Use spark on SCITAS cluster

Launching the cluster

The first step we have to take is to start a standalone master server. By default, when you submit a batch script asking multiple nodes, the linux hostname bash command will return the hostname of the first node allocated. It is only when the slurm command srun is invoked that the daemon propagate job to all the nodes.

Once the master is running we can start the worker as well. The script `start-cluster.sh` does exactly this job:

```bash
#!/bin/bash
MASTER=$1
HOST=$(hostname)
if [ "$HOST" == "$MASTER" ]; then
    $SPARK_ROOT/sbin/start-master.sh
fi
$SPARK_ROOT/sbin/start-slave.sh spark://$MASTER:7077
tail -f /dev/null #wait forever
```

Note that the `start-master.sh` script is launched only on the master node while the `start-slave.sh` is launched everywhere.

Testing the configuration on the provided examples

The following example shows how you can test the spark installation on SCITAS cluster. You can use the script below as template to build your own application.

Note that the `start-cluster.sh` is executed through `srun`.

```bash
# Example script to test spark installation
MASTER=$1
HOST=$(hostname)
if [ "$HOST" == "$MASTER" ]; then
    $SPARK_ROOT/sbin/start-master.sh
fi
$SPARK_ROOT/sbin/start-slave.sh spark://$MASTER:7077
```
#!/bin/bash
#SBATCH -N 3
#SBATCH -t 00:10:00

set -x

module load spark

MASTER=$(hostname)
echo $MASTER
export SPARK_LOCAL_DIRS=$(mktemp -d)
export SPARK_WORKER_DIR=$(mktemp -d)
export SPARK_LOG_DIR="spark-logs"
mkdir -p $SPARK_LOG_DIR

srun ../start-cluster.sh $MASTER &
sleep 60 #give the cluster some time to start

#submit your job
$SPARK_ROOT/bin/spark-submit \
   --master spark://$MASTER:7077 \
   pi.py

$SPARK_ROOT/sbin/stop-master.sh spark://$MASTER:7077