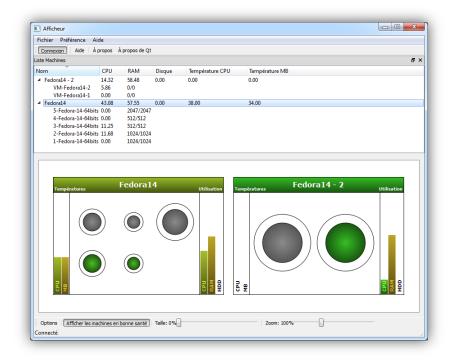
## Monitoring de machines virtuelles sous Linux-KVM

Projet de Bachelor : présentation





#### Plan

- Objectifs du projet et présentation
- Répartition du temps de travail
- Linux-KVM
- Outils de mesure d'un système Linux
- Architecture du superviseur
- Scénarios et logiciels associés
- Développement et structure du superviseur
- Conclusion
- Questions

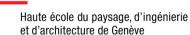


### Objectifs du projet et présentation

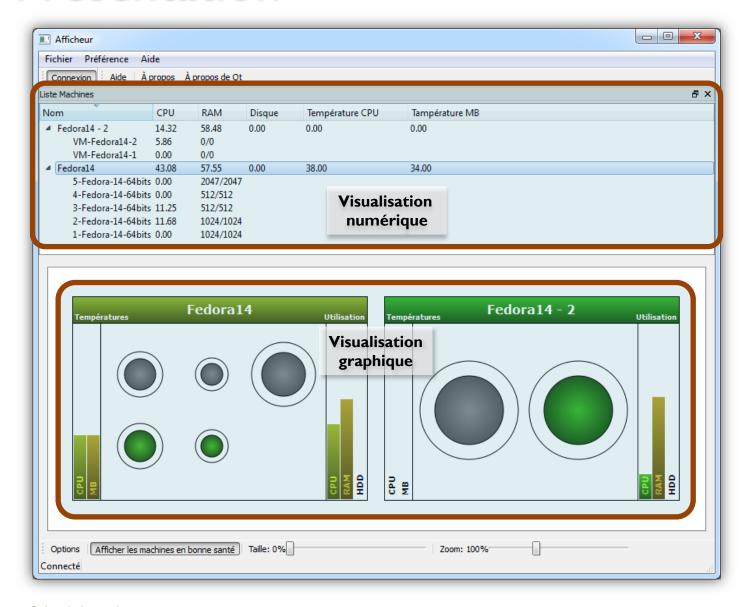
Projet Visag de la HES-SO. Travail de M. Pasche Sébastien.

#### Objectifs:

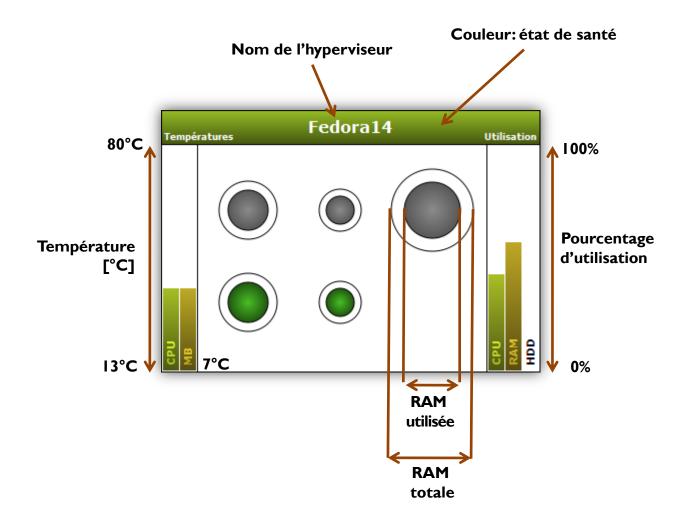
- Identifier les besoins dans la supervision d'une grille d'hyperviseur.
- Définir des scénarios de test.
- Développer un logiciel permettant de visualiser l'état de santé d'une grille d'hyperviseurs Linux-KVM.



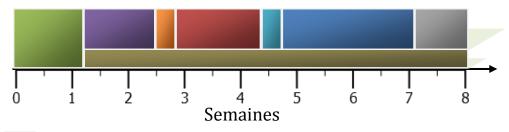
#### Présentation



#### Présentation



## Répartition du temps de travail



- Prise en main de Fedora et des outils de virtualisation.
- Rédaction du mémoire.
- Étude du travail de M. Sébastien Pasche et étude des outils de mesure de performances.
- Mise au point des scénarios.
- Développement des outils de génération de charge.
- Élaboration de l'architecture du superviseur.
- Développement des logiciels composant le superviseur.
- Mise en pratique des scénarios, génération de la documentation.

#### Linux-KVM

- Utilise Intel-VT ou AMD-SVM
- Réutiliser ce qui existe déjà
- Supporte la paravirtualisation (Virtio)
- Libvirt (gui/cli)
- Concurrent libre de VMware ESX(i)

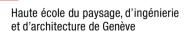


#### Outils de mesure d'un système Linux

- ~20 outils étudiés.
- 4 outils retenu:
  - free
  - iostat
  - Top
  - virt-top

```
top - 15:48:21 up 5 days, 20:21, 12 users, load average: 0.02, 0.06, 0.05
                  2 running, 216 sleeping,
Tasks: 218 total,
                                            0 stopped,
Cpu(s): 7.8%us, 4.3%sy, 1.3%ni, 86.6%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
      4020256k total, 3955236k used,
                                       65020k free,
                                                     934500k buffers
Swap: 6127612k total,
                        36404k used, 6091208k free,
                                                     463024k cached
 PID USER
11898 dev
                                12m S 6.3 3.7 488:30.01 pvthon
8891 qemu
                   0 809m 536m 3612 S 4.6 13.7 201:03.98 gemu-kvm
8931 gemu
                   0 1395m 945m 3636 S 4.6 24.1 191:46.88 gemu-kvm
11846 dev
                   0 522m 47m 11m S 4.6 1.2 421:41.98 gnome-system-mo
                   0 190m 48m
                                27m S 4.0 1.2 257:59.09 Xorg
1704 root
2269 dev
              39 19 466m 17m 13m S 3.0 0.5 34:45.71 vino-server
1621 root
                   0 1040m 20m 3948 S 2.7 0.5 193:53.52 libvirtd
19272 root
                   0 244m 8276 4284 S 1.0 0.2 46:32.88 daemon.py
                               25m S 0.7 5.2 14:41.89 firefox
23634 dev
                   0 797m 204m
                                                 3:42.69 kondemand/1
1235 root
                                  0 S 0.3 0.0
18802 root
                   0 15232 1208 832 R 0.3 0.0
                                                 0:00.30 top
```

- Les 3 premiers lisent /proc (pseudo-système de fichier).
- virt-top utilise l'API libvirt.



#### Outils de mesure d'un système Linux

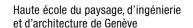
- Processeur
  - /proc/stat (cpu idle: ligne cpu, 4<sup>ème</sup> valeur)

$$\circ \quad utilisatio \ n[\%] = \left(1 - \frac{(cpu_B - cpu_A)}{100*(time_B - time_A)*\_SC\_CLK\_\_TCK}\right) * 100$$

- Mémoire RAM
  - /proc/meminfo (lignes MemTotal, MemFree, Buffers, Cached)

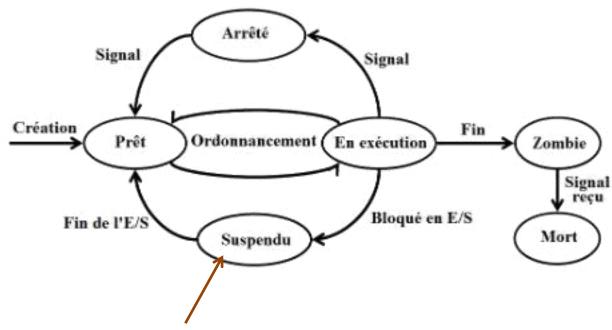
• 
$$utilisation[\%] = \frac{MemTotal - MemFree - Buffers - Cached}{MemTotal} *100$$

- Disque
  - /proc/diskstats (temps cpu: ligne sda, 10<sup>ème</sup> valeur)
  - $\circ \quad utilisatio \ n[\%] = \frac{(cpu_B cpu_A)}{10*(time_B time_A)}$
- Machines virtuelles: API libvirt



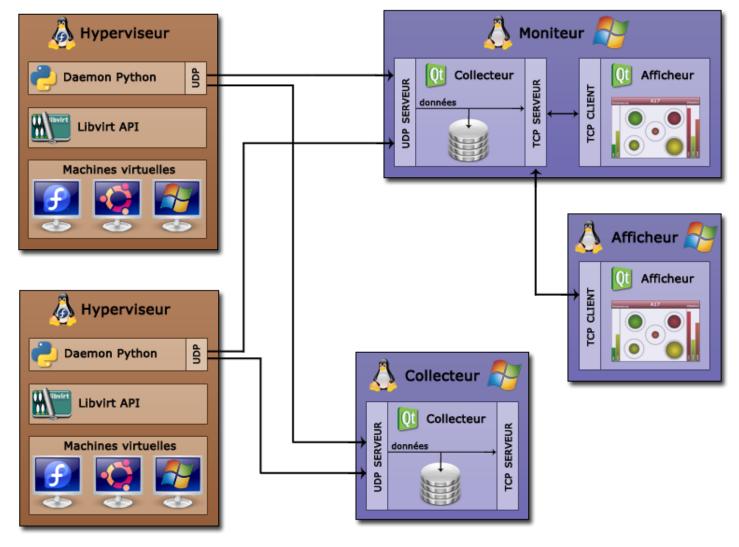
#### Outils de mesure d'un système Linux

• États d'un processus Linux



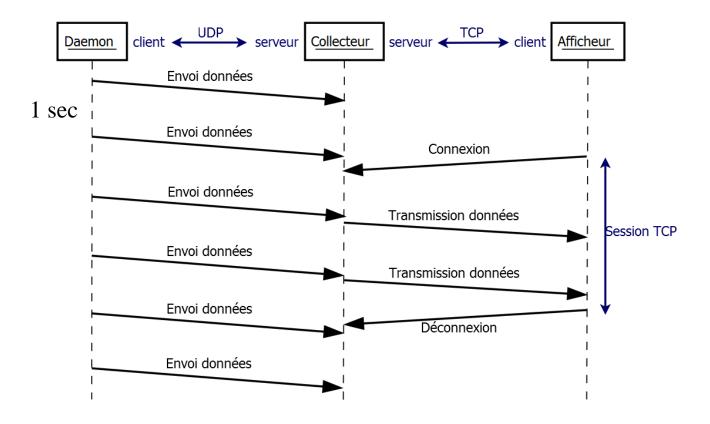
Valeur obtenue grâce à /proc/diskstats: temps passé dans l'état suspendu

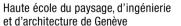
#### Architecture du superviseur



#### Architecture du superviseur

Communications entre les trois sous-programmes

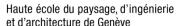




#### Architecture du superviseur

- Communication
  - Format JSON (JavaScript Object Notation)

```
"cputemp": 37.0, // commande sensors
"mbtemp": 36.0, // idem
"ram": 17.27959612522188, // /proc/meminfo
"vms": [ // libvirt
             "maxram": 1048576,
             "name": "1-Fedora-14-64bits",
             "ram": 1048576,
             "state": 5,
             "cpu": 0.0,
             "uuid": "87c63557-2a88-d1ee-b38b-9d7211f0256d"
      }, {
             "maxram": 1048576,
             "name": "2-Fedora-14-64bits",
             "ram": 1048576,
             "state": 5,
             "cpu": 0.0,
             "uuid": "6cdbd6d2-266d-c568-bc78-7aaab689c0e5"
"time": 1308674056.734489,
"disk": 0.0, // /proc/diskstats
"cpu": 5.350496117327408, // /proc/stat
"name": "Fedora14" // /etc/sysconfig/network
```



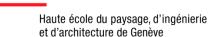
#### Architecture du superviseur

- Communication
  - Format JSON
  - Liaison Daemon-Collecteur format données:



Liaison Collecteur-Afficheur format données:





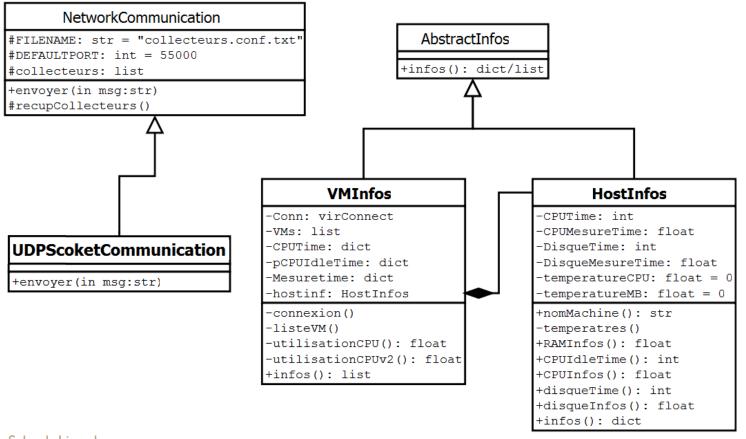
## Scénarios et logiciels associés

- But: Valider le fonctionnement et l'utilité du superviseur.
- Réalisation de logiciels de génération de charge.
- Exemple de scénario réalisé:
  - Démo I: Varier la charge CPU d'une VM entre 30%, 60% et 90%.
  - Démo 2: Varier la charge RAM d'un hyperviseur (2.5Go, 3Go, 3.5Go).
  - Démo 3: Générer une charge disque sur un hyperviseur (50%)

29.06.2011

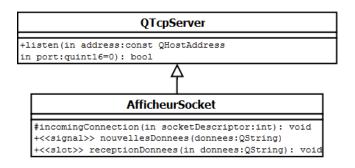
## Développement et structure du superviseur

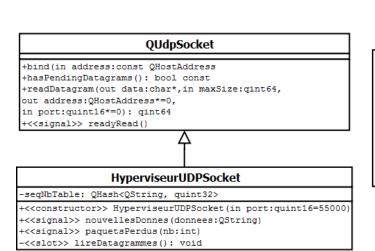
Daemon : diagramme de classe



## Développement et structure du superviseur

Collecteur : diagramme de classe





## +setSocketDescriptor(in socketDescriptor:int, in socketState:SocketState=ConnectedState, in openMode:OpenMode=ReadWrite): bool +error(): SocketError const +write(in byteArray:const QByteArray AfficheurSocketThread <<constructor>> AfficheurSocketThread(in socketDescriptor:int) +<<signal>> erreur(socketError:QTcpSocket::SocketError) +<<slot>> receptionDonnees(in donnees:QString): void QSqlDatabase \$addDatabase(type:QString, connectionName:QString): QSqlDatabase

GestionBDD

-executerRequete(in reqStr:QString,in listeColonnes:QStringList,

+<<slot>> ajouterDonnees(in donnees:QString): void

-verifierErreur(in requete:QSqlQuery): bool

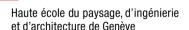
in liste:QList<QVariantMap>): QVariant

OTcpSocket

Étudiant: Schaub Lionel
Professeur: Gérald Litzistorf

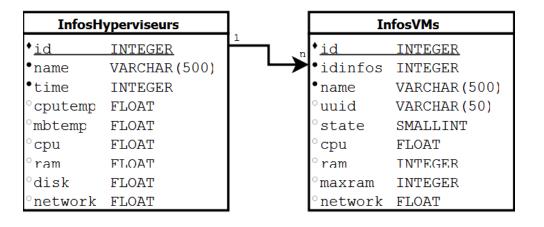
-bdd: QSqlDatabase

-creerBDD(): void



## Développement et structure du superviseur

Collecteur : schéma relationnel

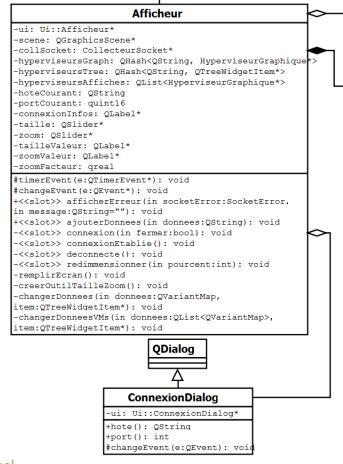


## Développement et structure du superviseur

Afficheur:

QMainWindow

# liagramme de classe



QTcpSocket

+connectToHost(in hostName:const QString,
in port:quint16=ConnectedState,
in openMode:OpenMode=ReadWrite): void
+abort(): void const
+read(in maxSize:qint64): QByteArray
+bytesAvailable(): qint64 const
+<<signal>> readyRead()

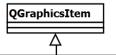
CollecteurSocket

-tailleDonnees: quint32 +<<signal>> nouvellesDonnees(donnees:QString)

-<<slot>> envoyerCommande(in commande:QString): bool

+<<slot>> recevoirDonnees(): void

+<<slot>> connexion(in host:QString,in port:quint16)



#### HyperviseurGraphique

-etatSante: float

-derniereReception\_d: QTime

-donnees: QVariantMap

-donneesVMs: QList<QVariantMap>

+HyperviseurGraphique(in donnees:QVariantMap,

in donneesVMs:QList<QVariantMap>)

+name(): QString

+etatDeSante(): float

+derniereReception(): QTime

+changerDonnees(in donnees:QVariantMap,

in donneesVMs:QList<QVariantMap>): void

+boundingRect(): QRectF const

+paint(painter:QPainter\*,option:const QStyleOptionGraphicsItem

widget:QWidget\*): void

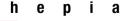
-calcBarreBrush(in pourcent:float): QBrush

+calcCercleBrush(in pourcent:float,

Carcellebrash(in pourcementions

in rayon:int,in centre:QPoint, in desactive:bool=false): OBrush

29.06.2011



#### Conclusion

Étudiant: Schaub Lionel

29.06.2011



#### Questions