Réalisation d'expériences avec Grid'5000

Simon Delamare[1] Pascal Morillon[2] Lucas Nussbaum[3]

[1]Laboratoire de l'informatique du parallélisme (LIP)
Université de Lyon

[2]Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)
Université de Rennes

[3]Laboratoire Lorrain de Recherche en Informatique et ses Applications (LORIA)
Université de Lorraine

Journées Réseaux de l'Enseignement et de la Recherche
14 novembre 2017

Grid'5000

jres Nantes 2017

Plan

Introduction

Expérimentation ?

La réalisation ≪d'expériences≫ est incontournable en informatique

- pour valider un déploiement, étudier les performances, tester...
- objet d'étude : un logiciel, une configuration, une infrastructure complexe

Introduction

Expérimentation ?

La réalisation ≪d'expériences≫ est incontournable en informatique

- pour valider un déploiement, étudier les performances, tester...
- objet d'étude : un logiciel, une configuration, une infrastructure complexe

Où ?

- Sur sa machine perso, avec de la virtualisation, etc.
- Sur les serveurs du labo, dans un Cloud

Introduction

Expérimentation ?

La réalisation ≪d'expériences≫ est incontournable en informatique

- pour valider un déploiement, étudier les performances, tester...
- objet d'étude : un logiciel, une configuration, une infrastructure complexe

Où ?

- Sur sa machine perso, avec de la virtualisation, etc.
- Sur les serveurs du labo, dans un Cloud

→ Ressources insuffisantes pour un déploiement large échelle

→ Virtualisation et containers $\neq$ Hardware

→ Mutualisation ? matériel, outils, bonnes pratiques

Introduction

Expérimentation ?

La réalisation ≪d'expériences≫ est incontournable en informatique

- pour valider un déploiement, étudier les performances, tester...
- objet d'étude : un logiciel, une configuration, une infrastructure complexe

Où ?

- Sur sa machine perso, avec de la virtualisation, etc.
- Sur les serveurs du labo, dans un Cloud

→ Ressources insuffisantes pour un déploiement large échelle

→ Virtualisation et containers $\neq$ Hardware

→ Mutualisation ? matériel, outils, bonnes pratiques

→ Grid'5000

Introduction

Grid'5000

- Une plateforme pour l'expérimentation
  - → À destination de la recherche autour de l'informatique
- Dans tous les domaines de l'informatique distribuée
  - Cloud Computing, Virtualisation, Systèmes distribués, HPC, Réseaux, etc.
  - Informatique ≪lourde≫ : Pas d'embarqué, pas de sans-fil.
- Propose à ses utilisateurs :
  - Du matériel varié . . .
  - . . . complètement reconfigurable selon leurs besoins
  - Un ensemble d'outils pour faciliter la réalisation d'expériences

Plan

Plan

Fonctionnement

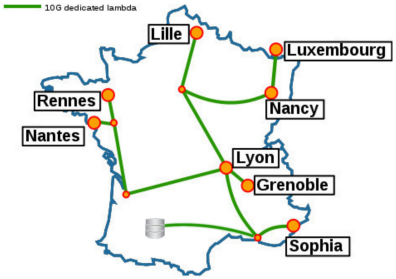
- Groupement d'Intérêt Scientifique
- Contributeurs : Principaux établissements de recherche FR
- Équipes de scientifiques pour orienter les évolutions de la plate-forme
- Équipe technique : 10 ingénieurs ETP

Chiffres :

- 12 ans d'existence
- 500 utilisateurs actifs / an
- 40M cœurs heures distribuées en 2016

→ Communauté active : site Web, mailing list, écoles et formations . . .

L'infrastructure Grid'5000



→ 8 sites

→ réseau de cœur 10Gbit/s

Poster !

Ressources matérielles :

- **Nœuds** groupés en **clusters** présents sur les **sites**
  - 900 nœuds (10k cœurs), ≈ 20 clusters
  - → *nova-5.lyon.grid5000.fr*
- Matériel varié :
  - CPUs : Différentes générations Intel (et AMD)
  - Réseau : Ethernet 10 Gbit/s, Infiniband, Omni-Path
  - Stockage SSD, grands volumes
  - Accélérateurs : GPU, Xeon Phi

Utilisation des Ressources

Accès exclusif au matériel, via un système de réservation

Utilisation des Ressources

Accès exclusif au matériel, via un système de réservation

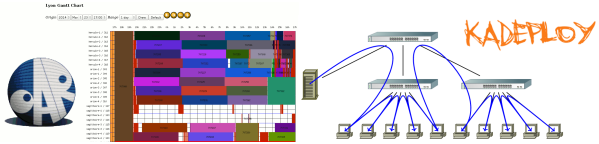
→ Y compris en tant que *root*

Utilisation des Ressources

Accès exclusif au matériel, via un système de réservation

→ Y compris en tant que *root*

→ Possibilité de déployer son propre OS



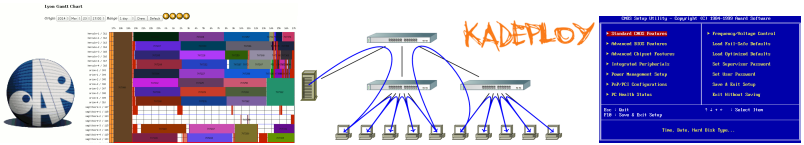
Utilisation des Ressources

Accès exclusif au matériel, via un système de réservation

→ Y compris en tant que *root*

→ Possibilité de déployer son propre OS

→ Modification de certains paramètres BIOS (config. CPU pour l'énergie . . .)



Utilisation des Ressources

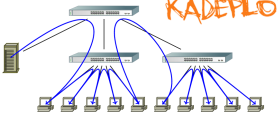
Accès exclusif au matériel, via un système de réservation

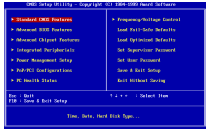
→ Y compris en tant que *root*

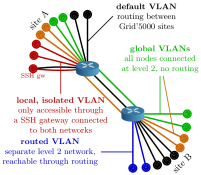
→ Possibilité de déployer son propre OS

→ Modification de certains paramètres BIOS (config. CPU pour l'énergie . . .)

→ Isolation réseau pendant l'expérience



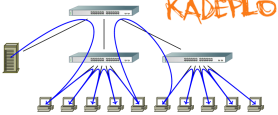


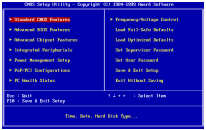


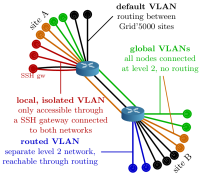
Utilisation des Ressources

Accès exclusif au matériel, via un système de réservation

- → Y compris en tant que *root*
- → Possibilité de déployer son propre OS
- → Modification de certains paramètres BIOS (config. CPU pour l'énergie . . .)
- → Isolation réseau pendant l'expérience







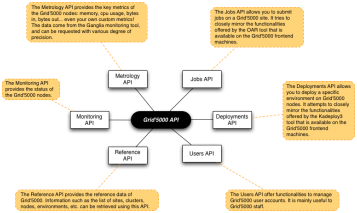
Modèle *Hardware as a Service* : on manipule du ≪vrai≫ matériel

Outils pour la réalisation d'expériences

Grid'5000 API :

- Alternative à la ligne de commande
- API REST
- Plusieurs composantes

→ Bibliothèque pour scripter l'utilisation de Grid'5000 dans un langage de haut niveau

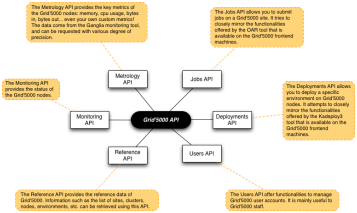


Outils pour la réalisation d'expériences

Grid'5000 API :

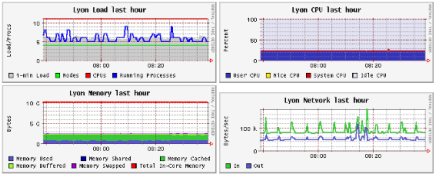
- Alternative à la ligne de commande
- API REST
- Plusieurs composantes

→ Bibliothèque pour scripter l'utilisation de Grid'5000 dans un langage de haut niveau



CPUs Total: 11
Hosts up: 4
Hosts unknown: 129
Hosts down: 6

Avg Load (15, 5, 1m):
54%, 56%, 56%
Localtime:
2008-08-08 08:38



Monitoring des performances :

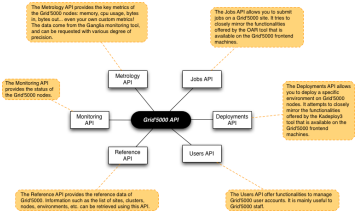
- Métrique ≪classique≫ du nœud : CPU, mémoire . . .
- Mais aussi, électricité consommée à la prise, réseau

Outils pour la réalisation d'expériences

Grid'5000 API :

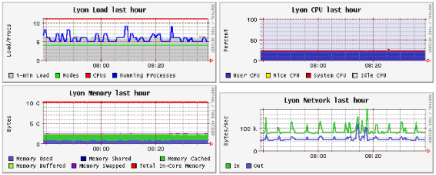
- Alternative à la ligne de commande
- API REST
- Plusieurs composantes

→ Bibliothèque pour scripter l'utilisation de Grid'5000 dans un langage de haut niveau



CPUs Total:	11
Hosts up:	4
Hosts unknown:	129
Hosts down:	6

Avg Load (15, 5, 1m):
54%, 56%, 56%
Localtime:
2008-08-08 08:38



Monitoring des performances :

- Métrique ≪classique≫ du nœud : CPU, mémoire . . .
- Mais aussi, électricité consommée à la prise, réseau

Déploiement automatisé d'infrastructures complexes :

- Proposé par l'équipe technique ou par les utilisateurs

→ *OpenStack, Cluster Ceph, Hadoop*

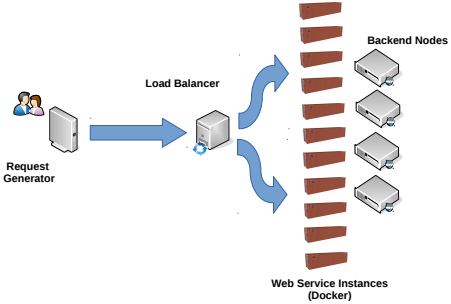
Plan

Plan

Exemple de scénario d'expérience

Description

Dimensionnement pour hébergement d'un service Web répliqué

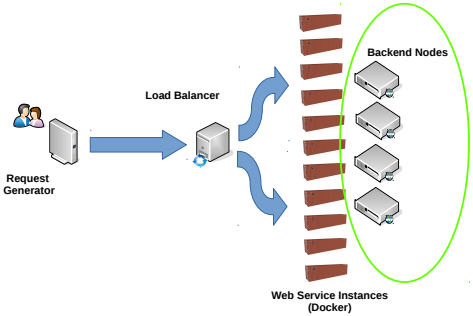


- Nombre de serveurs nécessaires pour héberger une appli Web
- En fonction du nombre de clients

Exemple de scénario d'expérience

Description

Dimensionnement pour hébergement d'un service Web répliqué

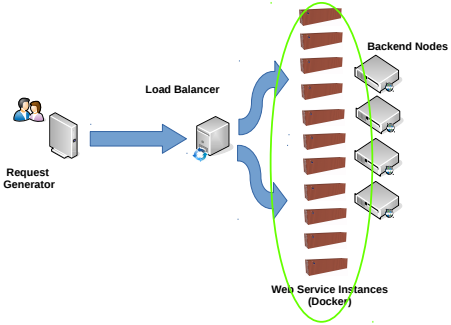


- Docker
- Une instance par cœur

Exemple de scénario d'expérience

Description

Dimensionnement pour hébergement d'un service Web répliqué



- *Jupyter NBViewer* : Service Web, interprète un Notebook Python, le retourne en HTML
  - Ports redirigés vers le *backend node*

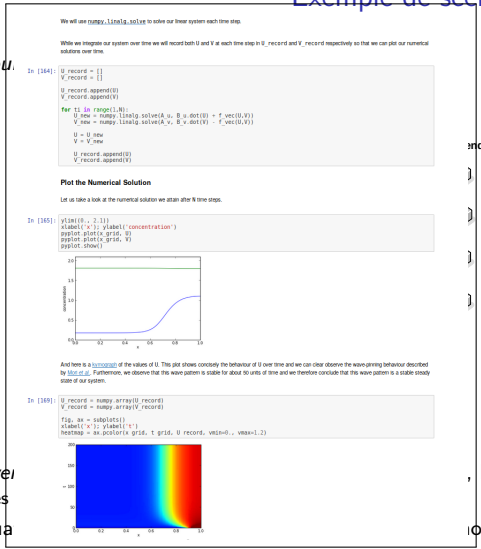
→ Résolution d'équation de la chaleur par la méthode de Crank Nicholson [1]

1. Voir `http://georg.io/2013/12/Crank_Nicolson` et `http://nbviewer.jupyter.org/github/waltherg/notebooks/blob/master/2013-12-03-Crank_Nicolson.ipynb`

Exemple de scénario d'expérience

Description

Dimensionnement pou



We will use `numpy.linalg.solve` to solve our linear system each time step.

While we integrate our system over time we will record both U and V at each time step in `U_record` and `V_record` respectively so that we can plot our numerical solutions over time.

In [164]:
```
U_record = []
V_record = []

U_record.append(U)
V_record.append(V)

for ti in range(1,N):
    U_new = numpy.linalg.solve(A_u, B_u.dot(U) + f_vec(U,V))
    V_new = numpy.linalg.solve(A_v, B_v.dot(V) - f_vec(U,V))

    U = U_new
    V = V_new

    U_record.append(U)
    V_record.append(V)
```

Plot the Numerical Solution

Let us take a look at the numerical solution we attain after N time steps.

In [165]:
```
ylim((0., 2.1))
xlabel('x'); ylabel('concentration')
pyplot.plot(x_grid, U)
pyplot.plot(x_grid, V)
pyplot.show()
```


And here is a kymograph of the values of U. This plot shows concisely the behaviour of U over time and we can clear observe the wave-pinning behaviour described by Mori et al. Furthermore, we observe that this wave pattern is stable for about 50 units of time and we therefore conclude that this wave pattern is a stable steady state of our system.

In [169]:
```
U_record = numpy.array(U_record)
V_record = numpy.array(V_record)

fig, ax = subplots()
xlabel('x'); ylabel('t')
heatmap = ax.pcolor(x_grid, t_grid, U_record, vmin=0., vmax=1.2)
```


…nd Nodes

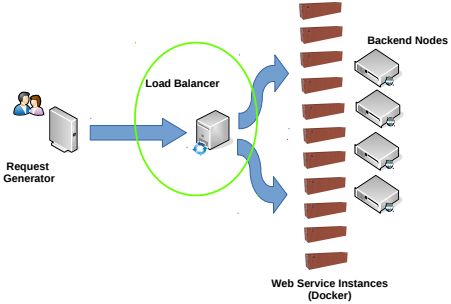
- *Jupyter NBViewe*, le retourne en HTML
  - Ports redirigés
→ Résolution d'équa…olson [1]

1. Voir http://georg.io/2013/12/Crank_Nicolson et http://nbviewer.jupyter.org/github/waltherg/notebooks/blob/master/2013-12-03-Crank_Nicolson.ipynb

Exemple de scénario d'expérience

Description

Dimensionnement pour hébergement d'un service Web répliqué



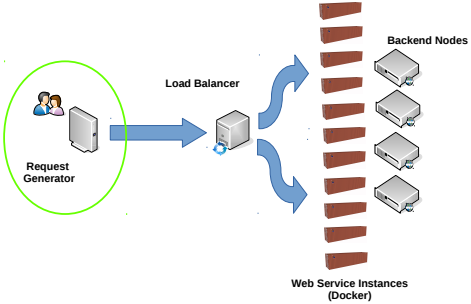
- Load Balancer avec NGINX

→ Round robin vers les instances Docker, via les *backends nodes*

Exemple de scénario d'expérience

Description

Dimensionnement pour hébergement d'un service Web répliqué



- Générateur de requête avec Locust
  - → Requêtes toutes les 5 secondes en moyenne × N clients.

Plan

Choix et réservation des ressources matérielles

- Page *Hardware* du site
- Via l'API Grid'5000 de description, en JSON

https://api.grid5000.fr/stable/sites/lyon/clusters/nova/nodes/nova-1?pretty


```
{
  "architecture": {
    "nb_cores": 16,
    "nb_procs": 2,
    "nb_threads": 32,
    "platform_type": "x86_64"
  },
  "bios": {
    "configuration": {
      "cstate_clc": true,
      "cstate_enabled": true,
      "ht_enabled": true,
      "turboboost_enabled": true
    },
    "release_date": "09/08/2016",
    "vendor": "Dell Inc.",
    "version": "2.2.5"
  },
  "chassis": {
    "manufacturer": "Dell Inc.",
    "name": "PowerEdge R430",
(...)
```


- Gestionnaire de ressource OAR
- Différents modes d'utilisation :
  - Accès interactif
  - Soumission d'un *job* et scheduling
  - Réservation à l'avance

- Exemple :

```
$ oarsub -t deploy -l nodes=10,walltime=2: -p "cluster='nova'" -r "2017-11-14 09:00" "./experiment.sh"
```

Installation et configuration des ressources

- Déploiement d'un système d'exploitation avec Kadeploy
  - Environnements mis à disposition par Grid'5000 (*jessie-x64-base* ...)
- Exemple :

```
$ kadeploy3 -k -m nova-[1-10] -e jessie-x64-base
```

- Ensuite, installation des logiciels et configuration
  - Via *SSH*, en tant que *root*

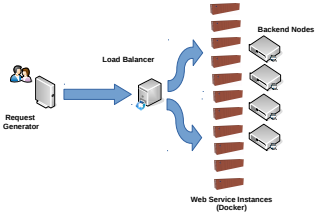
Exemple :

- nova-1 : *Request Generator* → `pip install locustio`
- nova-2 : *Load Balancer* → `apt-get install nginx; vim /etc/nginx/sites-enabled/lb.conf` ...
- nova-[3-10] : *Backends Nodes* → `apt-get install docker-ce; docker pull jupyter/nbviewer; docker run` ...

Exécution de l'expérience

et collecte des résultats

Exécution de Locust sur le *Request Generator*, via SSH (ex : 100 clients pendant 1 min.)



→ Transite par le *Load Balancer*, les noeuds de *Backend* et les instances Docker, qui exécutent le NoteBook et font la convertion HTML, puis retour au client

Locust retourne des statistiques sur les requêtes (durées moyennes, percentiles, taux d'échecs, etc.)

Collecte des résultats :

- Récupération des fichiers d'output de Locust
- Information de performance via l'API Grid'5000 (ou interface Web)
  - CPU, réseau, mémoire, énergie, etc.
- Exemple, métrique de consommation électrique :

```
curl https://api.grid5000.fr/3.0/sites/lyon/metrics/power/timeseries?only=nova-3,nova-4&from=1506345459&to=1506345489
```

Plan

Mise en place d'une campagne d'expérimentation

- Campagne d'expérimentation = Répétition de l'exécution du scénario
- Souvent nécessaire :
  - Variance des résultats → Erreur statistique
  - Étude de l'influence des conditions expérimentales sur les résultats

Mise en place d'une campagne d'expérimentation

- Campagne d'expérimentation = Répétition de l'exécution du scénario
- Souvent nécessaire :
  - Variance des résultats → Erreur statistique
  - Étude de l'influence des conditions expérimentales sur les résultats

→ Script d'exécution de l'expérience

→ Paramètres d'expérience

Script d'exécution

- Exécution manuelle et interactive : pas satisfaisant
- Script pour l'automatisation
  - Moins d'erreur
  - Reproductibilité des résultats de recherche
- Shell ou (mieux) langage de haut niveau
  - Execo [1]
  - Ruby-Cute [2]
- Exemple

1. http://execo.gforge.inria.fr/doc/latest-stable/index.html
2. https://github.com/ruby-cute/ruby-cute

Script d'exécution

- Exécution manuelle et interactive : pas satisfaisant
- Script pour l'automatisation
  - Moins d'erreur
  - Reproductibilité des résultats de recherche
- Shell ou (mieux) langage de haut niveau
  - Execo [1]
  - Ruby-Cute [2]
- Exemple

Fonction principale du programme :

```
if __name__ == "__main__":
    (client, lb, backends) = reserve_resources()
    install_resources(client, lb, backends)
    configure_scenario(client, lb, backends)
    start_scenario(client, lb, backends)
    collect_results(client, lb, backends)
    clean_scenario(client, lb, backends)
```

Script d'exécution

- Exécution manuelle et interactive : pas satisfaisant
- Script pour l'automatisation
  - Moins d'erreur
  - Reproductibilité des résultats de recherche
- Shell ou (mieux) langage de haut niveau
  - Execo [1]
  - Ruby-Cute [2]
- Exemple

Fonction de réservation des ressources :

```
def reserve_resources():
    logger.info("Submitting OAR job ...")
    jobid, site = oarsub([
        (OarSubmission(resources="nodes=10",
                       walltime="1:00:00",
                       job_type="deploy",
                       sql_properties="cluster='nova'"
                ), 'lyon')
        ], abort_on_error=True)[0]

    logger.info("Getting OAR node list from %s@%s ..." % (jobid, site))
    nodes = sorted(get_oar_job_nodes(jobid, site, timeout=180),
                   key=lambda h: str(h))

    logger.info("Node list is %s" % nodes)
    return nodes[0], nodes[1], nodes[2:]
```

Paramètres de l'expérience

- Environnement d'exécution décrit par un ensemble de paramètres
- Étude de l'influence sur résultat → réalisation de l'expérience avec différentes combinaisons de paramètres
  - ex : matériel utilisé (= cluster), nombre de noeuds *backend*, nombre de clients

→ Facile si utilisation d'un script dans langage de haut niveau

- Exemple

Paramètres de l'expérience

- Environnement d'exécution décrit par un ensemble de paramètres
- Étude de l'influence sur résultat → réalisation de l'expérience avec différentes combinaisons de paramètres
  - ex : matériel utilisé (= cluster), nombre de noeuds *backend*, nombre de clients

→ Facile si utilisation d'un script dans langage de haut niveau

- Exemple

Fonction principale du programme modifiée :

```
if __name__ == "__main__":

    num_clients_p = [10, 100, 1000, 10000]
    num_backends_p = [1, 3, 5, 10]

    (client, lb, backends) = reserve_resources(num_backends=max(num_backends_p))
    install_resources(client, lb, backends)

    for num_clients, num_backends in product(num_clients_p, num_backends_p):
        configure_scenario(client, lb, backends[:num_backends])
        start_scenario(client, lb, backends[:num_backends], num_clients)
        collect_results(client, lb, backends[:num_backends],
            result_dir="./jres_xp/c%d-b%d" % (num_clients, num_backends))
        clean_scenario(client, lb, backends[:num_backends])
```

Paramètres de l'expérience

- Environnement d'exécution décrit par un ensemble de paramètres
- Étude de l'influence sur résultat → réalisation de l'expérience avec différentes combinaisons de paramètres
  - ex : matériel utilisé (= cluster), nombre de noeuds *backend*, nombre de clients

→ Facile si utilisation d'un script dans langage de haut niveau

- Exemple

La fonction de réservation des ressources modifiée :

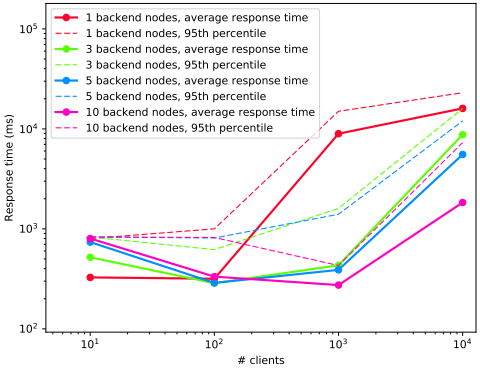
```
def reserve_resources(cluster='nova', num_backends=3):
    logger.info("Submitting OAR job...")
    jobid, site = oarsub([
        (OarSubmission(resources="nodes=%d" % int(num_backends+2),
                       walltime="1:00:00",
                       job_type="deploy",
                       sql_properties="cluster='%s'" % cluster
                ), get_cluster_site(cluster))
        ], abort_on_error=True)[0]

    logger.info("Getting OAR node list from %s@%s..." % (jobid, site))
    nodes = sorted(get_oar_job_nodes(jobid, site, timeout=180),
                   key=lambda h: str(h))
    logger.info("Node list is %s" % nodes)
    return nodes[0], nodes[1], nodes[2:]
```

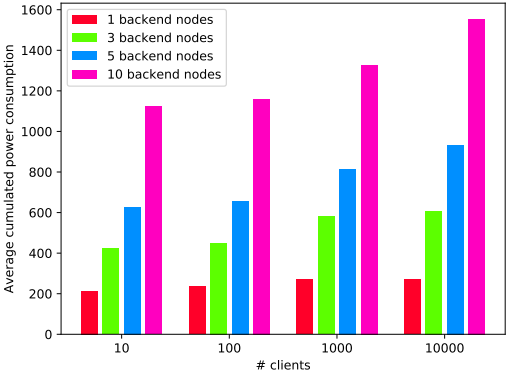
Plan

Résultats

- Exécution répétée de l'expérience, résultats agrégés



Temps de réponse moyen et du dernier décile nécessaire à l'application NBViewer pour servir un notebook Jupyter à ses clients



Consommation instantanée moyenne cumulée par le load balancer et les nœuds backend

→ Sur nova, OK avec 3 *backends* pour 1000 clients, consommation 600W

Script complet en ligne : `https://gitlab.inria.fr/delamare/jres17_experiment`

Plan

Conclusion

- Grid'5000 : une plateforme pour la réalisation d'expérience à grande échelle
  - des ressources matérielles
  - des outils pour la réalisation d'expérimentation

Conclusion

- Grid'5000 : une plateforme pour la réalisation d'expérience à grande échelle
  - des ressources matérielles
  - des outils pour la réalisation d'expérimentation

→ réalisation d'une expérimentation illustrée dans un scénario d'exemple
  - (petite partie des possibilités)

Conclusion

- Grid'5000 : une plateforme pour la réalisation d'expérience à grande échelle
  - des ressources matérielles
  - des outils pour la réalisation d'expérimentation

→ réalisation d'une expérimentation illustrée dans un scénario d'exemple
  - (petite partie des possibilités)

- Obtenir un accès à la plateforme
  - `https://www.grid5000.fr/open-access`
  - (de plein droit pour ESR français)

  → Ensuite, tutoriels disponibles sur le site Web (*Getting Started*, ...)