

Identification des checks les plus consommateurs à l'aide de la commande `shinken-scheduler-export-data`

Sommaire

- Économiser des CPUs en identifiant les checks les plus consommateurs
- Création du tableau récapitulatif sur la consommation totale des temps CPU des royaumes respectifs
 - Consolidation des données: utilisation d'un tableau croisé dynamique
 - Création du tableau croisé dynamique
 - Sélection des commandes en tant que lignes de notre tableau
 - Obtenir la consommation CPU totale par check
 - Passer du nombre de lignes avec "cpu_time" à une vraie somme des temps CPU
 - Rendre le titre de la colonne explicite
 - Obtenir la répartition de la consommation CPU par check
 - Passer la seconde colonne en somme de temps, comme la première
 - Passer du nombre total de temps CPU consommé à la répartition de la consommation CPU par check
 - Rendre le titre de la colonne explicite
 - Obtenir la répartition de la consommation CPU moyenne pour une exécution (en s)
 - Passer du nombre total de temps CPU consommé à la consommation CPU moyenne pour une exécution de check
 - Rendre le titre de la colonne explicite
- Analyse des résultats
 - Analyse du cas présent: quel check doit être optimisé en priorité pour économiser du CPU?
 - Quel check est le plus consommateur?
 - Exercice d'exemple: quel check optimiser afin de limiter la consommation CPU à un seul CPU?

Économiser des CPUs en identifiant les checks les plus consommateurs

Lorsque l'on souhaite réduire la consommation CPU de sa supervision, optimiser ses commandes de vérification est une étape primordiale. En effet, les commandes de vérification sont la plupart du temps la première source de consommation CPU de la supervision. L'objectif de cette page est de permettre l'identification des commandes dont l'optimisation aura le plus d'influence sur la consommation CPU.

Pour cela, nous allons extraire des informations issues de la commande `shinken-scheduler-export-data` afin d'avoir:

- **le temps CPUs utilisés par chaque commande sur l'ensemble des royaumes**

Dans notre exemple. Nous avons un cas simplifié avec 3 checks:

- **check-host-alive:**
 - check par défaut pour la vérification des hôtes
- **dump-check-example-1:**
 - check d'exemple
- **dump-check-example-2:**
 - check d'exemple

Sur une installation réelle, il y aura plus de commandes, mais l'analyse restera la même vu que le principe sera de trouver les plus consommateurs.

On a besoin d'un dump de données avec en option une prévision de la charge sur une période, disons par exemple 1 heure. Il suffit alors de lancer la commande comme ceci:

```
shinken-scheduler-export-data --export-type=identify-most-consuming-checks --simulate-scheduling-for-X-seconds=3600
```

L'importation du fichier .csv généré est décrit dans la page suivante : [shinken-scheduler-export-data - export des données du Scheduler](#)



Avec ce lancement les noms (*hôte, check, commandes et royaumes*) seront présents dans l'export. Il est tout à fait possible de faire l'analyse sur un export en `--anonymous`, des hash des noms des commandes seront utilisés au lieu des noms finaux.

Le client pourra lancer un export avec les noms en parallèles afin de faire le lien avec les commandes une fois l'analyse effectuée.

Création du tableau récapitulatif sur la consommation totale des temps CPU des royaumes respectifs

Consolidation des données: utilisation d'un tableau croisé dynamique

On va devoir procéder à une consolidation des nombreuses données de checks obtenues afin d'obtenir un résultat exploitable.

- Pour cela on va utiliser la fonctionnalité d'Excel : **le tableau croisé dynamique**.
- Un tableau croisé dynamique dans un outil de analyse de données qui vous permet de créer une vue synthétique et facile à lire d'une grande quantité de données (*ici nos exécutions de checks*).

On y choisit les données à inclure, comment les organiser et comment les synthétiser, filtrer, classer et totaliser les données en fonction de nos besoins

Afin de pouvoir faire une analyse complète, notre objectif sera ici d'obtenir dans notre tableau:

- **la consommation CPU totale par check,**
- **la répartition de la consommation CPU par check par rapport au autres,**
- **et enfin la consommation CPU moyenne pour une exécution pour chaque check**

Création du tableau croisé dynamique

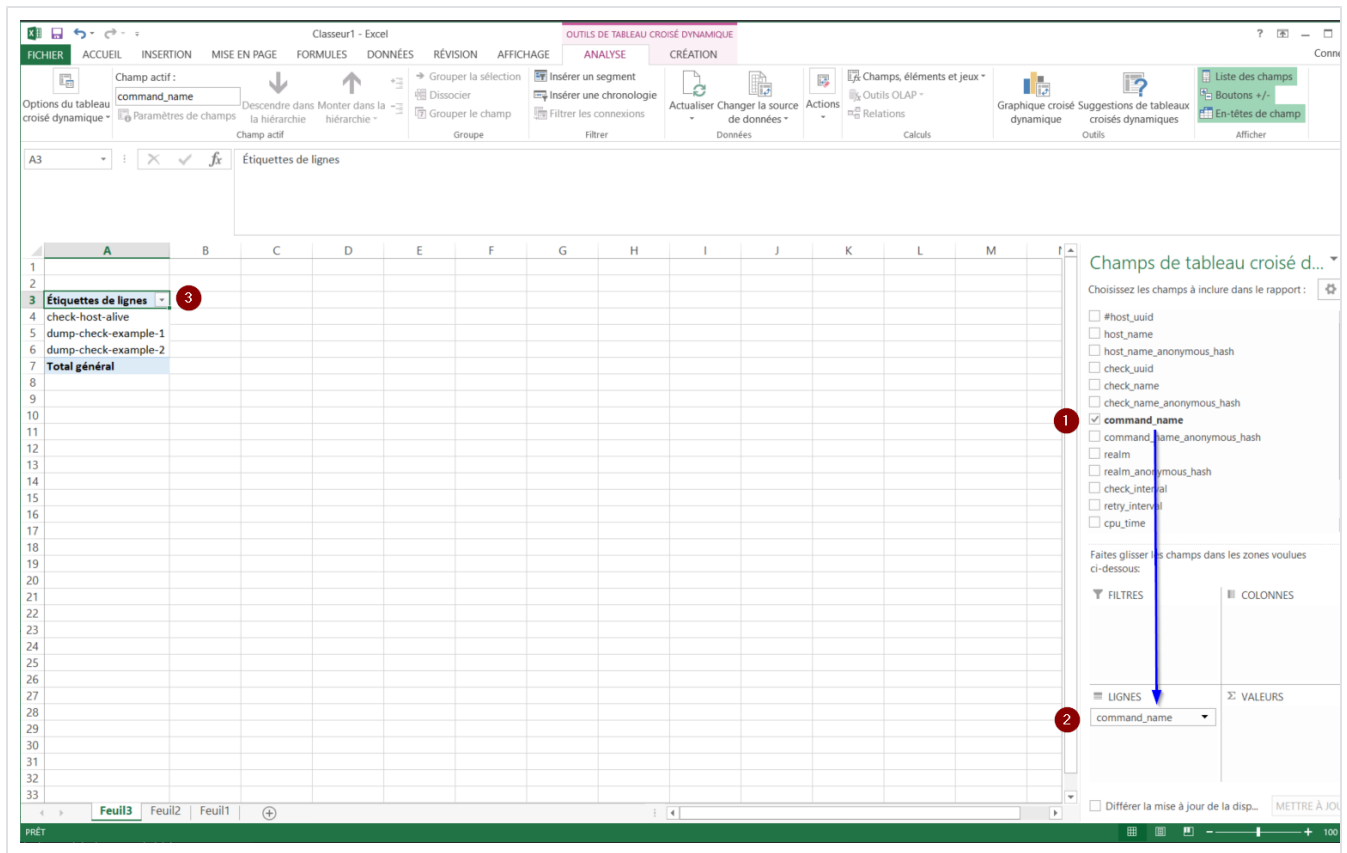
La création du tableau récapitulatif passe par la création d'un **Tableau croisé dynamique**. Depuis votre Feuille d'importation des données, il faut cliquer sur **Insertion Tableau croisé dynamique**, et valider :

#host_uid	host_name	host_name_anonymous_hash	check_uid	check_name	check_name_anonymous_hash	command_name	command_name_anonymous_hash	rec
df6e5e40072b41b8950ed69d1329cc48	srv-15	anonymous-hash-0f628613ec	0f31a3424df44ae41053d5c72dbc7eda	dump-check-example-2-3	anonymous-hash-ab3dec090e	dump-check-example-2	anonymous-hash-aa36432587	All
1b7ce868df5547038603eecd6161cae2	srv-7	anonymous-hash-eb645d6b77	24e86f31b7f133bda96ac03a67ff0f14	dump-check-example-2-13	anonymous-hash-a6eb5d73e9	dump-check-example-2	anonymous-hash-aa36432587	All
8bc339daa0a311eda85d080027940ca8	srv-2	anonymous-hash-6fb60b8226	bb64373e94c329a136099798292c7db56	dump-check-example-2-14	anonymous-hash-19848ce407	dump-check-example-2	anonymous-hash-aa36432587	cu
8bcd91c8a0a311ed99d080027940ca8	srv-8	anonymous-hash-3			anonymous-hash-2e9e104669	dump-check-example-2	anonymous-hash-aa36432587	cu
8bd19548a0a311ed8e0f080027940ca8	srv-10	anonymous-hash-ab			anonymous-hash-19848ce407	dump-check-example-2	anonymous-hash-aa36432587	cu
8ae43305c6b6490d8129c92d58ca8082	srv-8	anonymous-hash-9			anonymous-hash-688a5ae71d	dump-check-example-2	anonymous-hash-aa36432587	All
1884e097f2a74ee7b82f1a8479852ea1	srv-5	anonymous-hash-4			anonymous-hash-ff61e06169	dump-check-example-2	anonymous-hash-aa36432587	All
ff4f1c89d4cea4c938ca2f86acfe7a2c0	srv-17	anonymous-hash-at			anonymous-hash-ab3dec090e	dump-check-example-2	anonymous-hash-aa36432587	All
8bd4e2aa0a311eda148080027940ca8	srv-16	anonymous-hash-bl			anonymous-hash-19848ce407	dump-check-example-2	anonymous-hash-aa36432587	cu
8bd19548a0a311ed8e0f080027940ca8	srv-10	anonymous-hash-at			anonymous-hash-2e9e104669	dump-check-example-2	anonymous-hash-aa36432587	cu
8be631e2a0a311ed99d080027940ca8	srv-20	anonymous-hash-4			anonymous-hash-5f8ecad9f7	dump-check-example-2	anonymous-hash-aa36432587	cu
340f7d2fcaab43ccb394e0fd6d427469	srv-13	anonymous-hash-2f			check-host-alive	anonymous-hash-2d6a3ff550	anonymous-hash-2d6a3ff550	All
df6e5e40072b41b8950ed69d1329cc48	srv-15	anonymous-hash-0			anonymous-hash-688a5ae71d	dump-check-example-2	anonymous-hash-aa36432587	All
d26e317996a2441e955f04f759f93c18	srv-9	anonymous-hash-8			anonymous-hash-b524c09dc	dump-check-example-2	anonymous-hash-aa36432587	All
1b7ce868df5547038603eecd6161cae2	srv-7	anonymous-hash-ei			anonymous-hash-ab3dec090e	dump-check-example-2	anonymous-hash-aa36432587	All
8bc7135ca0a311eda148080027940ca8	srv-4	anonymous-hash-9f			anonymous-hash-2e9e104669	dump-check-example-2	anonymous-hash-aa36432587	All
8bd9ad28a0a311eda85d080027940ca8	srv-14	anonymous-hash-7f			anonymous-hash-ab3dec090e	dump-check-example-2	anonymous-hash-aa36432587	cu
8bd524b4a0a311ed6b35080027940ca8	srv-12	anonymous-hash-0			anonymous-hash-2e9e104669	dump-check-example-2	anonymous-hash-aa36432587	cu
2d833e2aeebc488385c10043026bedda	srv-19	anonymous-hash-5			anonymous-hash-19848ce407	dump-check-example-2	anonymous-hash-aa36432587	All
340f7d2fcaab43ccb394e0fd6d427469	srv-13	anonymous-hash-2f			anonymous-hash-8ac2db040f	dump-check-example-2	anonymous-hash-aa36432587	All
340f7d2fcaab43ccb394e0fd6d427469	srv-13	anonymous-hash-2f	512071c5e2b49849ed7d72ae5f657578	dump-check-example-2-7	anonymous-hash-176d0d8c03	dump-check-example-2	anonymous-hash-aa36432587	All
8bc7135ca0a311eda148080027940ca8	srv-4	anonymous-hash-954836fc25	424c8690e6ff9da09f892f383b49de1	dump-check-example-2-15	anonymous-hash-80e21d30ac	dump-check-example-2	anonymous-hash-aa36432587	cu
1884e097f2a74ee7b82f1a8479852ea1	srv-5	anonymous-hash-44610da532			check-host-alive	anonymous-hash-2d6a3ff550	anonymous-hash-2d6a3ff550	All
8bd9ad28a0a311eda85d080027940ca8	srv-10	anonymous-hash-72169f19f2	b4e36da700f37552d37eacd9fbee332e	dump-check-example-2-6	anonymous-hash-8ac2db040f	dump-check-example-2	anonymous-hash-aa36432587	cu
8bd19548a0a311ed8e0f080027940ca8	srv-14	anonymous-hash-a89f7681d1			check-host-alive	anonymous-hash-2d6a3ff550	anonymous-hash-2d6a3ff550	cu
8bc339daa0a311eda85d080027940ca8	srv-2	anonymous-hash-6fb60b8226	0f31a3424df44ae41053d5c72dbc7eda	dump-check-example-2-3	anonymous-hash-ab3dec090e	dump-check-example-2	anonymous-hash-aa36432587	cu
340f7d2fcaab43ccb394e0fd6d427469	srv-13	anonymous-hash-20686577d2	1769d9759e8be6d3680ff5739f431752	dump-check-example-2-12	anonymous-hash-bb4ef5977d	dump-check-example-2	anonymous-hash-aa36432587	All
340f7d2fcaab43ccb394e0fd6d427469	srv-13	anonymous-hash-20686577d2	424c8690e6ff9da09f892f383b49de1	dump-check-example-2-15	anonymous-hash-80e21d30ac	dump-check-example-2	anonymous-hash-aa36432587	All
8bc20806a0a311ed9182080027940ca8	srv-18	anonymous-hash-c7f9a1e80	1769d9759e8be6d3680ff5739f431752	dump-check-example-2-12	anonymous-hash-bb4ef5977d	dump-check-example-2	anonymous-hash-aa36432587	cu
8bcd91c8a0a311ed99d080027940ca8	srv-8	anonymous-hash-35a25c2308	6582a351a934f732b7096e3c8ddaf10a	dump-check-example-2-8	anonymous-hash-3395da4466	dump-check-example-2	anonymous-hash-aa36432587	cu
8bd4e2aa0a311eda148080027940ca8	srv-16	anonymous-hash-bb7dc70c8e	512071c5e2b49849ed7d72ae5f657578	dump-check-example-2-7	anonymous-hash-176d0d8c03	dump-check-example-2	anonymous-hash-aa36432587	cu

Sélection des commandes en tant que lignes de notre tableau

Arrivé sur la nouvelle feuille, Excel demande quelles lignes sélectionner, dans le bloc de droite nommé "Champs de tableau croisé dynamique".

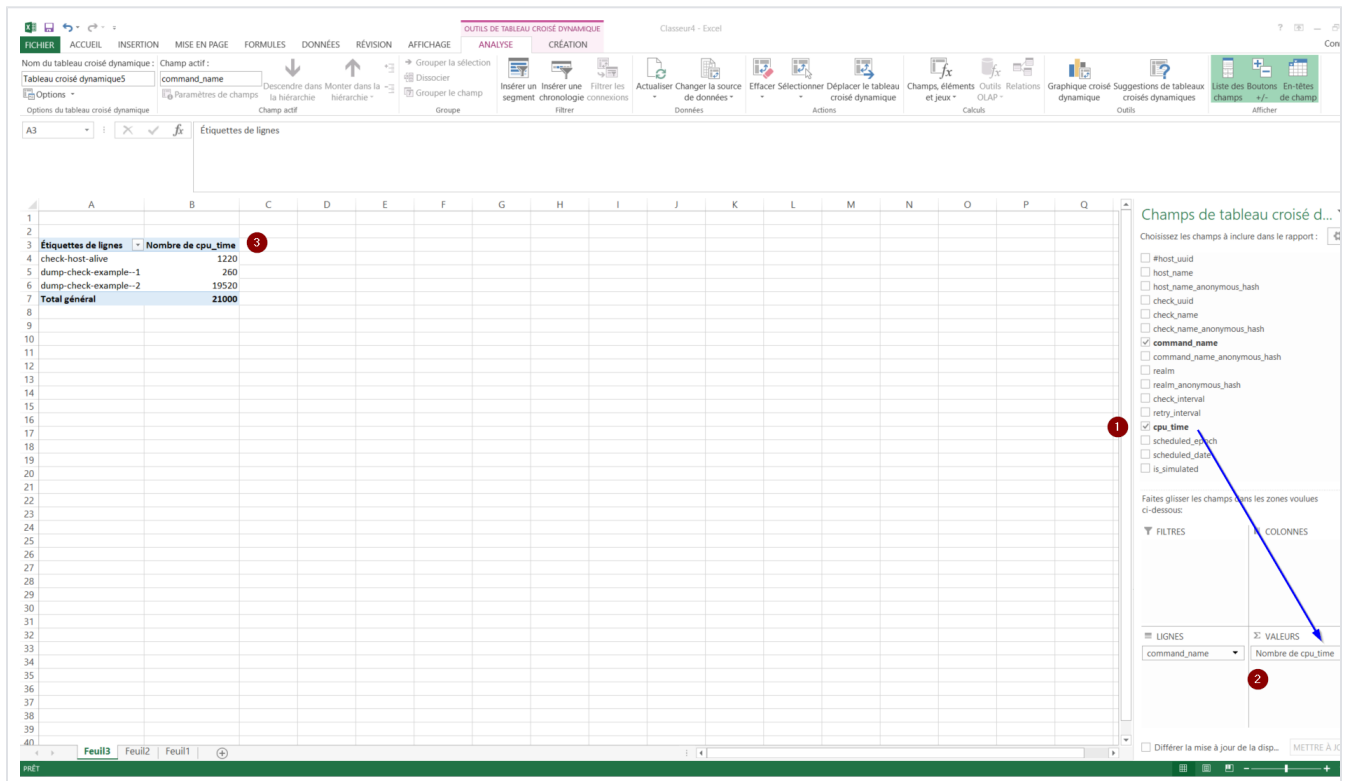
Dans notre cas, il faut faire glisser le champ **command_name** (ou bien **command_name_anonymous_hash** si on a une version anonyme de l'export) vers le bloc "Lignes" :



Ceci permet de définir nos lignes dans la colonne A avec l'ensemble de nos commandes.

Obtenir la consommation CPU totale par check

A chaque commande, il faut lui assigner une (*ou plusieurs*) "Valeurs". Pour cela, on fait glisser le champ "cpu_time" vers le bloc "Valeurs" afin d'avoir pour chaque royaume son champs `cpu_time` associé :



A noter que, par défaut, Excel prend le nombre d'occurrences du champ `cpu_time` comme "Valeur", ce qui n'est pas ce qui est souhaité ("Nombre de `cpu_time`").

Passer du nombre de lignes avec "cpu_time" à une vraie somme des temps CPU

Une modification des "Paramètres des champs de valeurs" est nécessaire sur "Nombre de `cpu_time`" qu'il faut changer en Somme pour donner au final "Somme de `cpu_time`" :

The screenshot shows the Microsoft Excel interface with a PivotTable. The PivotTable has 'command_name' as the row field and 'Nombre de cpu_time' as the column field. The data is as follows:

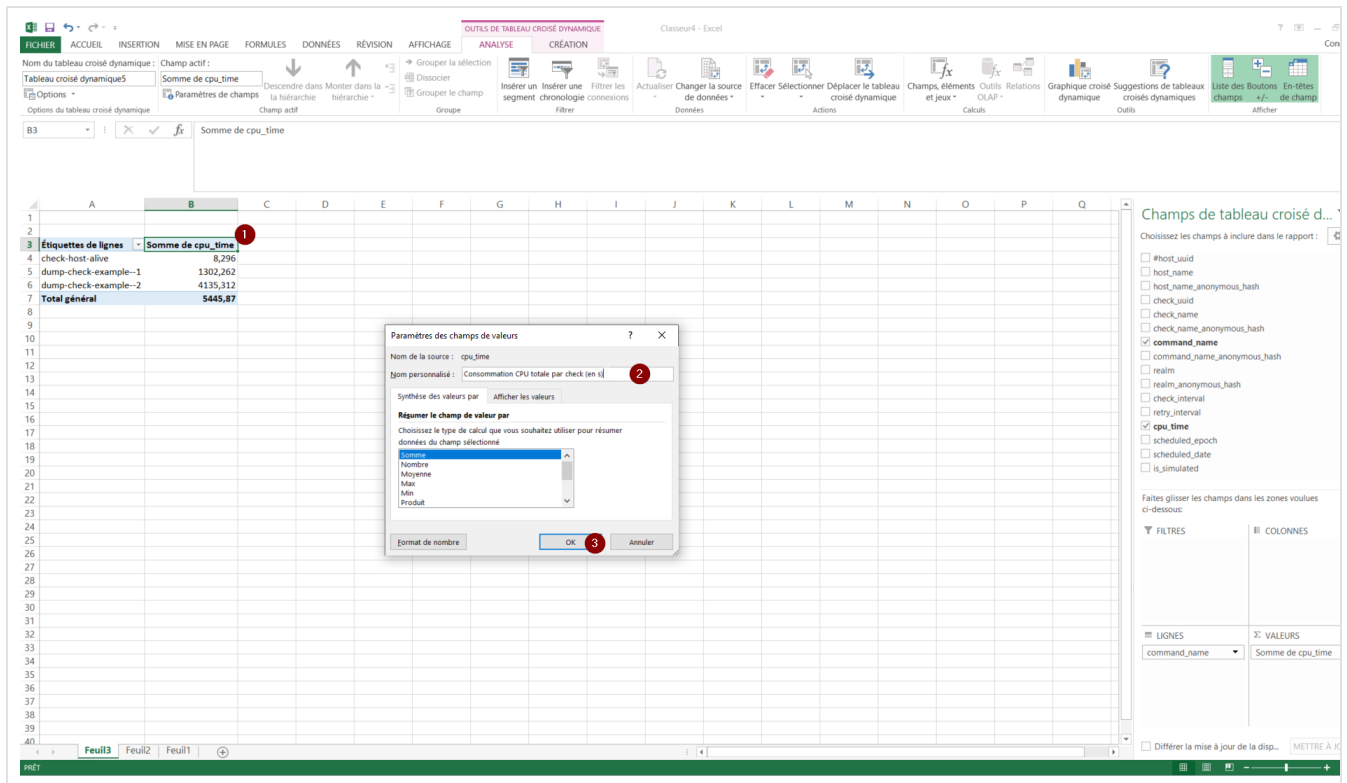
command_name	Nombre de cpu_time
check-host-alive	1220
dump-check-example-1	250
dump-check-example-2	19520
Total général	21000

A dialog box titled "Paramètres des champs de valeurs" is open, showing the configuration for the field "Nombre de cpu_time". The "Nom de la source" is "cpu_time" and the "Nom personnalisé" is "Somme de cpu_time". Under "Résumer le champ de valeur par", the "Somme" option is selected. The dialog also has an "Ecart de nombre" field and "OK" and "Annuler" buttons.

On obtient alors la somme en temps (*secondes*) consommée par chaque commande.

Rendre le titre de la colonne explicite

Le titre de la colonne n'étant pas des plus lisible pour notre analyse, on le change en double cliquant dessus, et le renomme alors "Consommation CPU totale par check (en s)" :



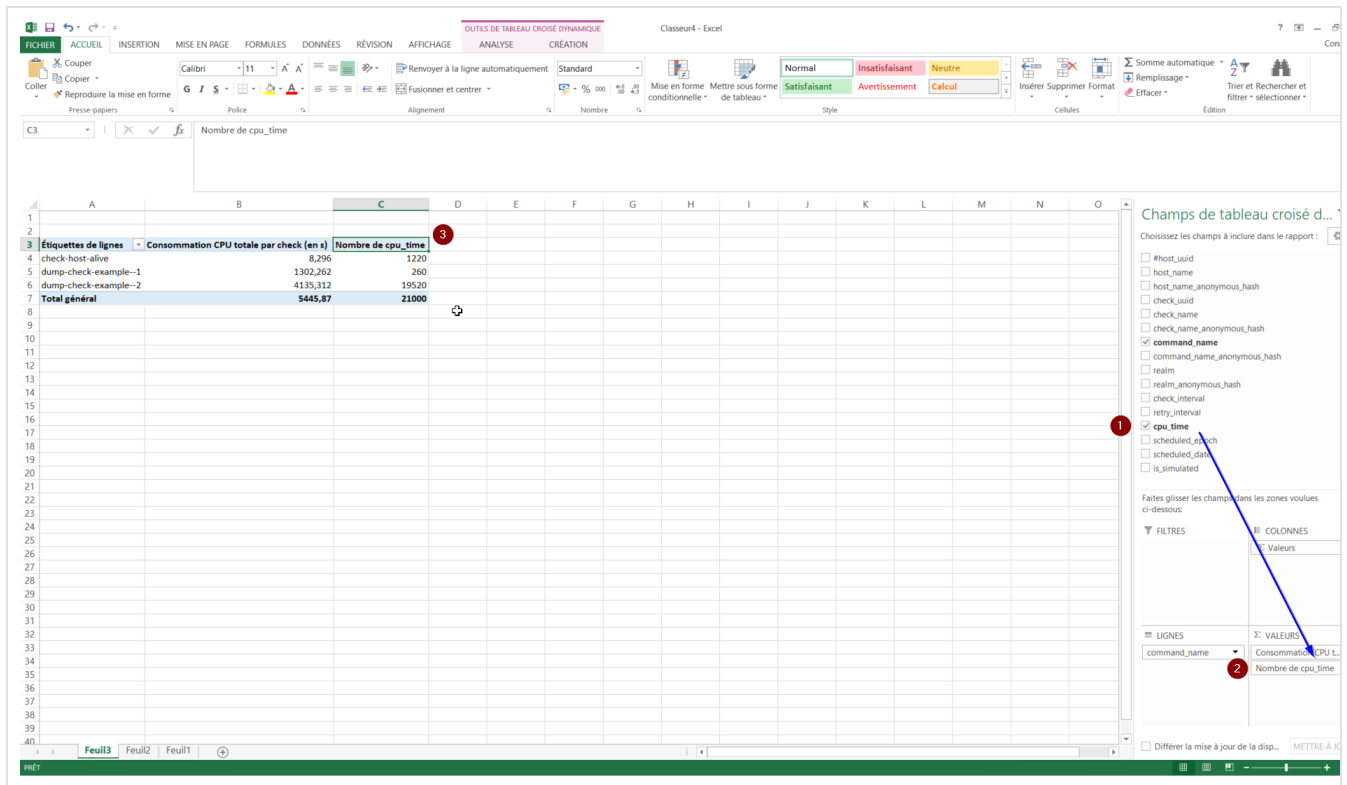
Notre première valeur est obtenue: **la consommation CPU totale par check.**

Obtenir la répartition de la consommation CPU par check

S'il est possible de trier par rapport au temps absolu consommé par un check et donc avoir les plus consommateurs, il peut être également très intéressant d'avoir leur pourcentage par rapport à la consommation totale.

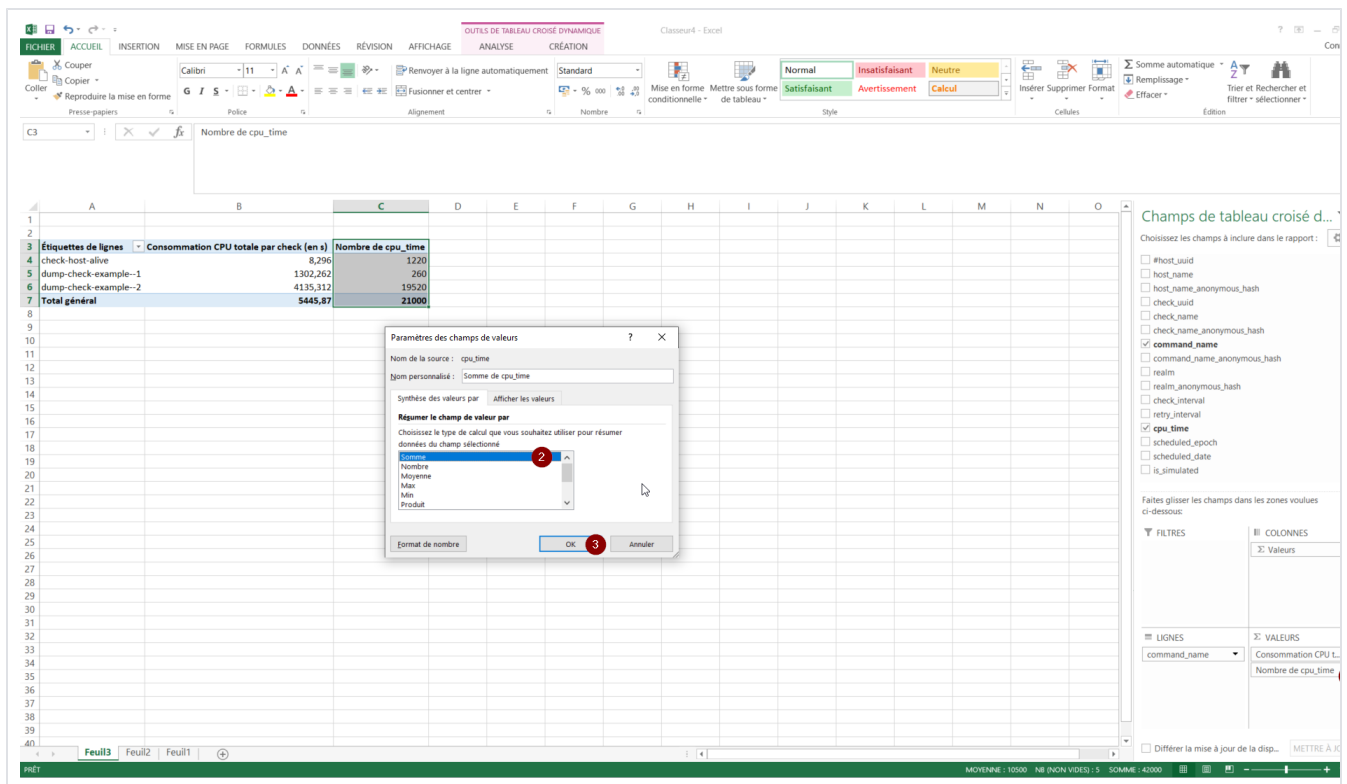
Ceci en facilitera grandement la lecture, et il sera ainsi possible de savoir de quel ordre de grandeur une optimisation sera bénéfique pour la consommation.

Il faut rajouter une seconde fois le champ "**cpu_time**" dans les valeurs, ce qui rajoute une nouvelle colonne :



Passer la seconde colonne en somme de temps, comme la première

Ensuite il faut passer, comme pour la colonne précédente, d'un calcul en **Nombre** à un calcul en "Somme de cpu_time" :



Passer du nombre total de temps CPU consommé à la répartition de la consommation CPU par check

Le changement dans cette étape consiste à changer l'affichage dans le menu "Afficher les valeurs" en choisissant "% du total général" pour obtenir la Répartition de la consommation CPU par check :

The screenshot shows an Excel spreadsheet with a PivotTable. The PivotTable is located in columns C and D, with rows 3 to 7. The column header is 'Somme de cpu_time' and the row header is 'Total général'. The value for 'Total général' is 5445,87. A context menu is open over the cell containing 5445,87, and the option '% du total général' is selected. The PivotTable Fields task pane on the right shows 'command_name' in the Rows area and 'Somme de cpu_time' in the Values area.

Étiquettes de lignes	Consommation CPU totale par check (en s)	Somme de cpu_time
check-host-alive	8,236	8,236
dump-check-example-1	1302,262	1302,262
dump-check-example-2	4135,312	4135,312
Total général	5445,87	5445,87

Rendre le titre de la colonne explicite

Comme précédemment, on change le titre par défaut en double cliquant dessus. On le renomme alors "Répartition de la consommation CPU par check" :

The screenshot shows an Excel PivotTable with the following data:

Étiquettes de lignes	Consommation CPU totale par check (en s)	Somme de cpu_time
check-host-alive	8,236	0,15%
dump-check-example--1	1302,262	23,91%
dump-check-example--2	4135,312	75,93%
Total général	5445,87	100,00%

The 'Paramètres des champs de valeurs' dialog box is open, showing the following settings:

- Nom de la source: cpu_time
- Nom personnalisé: Répartition de la consommation CPU par check
- Synthèse des valeurs par: Afficher les valeurs
- Résumer le champ de valeur par: Moyenne
- Format de nombre: Nombre

Ces deux premières colonnes nous permettent d'identifier rapidement les checks les plus consommateurs.

On a alors obtenu la seconde donnée intéressante: **la répartition de la consommation CPU par check par rapport au autres.**

La suivante, et dernière, va nous permettre d'affiner l'analyse en calculant le temps moyen de CPU par exécution de check.

Obtenir la répartition de la consommation CPU moyenne pour une exécution (en s)

La dernière statistique qui va nous intéresser pour prioriser nos actions d'optimisation va être d'obtenir la consommation CPU moyenne pour une exécution de check.

Pour cela, il faut rajouter une dernière fois le champ "cpu_time" dans les valeurs, exactement comme pour les deux précédentes colonnes :

Champs de tableau croisé d...

Choisissez les champs à inclure dans le rapport :

- #host_uid
- host_name
- host_name_anonymous_hash
- check_uid
- check_name
- check_name_anonymous_hash
- command_name
- command_name_anonymous_hash
- realm
- realm_anonymous_hash
- check_interval
- retry_interval
- cpu_time
- scheduled_epoch
- scheduled_date
- is_simulated

Faites glisser les champs dans les zones voulues ci-dessous :

FILTRES

COLONNES

VALEURS

LIGNES

command_name

Consommation CPU L...

Répartition de la con...

Nombre de cpu_time

Passer du nombre total de temps CPU consommé à la consommation CPU moyenne pour une exécution de check

Pour cette colonne, il faut changer le calcul de **Nombre** à un calcul en "Moyenne de cpu_time" :

Paramètres des champs de valeurs

Nom de la source : cpu_time

Nom personnalisé : Moyenne de cpu_time

Synthèse des valeurs par Afficher les valeurs

Résumer le champ de valeur par

Choisissez le type de calcul que vous souhaitez utiliser pour résumer données du champ sélectionné

- Somme
- Nombre
- Moyenne**
- Max
- Min
- Produit

Format de nombre

OK Annuler

Rendre le titre de la colonne explicite

Comme pour les deux colonnes précédentes, on change le titre de la colonne en change en double cliquant dessus. On le renomme alors "**Consommation CPU moyenne pour une exécution (en s)**" :

The screenshot shows an Excel spreadsheet with a PivotTable. The PivotTable has three columns: 'Consommation CPU totale par check (en s)', 'Répartition de la consommation CPU par check', and 'Moyenne de cpu_time'. A dialog box titled 'Paramètres des champs de valeurs' is open, showing the 'Moyenne' (Average) option selected for the 'Moyenne de cpu_time' field. The dialog box also shows the source name 'cpu_time' and the personalized name 'Consommation CPU moyenne pour une exécution (en s)'. The 'OK' button is highlighted with a red circle.

	Consommation CPU totale par check (en s)	Répartition de la consommation CPU par check	Moyenne de cpu_time
check-host-alive	8,296	0,15%	0,0066
dump-check-example-1	1302,262	23,91%	5,0087
dump-check-example-2	4135,312	75,93%	0,21185
Total général	5445,87	100,00%	0,259327143

On obtient alors la dernière de nos trois données d'analyse: **la consommation CPU moyenne pour une exécution pour chaque check.**

Analyse des résultats

Le tableau final obtenu, on peut analyser afin d'identifier de la manière la plus efficace possible le ou les checks les plus consommateurs.

Il est ainsi possible de se concentrer sur l'optimisation des checks qui auront un véritable impact sur les performances de la plateforme.

Étiquettes de lignes	Consommation CPU totale par check (en s)	Répartition de la consommation CPU par check	Consommation CPU moyenne pour une exécution (en s)
check-host-alive	8,296	0,15%	0,0068
dump-check-example-1	1302,262	23,91%	5,0087
dump-check-example-2	4135,312	75,93%	0,21185
Total général	5445,87	100,00%	0,259327143

Analyse du cas présenté: quel check doit être optimisé en priorité pour économiser du CPU?

Quel check est le plus consommateur?

Faisons l'exercice avec notre exemple. Nous avons **3** checks (leur temps d'exécution est extrait de la colonne "Consommation CPU moyenne pour une exécution (en s)") :

- **check-host-alive:**
 - check par défaut pour la vérification des hôtes, seulement **0.006s (6ms)** de temps moyen d'exécution, très peu couteux car il lance principalement un simple ping
- **dump-check-example-1:**
 - check d'exemple qui consomme en moyenne **5s** de temps CPU par exécution, ce qui est **très élevé** pour un check
- **dump-check-example-2:**
 - second check d'exemple qui consomme **0.2s (200ms)** de temps CPU par exécution, ce qui est **légèrement élevé** pour un check

En ne regardant que ces valeurs, et en ne se basant que sur notre première impression, on serait tenté de se dire que le check le plus consommateur est **dump-check-example-1** avec ses **5s** de temps d'exécution par check. C'est en effet un temps anormal pour un check de supervision.

Mais cette première impression est trompeuse. En effet, si on regarde mieux notre tableau on s'aperçoit que ce check ne représente qu'environ **25%** du temps CPU total consommé (**23,91%**, colonne "**Répartition de la consommation CPU par check**"). Il n'est ainsi pas le plus consommateur /prioritaire.

Le check le plus consommateur est en fait **dump-check-example-2**, avec ses plus de **75%** de temps CPU total consommé (**75,93%**). Il est donc intéressant de voir en priorité les gains que l'on peut faire sur ce dernier, surtout qu'avec ses **200ms** de temps d'exécution, il n'est pas léger, et donc des optimisations doivent être possibles.

Et seulement dans un second temps voir l'optimisation de **dump-check-example-1** qui avec ses **5s** de temps d'exécution doit largement avoir de quoi être optimisé.

Exercice d'exemple: quel check optimiser afin de limiter la consommation CPU à un seul CPU?

On utilise nos données pour faire un exercice pratique et concret: on ne souhaite utiliser qu'un seul CPU pour l'ensemble de nos checks.

Dans notre cas, on avait pris un temps de simulation de **3600s=1h** pour notre extraction d'ordonnancement. La somme totale des temps d'exécutions de toutes les checks est de **5445s**, soit moins de **2h** de temps de calcul (**7200s**).

L'ensemble des checks consomment ainsi actuellement moins de 2 CPUs ($5445/3600=1.5$ CPUs).

Si l'on souhaite ne consommer qu'un seul CPU sur cette plateforme, il faut ainsi gagner $5445-3600=1845s$ de temps de traitement. Deux possibilités s'offrent alors:

- si on se limite à optimiser le check **dump-check-example-1**:
 - l'objectif de se limiter à 1CPU est **impossible**, car il ne consomme au total que **1300s** (*1302*), ce qui est moins que le gain nécessaire de **1845s**,
- si on se concentre sur le check **dump-check-example-2**:
 - il est le plus consommateur, ceci demande une optimisation de $1845/4135=45\%$ de son temps d'exécution, et ainsi passer sous les **110ms** de temps d'exécution moyen, ce qui est raisonnablement atteignable comme optimisation.



Dans notre cas, il faut donc un gain de **45%** sur le check **dump-check-example-2** afin de limiter la consommation CPU de la plateforme à 1 seul CPU.