), et ensuite rendre les métriques de la VM disponibles.

Concernant la 2ème partie, l’exportateur est destiné à demander aux capteurs d’obtenir de nouvelles mesures et de les stocker pour une utilisation potentielle ultérieure. L’exportateur permet donc d’exporter les métriques actuelles. Par exemple, l’exportateur prometheus expose les métriques sur un point de terminaison HTTP, pour etre extraites par une instance prometheus. Alors que l’exportateur stdout expose simplement les métriques sur la sortie standard. Concernant l’exportateur qemu, il est destiné à collecter des métriques liées à l’exécution de machines virtuelles sur un hyperviseur Qemu/KVM. Ces métriques seront mises à la disposition de chaque machine virtuelle en exécutant le capteur PowercapRAPL avec l’option –vm. L’exportateur Qemu place les métriques de la machine virtuelle dans des fichiers de la même manière que le module de noyau powercap le fait. Il imite ce comportement afin que le capteur puisse agir de la même manière qu’il le ferait sur une machine non virtuelle.

Pour conclure sur cette partie, afin de mesurer et donc suivre cette quantité d’énergie utilisée par la machine elle-même, nous aurons besoin d’un capteur. Ce qui nous permettra donc de nous fournir un ensemble d’informations portant sur la quantité d’énergie utilisée en watts. Maintenant, à partir de ces données en wattheures, nous pouvons, dans la partie de la visualisation des données, les convertir en données plus concrètes à l’aide des requêtes promQL ou influxQL Ce qui nous permettra d’avoir une visualisation graphique des données assez efficace et significative.

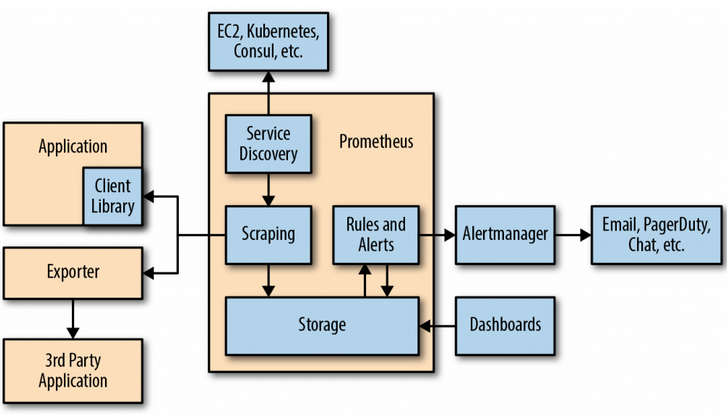
Les outils de stockage des données

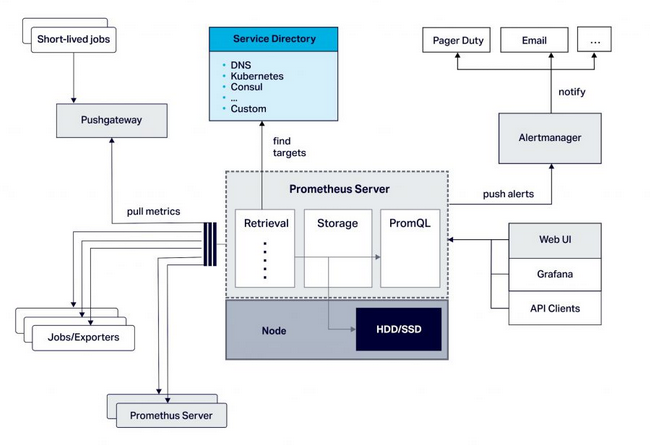
Prometheus

Prometheus est une application logicielle gratuite créée en 2012 par SoudCloud. Depuis 2016, cette application fait partie de la Cloud Native Computing Foundation. Elle sert à surveiller toute série chronologique purement numérique en les enregistrant et traitant. Elle rassemble, organise et stocke des métriques à partir des plates-formes d’infrastructure, des applications et des services, en prenant en compte aussi les métriques des points de terminaison HTTP. Prometheus peut également collecter les métriques de sa propre santé et de les surveiller. Cet écosystème est donc une solution qui englobe à la fois la gestion d’un modèle de données multidimensionnel et la collection de métriques évolutive tout en gardant une certaine simplicité opérationnelle.

Elle est développée avec le langage de programmation Go et possède son propre langage de requête, promQL, ce qui facilitera la manipulation et l’analyse des données collectées. Prometheus est un écosystème ayant plusieurs composants fonctionnant ensemble dans le but de générer des rapports sur les performances d’un système. Ainsi, Prometheus collecte et stocke les métriques des applications qui exposent les métriques dans un format de texte brut à travers des points de terminaison HTTP.

Les schéma ci-dessous sont complémentaires. Ils permettent de mieux comprendre l’architecture de l’écosystème prometheus que nous avons utiliser afin de construire notre tableau de bord.





Le premier composant de cet écosystème est le serveur Prometheus. Il s’occupe à la fois du stockage des métriques, et de la planification des taches de surveillance en interrogeant les sources de données à une fréquence d’interrogation prédéfinie. Les tâches de surveillance sont gérées à travers le fichier de configuration YAML, et configurées en utilisant la directive scrape config.

Dans le 1er schéma ci-dessus, nous avons un élément qui se trouve dans le serveur Prometheus qui s’appelle SD. En effet, afin de connaitre les cibles depuis lesquels la collecte des données se fera, Prometheus s’appuie sur plusieurs mécanismes de SD. Par exemple, nous pouvons avoir une SD basée sur des fichiers que les implémentations personnalisées SD peuvent utiliser. Ceci est donc possible en gérant directement un fichier de type YAML contenant toute une liste de cibles. De plus, Prometheus fournit aussi d’autres implémentations SD comme Kubernetes ou Amazon Elastic Compute Cloud comme le montre les deux schémas ci-dessus.

Le second composant de cet écosystème est l’exportateur Prometheus. Les exportateurs permettent de collecter des métriques à partir d’un système tiers spécifique et les rendre par la suite disponibles afin que les serveurs Prometheus puissent les récupérer. Les bibliothèques clientes, utilisées par des applications, activent un point de terminaison HTTP où les métriques internes sont exposées et collectées ensuite par les serveurs Prometheus.

Le troisième composant est les bibliothèques clients. En effet, les applications ne peuvent fournir des métriques qu’après l’ajout de l’instrumentation à leurs codes en utilisant directement les bibliothèques client Prometheus.

Le quatrième composant est Alertmanager. L’objectif principal de cet élément est de gérer les alertes envoyées par le serveur Prometheus en envoyant des notifications via plusieurs moyens comme le montre les 2 schémas ci-dessus.

Le cinquième composant est le pushgateway. Ce composant permet de récupérer les métriques des services externes à durée courte.

Enfin, nous avons le composant Web UI. Il s’agit d’une application web ayant pour but de visualiser sous forme de graphique ou de tableau, en temps réel, le résultat d’une requête promQL. Afin d’afficher les graphiques des métriques de ma vm, j’ai donc utilisé Grafana.

InfluxDB

InfluxDB est un système de gestion de base de données qui permet d’enregistrer un volume important de séries temporelles. Comme Prometheus, elle est développée en langage de programmation Go. Il est à l’écoute, par défaut, sur le port 8086 et posséde son propre langage de requêtes InfluxQL très proche du SQL, et Flux. Les bases de données InfluxDB stockent des mesures. Une base de données InfluxDB est composée d’une colonne représentant le temps en Epoch time, d’une colonne ou plusieurs colonnes représentant les tags key(s), et d’une ou plusieurs field key(s) représentant les field key(s). Les séries représentent l’ensemble des combinaisons, que nous pouvons obtenir, entre la mesure et les tags keys. InfluxDB stocke les données dans des shard groups. Ces groupes stockent les données par shard duration c’est-à-dire par intervalle de temps bien défini. C’est ce que nous appelleons la rétention de données.

Les outils de visualisation

Les deux outils de visualisation utilisés sont Grafana et Chronograf. Voici une présentation de ces 2 outils. Nous avons constaté qu’ils sont très similaires en termes de fonctionnalités.

Grafana

Grafana est une API open-source développée en langage de programmation Go et Node.js. Elle permet d’afficher graphiquement des données chronologiques collectées par d’autres applications ce qui permettra par la suite la facilité d’analyser ces métriques. L’avantage de cet outil est qu’il offre une connexion avec différentes sources de données possibles comme Prometheus, Influx DB, et MySQL. Par exemple, pour visualiser les métriques de la consommation énergétique d’une vm, j’ai utilisé la combinaison Prometheus-Grafana. Grafana constitue donc la partie front-end permettant l’affichage des données et Prometheus la partie back-end permettant de stocker les données de séries chronologiques.

De plus, l’outil Grafana possède une solution d’alerte intégrée. Ceci permet de prévenir l’utilisateur de différents problèmes, qui peuvent arriver, en envoyant des notifications par e-mail par exemple. Il peut utiliser également les régles d’alerte définies par Prometheus, Loki et Alertmanager.

Chronograf

Chronograf est une application web open source. Cette application est développée par InfluxDB. Cet outil donc nous permet de créer des tableaux de bord pré-créés ou personnalisés afin de visualiser graphiquement en temps réel les données collectées dans des bases de données InfluxDB à travers les requêtes InfluxQL ou Flux. Avec l’utilisation de Kapacitor, Chronograf offre la possibilité de création des alertes. Et l’exécution de tâches ETL. De plus, Chronograf nous offre la possibilité d’activation et de désactivation des règles d’alerte, d’affichage des alertes actives, ainsi que l’envoie des alertes à des terminaux comme l’email.

Ces 2 outils se ressemblent beaucoup et sont très riches en termes de fonctionnalités qu’ils proposent. Ainsi, pour travailler avec Prometheus, il est plus judicieux d’utiliser Grafana, et pour travailler avec InfluxDB, il est plus cohérent d’utiliser Chronograf.

Conclusion à rajouter

Réalisation

Dans cette partie, je vais présenter les missions que j’ai effectuées jusqu’à maintenant. Je vais d’abord commencer par expliquer et décrire la première mission qui est la mise en place d’un tableau de bord pour la visualisation des métriques de consommation énergétique depuis une vm, ensuite je vais enchainer par l’explication de la mise en place du tableau de bord contenant les métriques de consommation énergétique de 2 serveurs distants.

Tableau de bord des métriques d’une vm

Comme les APIs de mesure de la consommation énergétique existantes actuellement ne permettent pas malheureusement d’accéder directement aux informations de consommation énergétique d’une vm et que je n’ai pas un accès malheureusement aux hyperviseurs du laboratoire, j’ai installé un hyperviseur KVM et configuré une vm parce que l’outil, que l’API Scaphandre, ne fonctionne que seulement avec ce type de système pour la virtualisation de serveur. J’ai commencé par installer tous les packages que j’aurai besoin en exécutant les 2 commandes shell suivantes **sudo apt -y install qemu-kvm libvirt-daemon bridge-utils virtinst libvirt-daemon-system** puis **sudo apt -y install virt-top libguestfs-tools libosinfo-bin qemu-system virt-manager .** Après avoir téléchargé le fichier .iso de linux ubuntu, j’ai donc créé une machine virtuelle en utilisant directement le gestionnaire des machines virtuelles que j’ai installé. Une des difficultés rencontrées lors de cette partie est de mon rendre compte du non fonctionnement de l’API Scaphandre avec plus d’un seul processeur virtuel. J’ai donc configuré une machine virtuelle avec un seul vCPU.

Il faut savoir que le problème majeur dans la mesure de la consommation d'énergie est de le faire à l'intérieur d'une vm parce que d’une manière générale la vm n'a pas accès aux mesures de puissance. L’avantage de Scaphandre est qu’il résout cette problématique en permettant une communication entre une instance Scaphandre sur l'hyperviseur et une autre instance s'exécutant sur la vm. Ainsi, l'agent Scaphandre sur l'hyperviseur calculera les métriques pour cette vm et celui sur la VM accédera par la suite à ces métriques.

Après avoir cloné le répertoire « Scaphandre » sur la machine bare-metal et charger le module intel\_rapl\_common, j’ai créé le fichier binaire, qui permet de lancer l’API Scaphandre, avec **cargo build --release**. Ainsi, le fichier binaire se trouve dans le répertoire scaphandre/target/release/. Concernant la partie hyperviseur, j’ai enfin lancé Scaphandre avec l’exportateur QEMU en exécutant la commande suivante: **sudo ./scaphandre qemu**. Maintenant je vais pouvoir donner accées à la mv créée à ses métriques mesurées depuis l’hyperviseur. Pour cela, j’ai créé un point de montage tmpfs via l’exécution de la commande shell suivante:

**sudo mount -t tmpfs tmpfs\_ubuntu22.04 /var/lib/libvirt/scaphandre/ ubuntu22.04 -o size=5m**.

En ce qui concerne la vm, j’ai ajouté un nouveau matériel virtuel en mettant les paramétres suivants. Pour le « Pilote », j’ai sélectionné « virtio-9p ». Pour le chemin de source, j’ai mis « /var/lib/libvirt/scaphandre/ubuntu22.04». J’ai mis « scaphandre » pour le chemin cible, et coché la case « Exporter le système de fichiers en lecture seule ». Comme pour l’hyperviseur, j’ai cloné le répertoire « Scaphandre », et j’ai créé le fichier binaire qui permet de lancer l’API Scaphandre. Dans le répertoire « var », j’ai créé un répertoire nommé « scaphandre ». Ensuite, j’ai monté le système de fichiers sur cette vm à l’aide de la commande suivante : **sudo mount -t 9p -o trans=virtio scaphandre /var/scaphandre**. J’ai lancé l’API Scaphandre sur cette vm en exportant les métriques avec l’exportateur Prometheus avec cette commande shell : **sudo ./scaphandre –vm prometheus.**

Les métriques sont ainsi collectables, d’une manière donc non graphique, via le navigateur en tapant directement l’adresse ip de ma vm suivi du numéro de port 8080.

Afin de pourvoir afficher ces métriques de consommation énergétique, j’ai trouvé une solution assez efficace intégrée à l’écosystème Prometheus. Pour cela, il me faut 3 éléments importants : l’API Scaphandre fonctionnant avec l’exportateur Prometheus, afin de mesurer les métriques de la consommation énergétique en temps réel de la vm, Prometheus afin de les collecter, et Grafana, afin de les visualiser graphiquement en s’appuyant sur une connexion source de données vers le serveur Prometheus.

Pour la visualisation graphique, j’ai commencé donc par installer Prometheus à l’aide de la commande wget :

**wget** [**https://github.com/prometheus/prometheus/releases/download/v2.35.0/prometheus-2.35.0.linux-amd64.tar.gz**](https://github.com/prometheus/prometheus/releases/download/v2.35.0/prometheus-2.35.0.linux-amd64.tar.gz). Puis, j’ai créé un fichier de configuration afin de rendre ma vm reconnue par Prometheus, ce qui permettra par la suite de stocker les métriques de la consommation énergétique dans le serveur Prometheus. Voici le contenu de ce fichier :



J’ai lancé Prometheus avec l'indicateur --config.file afin de prendre en compte le fichier créé ci-dessus: **./prometheus --config.file=exporter-config.yml,** et j’ai installé Grafana. Pour cela, j’ai d’abord mis à jour les informations du package avec les 3 commandes suivantes : **sudo apt-get install -y apt-transport-https**, **sudo apt-get install -y software-properties-common wget** et **wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add –**

Ensuite, j’ai ajouté un repository de Grafana à l’aide de la commande suivante :

**echo "deb https://packages.grafana.com/enterprise/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list.** Puis, j’ai installé Grafana avec : **sudo apt-get install grafana-enterprise.**

Je suis enfin capable de commencer à construire le tableau de bord avec Grafana et visualiser les graphiques des métriques de consommation énergétique de la vm. J’ai donc lancé Grafana en exécutant les commandes suivantes **sudo systemctl daemon-reload** et **sudo systemctl start grafana-server.** Cet outil de visualisation est maintenant accessible via le port 3000**.** Après avoir définir la source de données Prometheus, et saisi l’adresse IP du serveur Promehteus qui fonctionne sur le port 9090,j’ai bien obtenu les métriques de la consommation énergétique de la vm fournies par l’API Scaphandre. Grace au langage de requêtes PromQL, j’ai créé le tableau de bord suivant :

****

Par exemple, pour obtenir le premier graphique de la figure ci-dessous, représentant la mesure de puissance sur l'ensemble de l'hôte watts « Puissance de l’hôte (en Watt) », j’ai exécuté la requête PromQL suivante en la divisant par 1000000:



Les graphiques à droite de la figure précédente représentent la consommation énergétique en watt de chaque processus entrain de s’exécuter sur la vm.

Tableau de bord des métriques de 2 serveurs distants

**Inttro à rajouter**

Réalisation d’un tableau excel pour les conversion énergétiques métaphoriquement

Les métriques de consommation énergétique d’une machine virtuelle (les étapes à effectuer afin d’avoir un dashboard avec des métriques d’une vm en temps réel)

Les métriques de consommation énergétique d’un serveur (bare-metal) (les étapes à effectuer afin d’avoir un dashboard avec des métriques d’un bare metal à distance)

Présentation du résulat final (décrire les données présentes dans les dashboard + si possible des relations entre eux avec justification)

Pour expliquer les métriques : https://hubblo-org.github.io/scaphandre-documentation/references/exporter-prometheus.html

**Conclusion à rajouter**

**Pour la période suivante, amélioration des dashboards en rajoutant plus de métriques et en convertissant les unités de mesures watts en unité plus parlante (du tableur réalisé).**

**Mesurer la consommation énergétique des applications avec powerJoular**

**Monitorer les applications avec la libraire de python prometheus-client**

Inconvénient & améliorations à apporter à Scaphandre:

1-D'autres améliorations seront apportées à scaphandre pour mesurer pleinement la consommation lorsque des GPU sont impliqués (ou beaucoup de disques durs sur le même hôte...).

2-Lorsque vous n'avez pas accès à l'hyperviseur/machine bare-metal (c'est-à-dire lorsque vous exécutez sur des instances de cloud public et que votre fournisseur n'exécute pas scaphandre), vous avez toujours la possibilité d'estimer la consommation d'énergie, en fonction à la fois des ressources (cpu/gpu/ram/io...) consommés par la machine virtuelle à un instant donné, et les caractéristiques du matériel sous-jacent. C'est ainsi que nous concevons le futur capteur basé sur l'estimation, pour correspondre à ce cas d'utilisation.

Bibliographie

## <https://liuppa.univ-pau.fr/fr/index.html>

## https://hubblo-org.github.io/scaphandre-documentation/

## https://www.devopsschool.com/blog/what-is-prometheus-and-how-it-works/