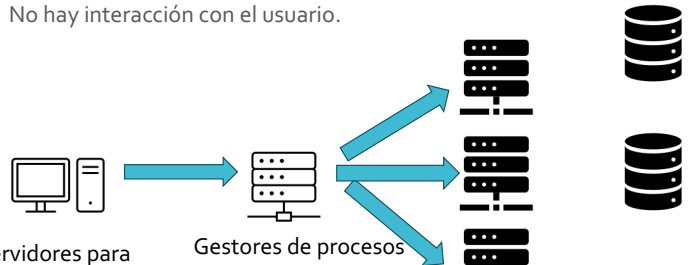
Introducción a la Granja de HPC del DTSC

1

Arquitectura

- Sistema basado en procesos BATCH
  - Desde un equipo se lanza la petición del proceso hacia el gestor de recursos
  - Este asigna un recurso (o recursos) y gestiona la ejecución
  - Distribuye la ejecución en los equipos de cómputo
  - No hay interacción con el usuario.



Servidores para Lanzar procesos:
- Servidor del Grupo

Gestores de procesos
- subserver1
- subserver2
- subserver3
- subserver4
- subserver5

2

¿Como lanzar un proceso a la granja

- Se usa un fichero JSON con la definición de la tarea
- Secciones
  - Cluster
  - Resources
  - Tasks
  - Frameworks

3

Sección Cluster

- Objetivo:
  - Indicar en cual de las dos granjas se lanzan los procesos
    - Granja de propósito general: 1240 Cores de propósito general.
      - Identificador master: **mesos**
    - Granja de GPUs: 32 GPUs
      - 8 GTX 1080Ti: 12GB Mem
      - 8 GTX 2080Ti: 12 GB Mem
      - 2 RTX A4000: 16 GB Mem
      - 10 RTX 3090: 24GB Mem
      - 4 RTX 4090: 24 GB Mem
    - Identificador master: **mesos_gpu**

```
"cluster": {
        "master":
"zk://10.0.12.18:2181,10.0.12.77:2181,10.0.12.60:2181,10.0.12.51:2181,10.0.12.75:2181,10.0.12.76:2181,10.0.12.78:2181/m
esos"
    },
```

4

Sección Resources

- **Objetivo**: Indicar los recursos necesarios para cada tarea:
- Se deben indicar los recursos necesarios en cada máquina para su ejecución, como:
  - Número de cores: CPUS
  - Memoria necesaria: MEM
  - GPUs a usar: GPUS

```
  "resources": {
    "CPUS": 4,
    "GPUS": 0,
    "MEMORY": 8192
},
```

Valores de recomendados:

- Número de CPUS:
  - En granja general: 4 por proceso
  - En granja de gpus: 5 por GPU
- Memoria:
  - En granja general: 4GBytes * CPU
  - En granja de GPUs: 32G por GPU

5

Sección TASK

- **Objetivo**: Definir como se ejecuta cada tarea en un nodo remoto
  - "**task_basename**": Nombre con el que se identificará cada tarea lanzada a los nodos remotos. El nombre se construirá a partir del task_basename + identificador de la tarea/numero total de tareas
  - "**cmdbase**": Define la sintaxis del comando que se debe ejecutar
  - "**var_parameters**": Conjunto de variables iterables
  - "**parameters**": Parámetros fijos

6

```
"tasks": {
    "task_basename"     : "Ejemplo_Matlab",
    "cmdbase"           : "{baseprogram} {funcion_matlab} {num_params} {var_param1} {param2} {var_param3} ",
    "var_parameters"    : {
        "var_param1"    : [3,6,8],
        "var_param3"    : "np.arange(10)"
    },
    "parameters"        : {
        "baseprogram"       : "MatlabFunctions",
        "funcion_matlab"    : "test",
        "num_params"        : 3,
        "param2"            : 100
    }
},
```

```
MatlabFunctions test 3 3 100 0
MatlabFunctions test 3 3 100 1
MatlabFunctions test 3 3 100 1
.
.
.
MatlabFunctions test 3 6 100 5
MatlabFunctions test 3 6 100 6

.
.
.
MatlabFunctions test 3 8 100 8
MatlabFunctions test 3 8 100 9
```

7

Sección Framework

- **Objetivo**: Definir los elementos del grupo de tareas a ejecutar
- "**name**": Nombre del Framework
- **"max_num_procs_simultaneos":** Máximo número de procesos simultaneos, si se desea limitar el número de tareas en ejecución. `-1` para no limitar el número
- **"max_failed_tasks"**: Máximo número de tareas que pueden fallar, antes de declarer el framework fallido. Los posibles valores son:
  - `0`: En caso de que una tarea falle, se para la ejecución de todas las tareas y del framework completo
  - `n>0` Pueden fallar hasta n tareas antes de parar todas las tareas y parar el Framework. En caso de que una tarea falle, se reasignará su ejecución.
  - `-1`: En caso de fallo de alguna tarea, se reasigna su ejecución. El framework no se para en ningún caso

8

- **"environment_vars":** Diccionario con las variables de entorno que se definirán en cada proceso remoto. En caso de procesos Matlab, es necesario exportar la variable HOME.
- **"uris":** Vector con el conjunto de ficheros que se distribuirán a cada uno de los procesos remotos.
- **"stdout":** Fichero donde se almacenará la salida de consola del proceso maestro.
- **"stderr":** Fichero donde se almacenará la salida de error del proceso maestro.
- **"notification":** 1 si desea que al finalizar las tareas se envíe un correo.
- **"email":** En caso de que "notification" sea 1, la dirección de correo de destino.
- **"max_num_task_per_host":** Máximo número de tareas

9

```
"framework" : {
    "name"                      : "Ejemplo Matlab",
    "max_num_procs_simultaneos" : -1,
    "max_failed_tasks"          : 0,
    "environment_vars"      : {
        "MATLABPATH"        : "/export/usuarios01/USUARIO/matlab",
        "HOME"              : "/export/usuarios01/USUARIO"
    },
    "uris"                  : [
        "file:///export/usuarios01/USUARIO/MatlbFunctions",
        "file:///export/usuarios01/USUARIO/test.m",
        "file:///export/usuarios01/USUARIO/data1.mat"
    ],
    "stdout"                : "stdout.log",
    "stderr"                : "stderr.log",
    "notification"          : 1,
    "email"                 : USUARIO@tsc.uc3m.es,
    "max_num_task_per_host" : 3

}
```

10

Autenticación en Framework

- **"credentials":** Valor del token obtenido de la URL indicada anteriorment
- **"credentials_file":** Fichero texto con el token obtenido de la URL arriba mencionada. Si se asegura razonablemente (usando el comando "chmod 400 FICHERO") soluciona estar definiendo variables o almacenando valores más sensibles en el disco.
- **"password":** El password del usuario almacenado en la definición de Frameworks. Es el método menos recomendable, ya que implica guardar el password de usuario en el fichero de definición de Frameworks.
- **"password_file":** Path completo a un fichero tipo texto con el password de usuario. No recomendable, ya que implica almacenar en abierto el password del usuario en un fichero. En caso de usar este método, asegurarse que el fichero tiene privilegios nivel 400 (solo lectura para el usuario).

11