Introduction to LISA GPU

Damian Podareanu
7 September 2018

SURF SARA

About SURFsara

History:

- 1971: Founded by the VU, UvA, and CWI
- 2013: SARA (Stichting Academisch Rekencentrum A'dam) becomes part of SURF

Super/cluster-computing group:

- 8 consultants
- 16 members in total (including admins/system-experts)

Other activities:

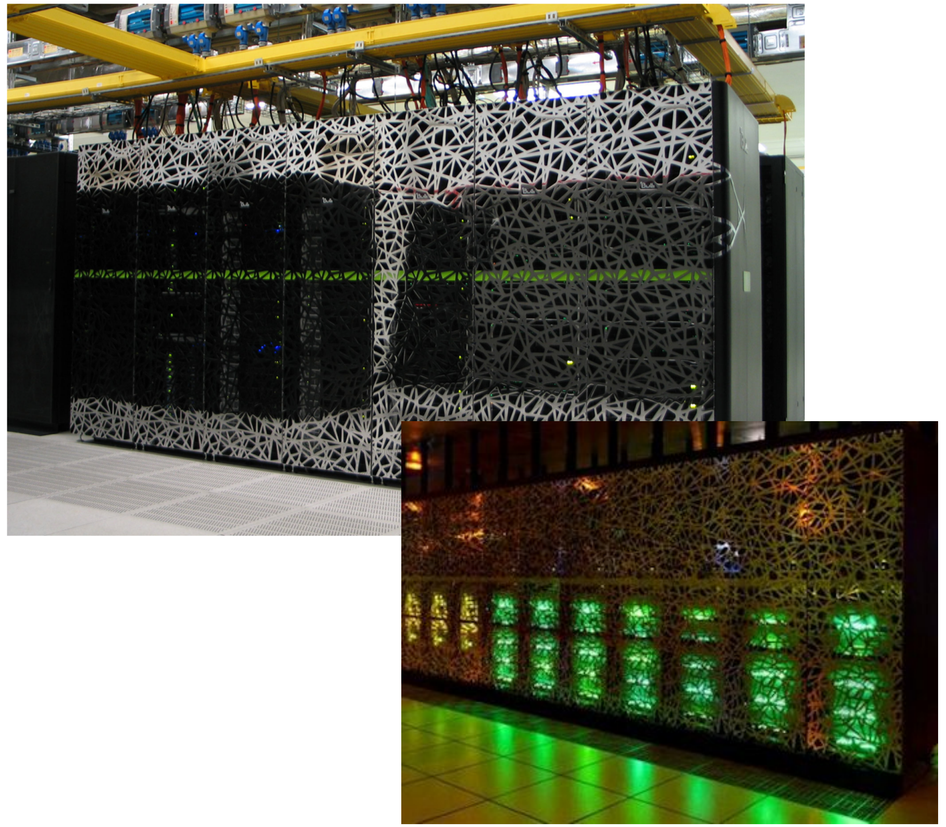
- HPC cloud / virtualisation
- Big Data
- Data services / storage
- Visualisation

SURF Open Innovation Lab
dr. Axel Berg
axel.berg@surfsara.nl

Our systems (1/2)

Cartesius (Bull supercomputer):

- 40.960 Ivy Bridge / Haswell cores: 1327 TFLOPS
- 56GBit/s Infiniband
- 64 nodes with 2 GPUs each: 210 TFLOPS
  - NVIDIA Tesla K40m GPU
  - 12GB GDDR5
  - GPU-Direct RDMA
- Accelerator island: #4 Green500 (June 2014)
- Broadwell & KNL extension (Nov 2016)
  - 177 BDW and 18 KNL nodes: 284TFLOPS
- 7.7 PB Lustre parallel file-system



Our systems (2/2)

LISA (Dell cluster):

- 7856 cores (16 cores per node, Xeon E5-2650)
- Peak performance: 149 TFLOPS
- 24 * 4 NVIDIA 1080TI

HPC cloud:

- Virtual machines
- Up to 64 cores and 2TB RAM

The archive:

- Tape-storage for long-term storage
- Virtually unlimited space

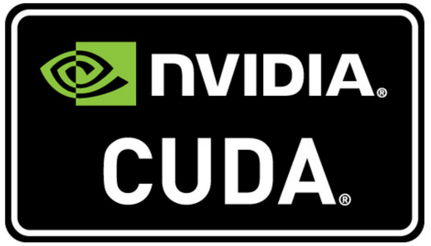
Others:

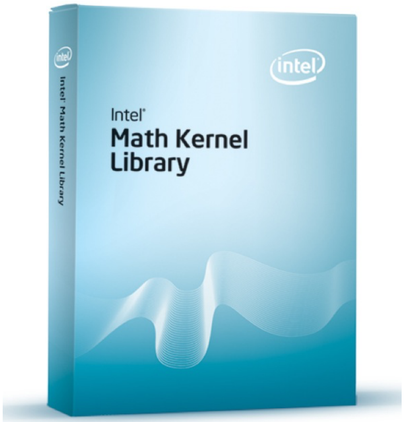
- Grid
- Hadoop

Challenges

Typical challenges:

- Bottleneck identification
- MPI/OpenMP parallelisation
- Inter-node communication
- I/O scaling
- GPU/Xeon Phi acceleration
- Algorithm optimization
- Vectorization





Approach:

- Discussions, scientific papers, manuals
- Hot-spot detection, timing analysis (manual)
- V-tune / Scalasca / score-p / Likwid / nvvp profiling (guided)

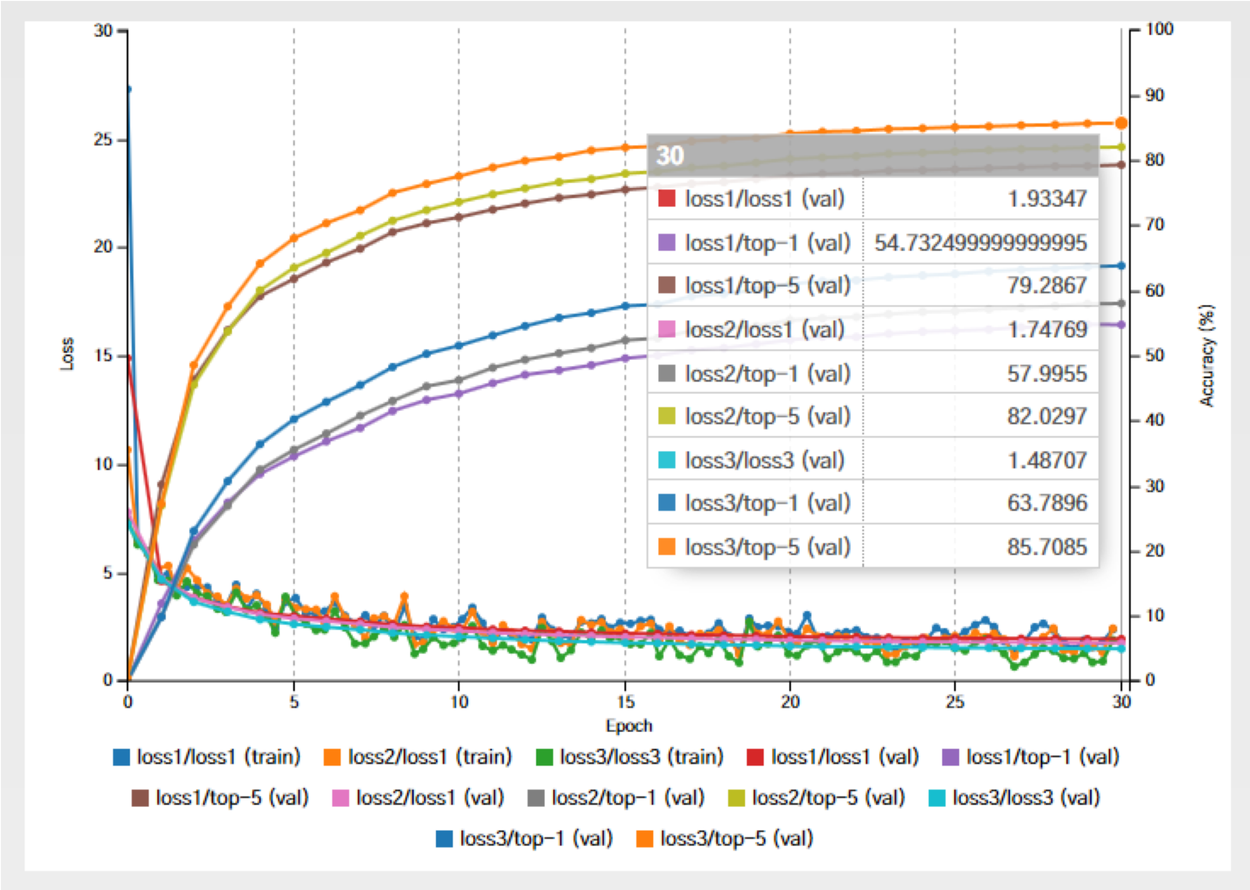
ML Software on Cartesius/LISA

Already installed software:

- Caffe
- Torch7
- Tensorflow
- Theano/Lasagne
- cuda-convnet2
- CNTK
- MXNet
- scikit-learn
- cuDNN
- NVIDIA DIGITS

Other software:

- Install yourself…
- …or ask us to install it

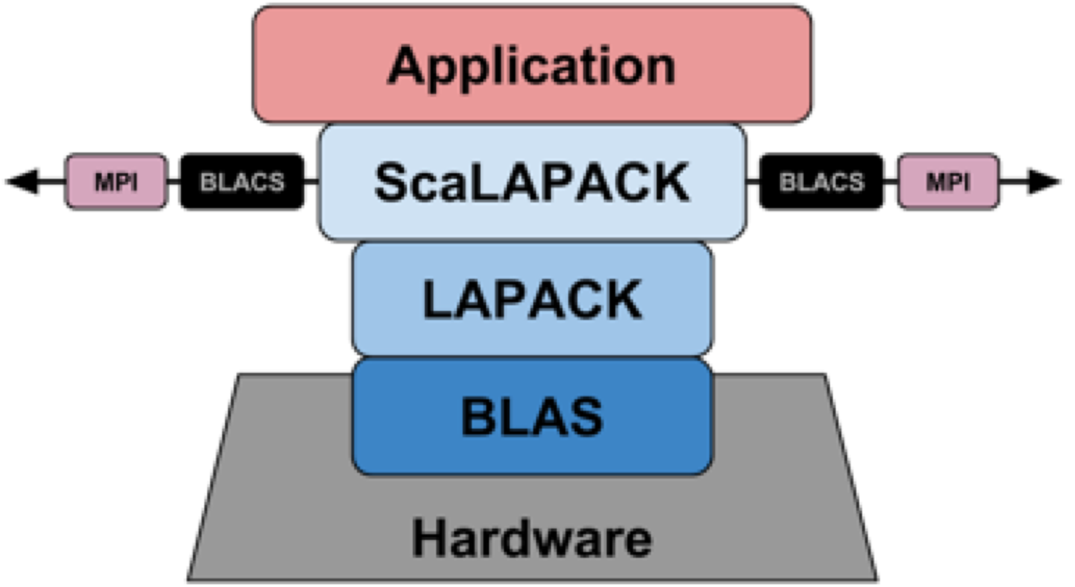


DIGITS training for GoogLeNet

Abstraction and the magic of “10 lines of code”

ConvnetJS, reinforce.js, mxnet.js

Theano, Caffe, Pylearn2, **Keras**, Lasagne
Mxnet, Chainer, OpenDeep



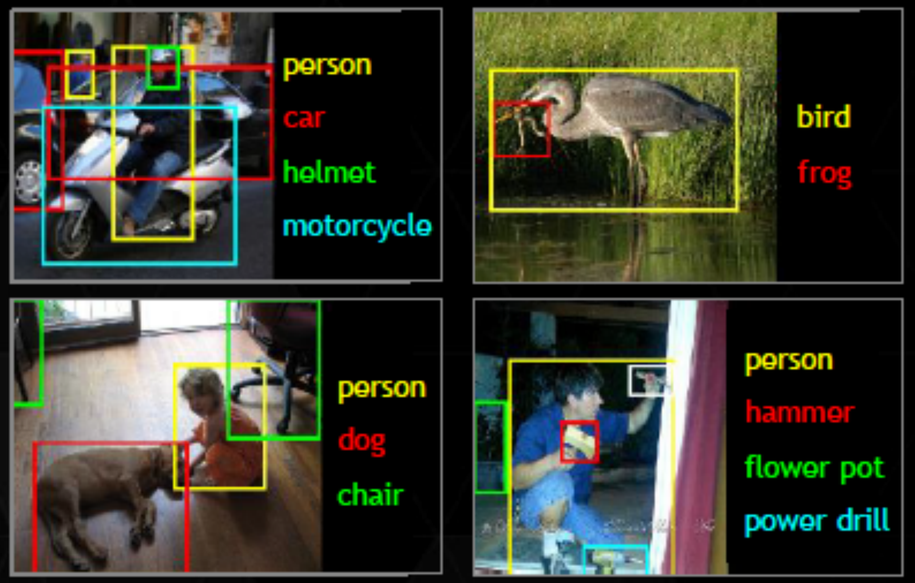
GPUs — THE PLATFORM FOR DEEP LEARNING

Image Recognition Challenge

1.2M training images • 1000 object categories

Hosted by

IMAGENET



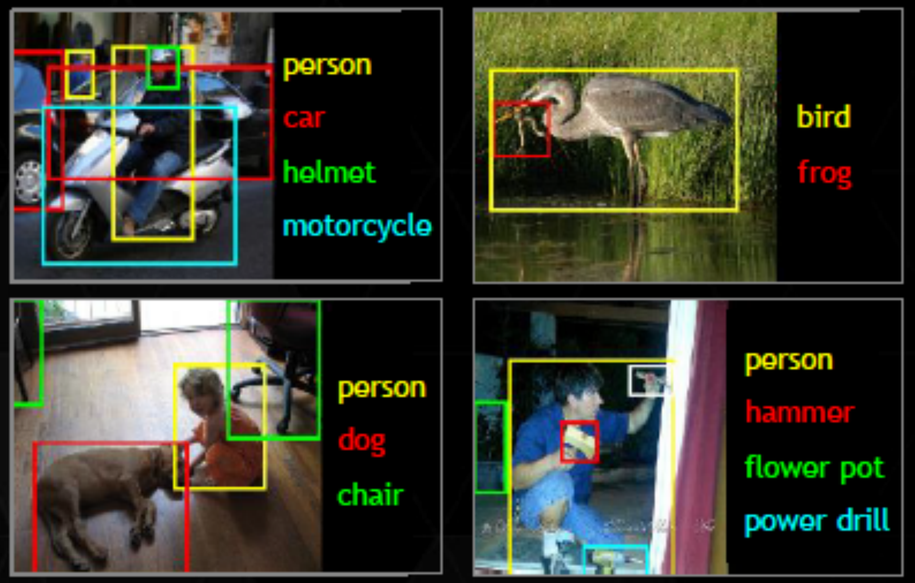
GPUs — THE PLATFORM FOR DEEP LEARNING

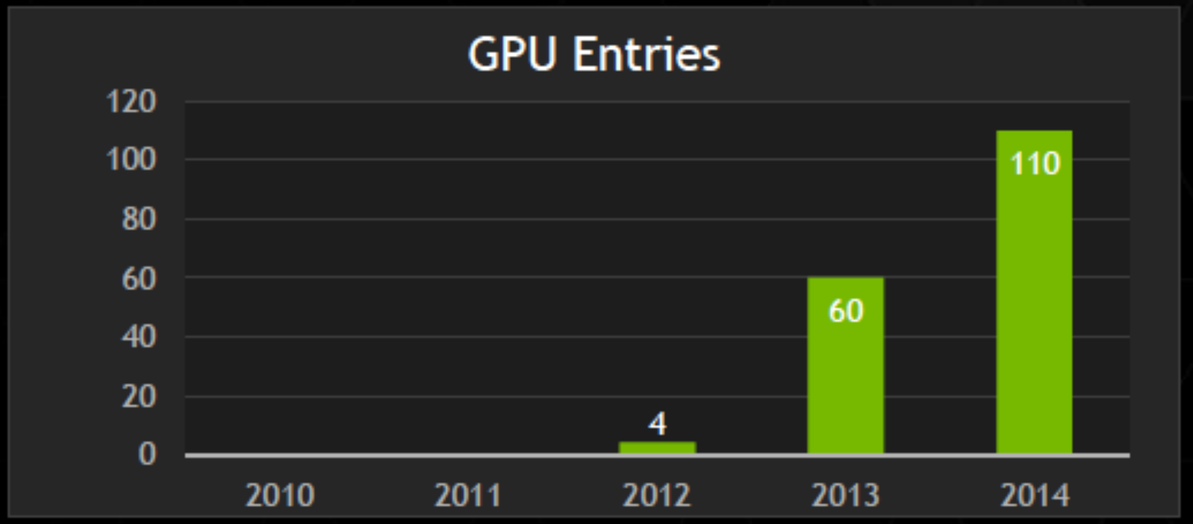
Image Recognition Challenge

1.2M training images • 1000 object categories

Hosted by

IMAGENET





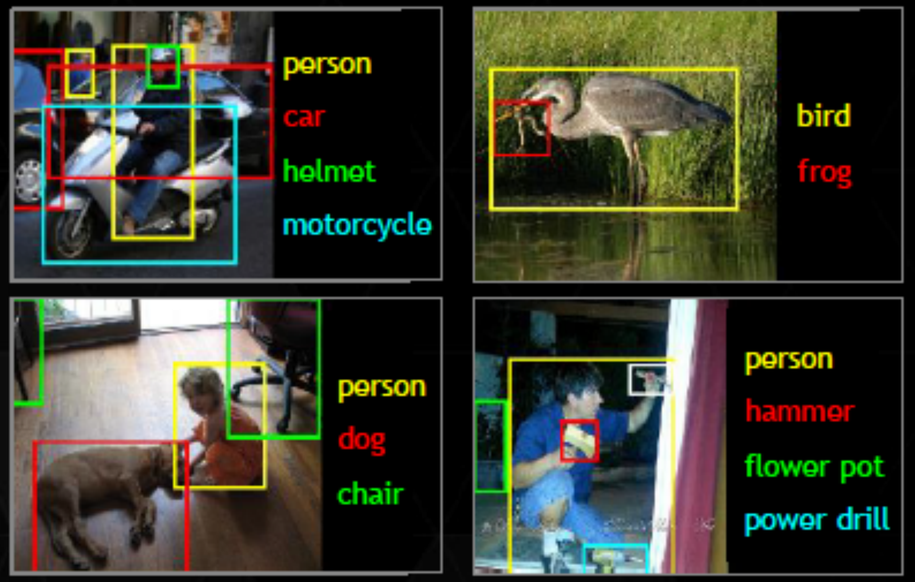
GPUs — THE PLATFORM FOR DEEP LEARNING

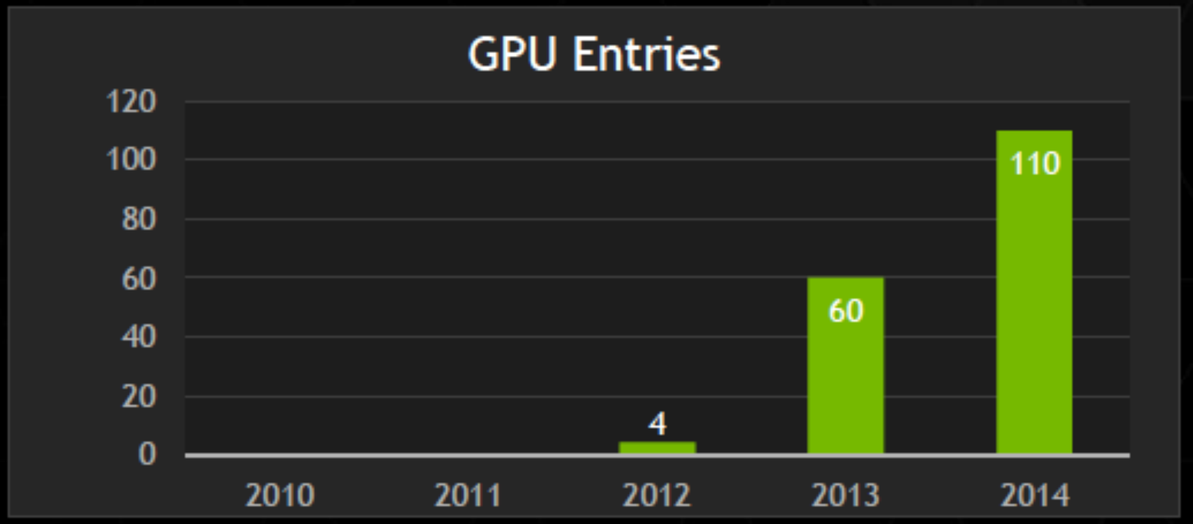
Image Recognition Challenge

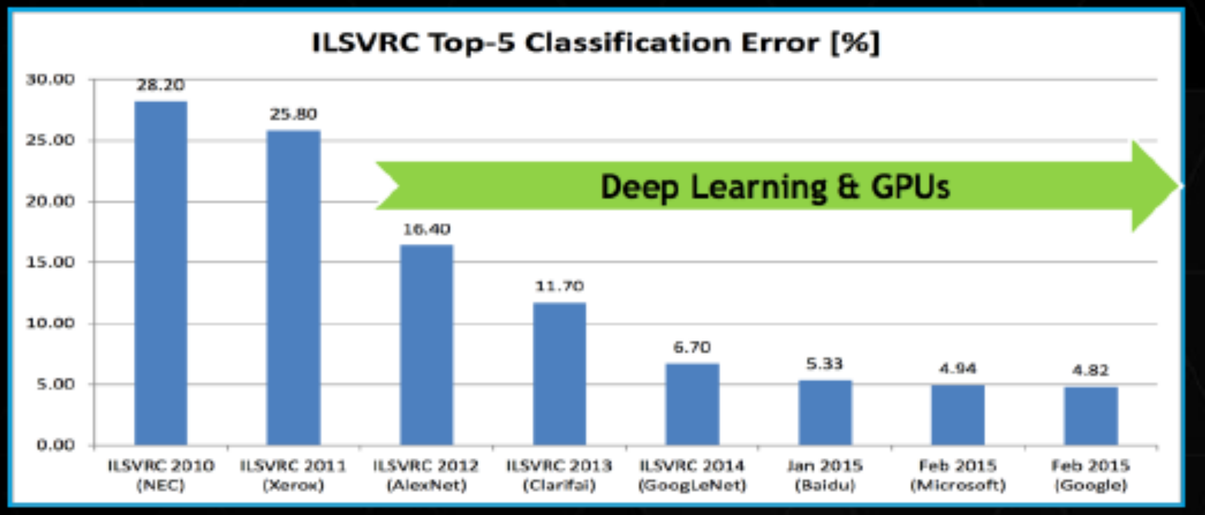
1.2M training images • 1000 object categories

Hosted by

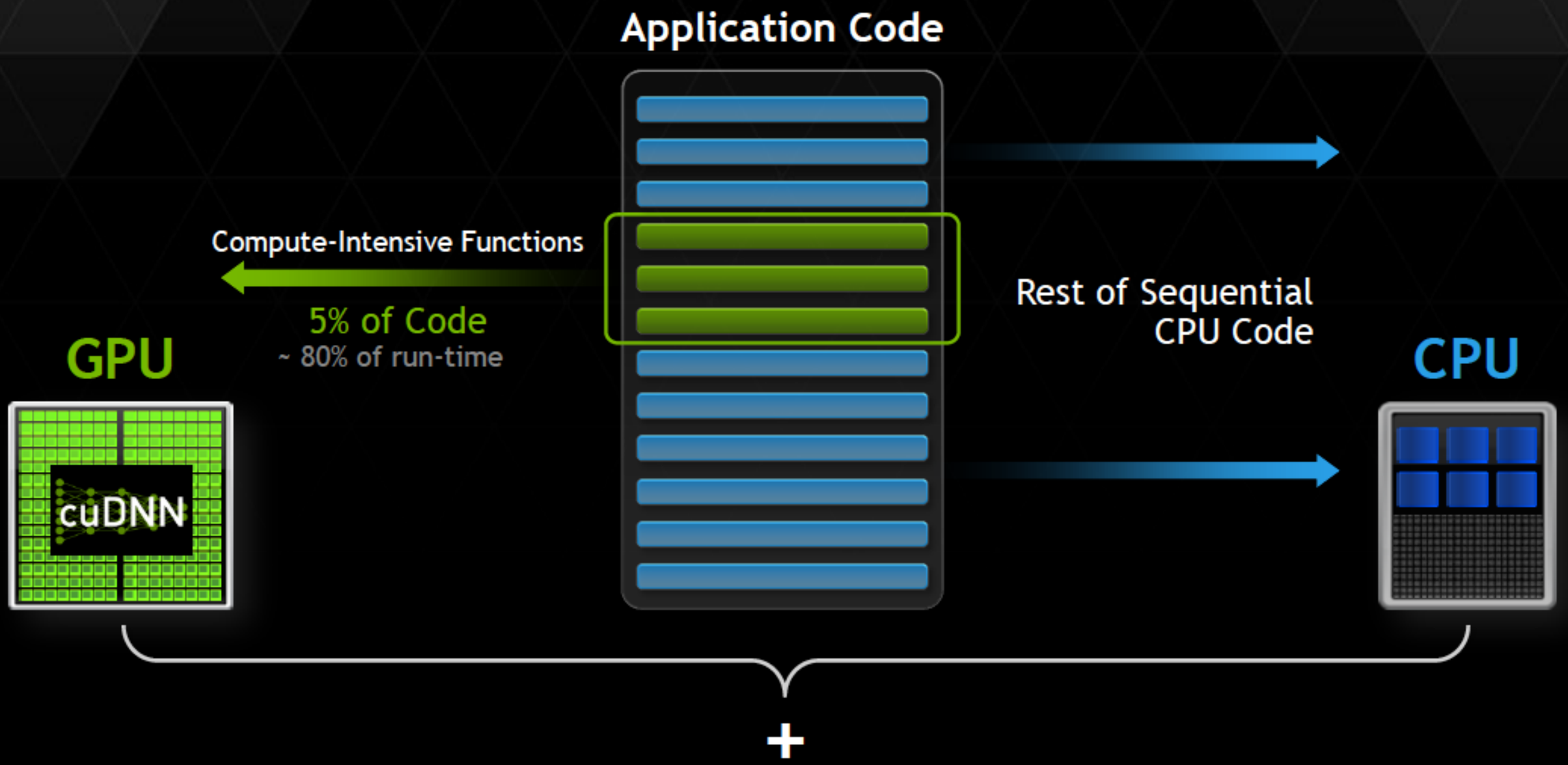
IMAGENET







HOW GPU ACCELERATION WORKS



Case Studies

19.6 GFLOPS

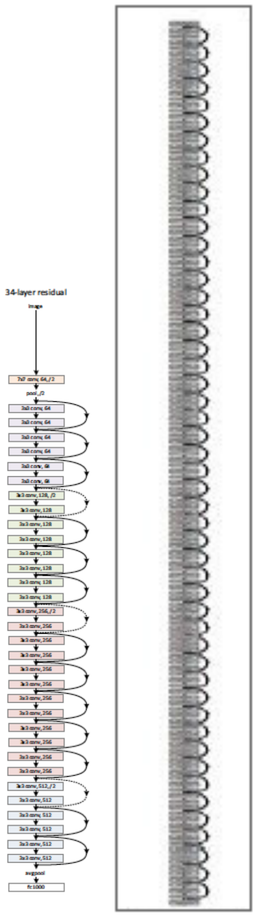
D	E
16 weight layers	19 weight layers
conv3-64	conv3-64
conv3-64	conv3-64
conv3-128	conv3-128
conv3-128	conv3-128
conv3-256	conv3-256
conv3-256	conv3-256
conv3-256	conv3-256
	conv3-256
conv3-512	conv3-512
conv3-512	conv3-512
conv3-512	conv3-512
	conv3-512
conv3-512	conv3-512
conv3-512	conv3-512
conv3-512	conv3-512
	conv3-512
maxpool	
FC-4096	
FC-4096	
FC-1000	
soft-max	

VGG (2014)

1.5 GFLOPS

GoogLeNet (2014)

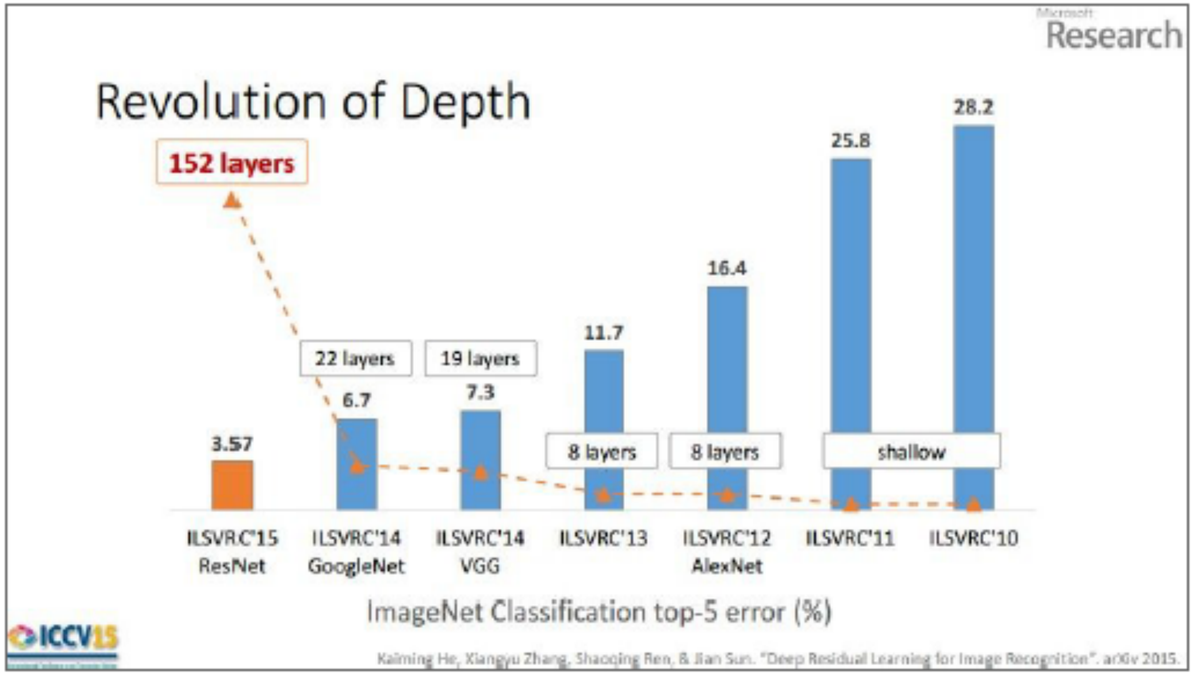
3.6-11.6 GFLOPS

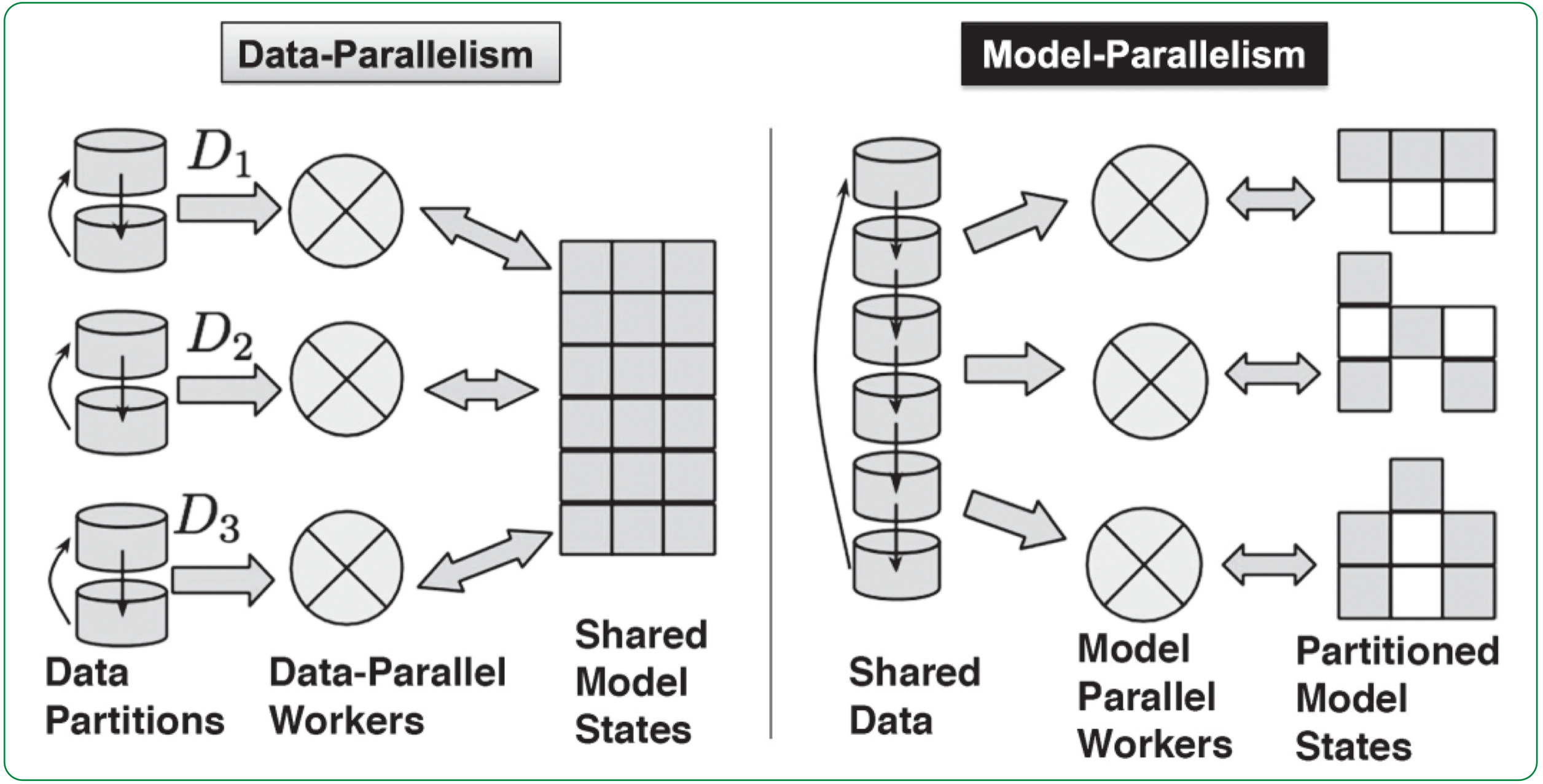


ResNet (2015)

Training VGG for 50 epochs on Imagenet uses more than 1 ExaFlop

True HPC distributed training is needed




Data-Parallelism
Model-Parallelism
D_1
D_2
D_3
Data Partitions
Data-Parallel Workers
Shared Model States
Shared Data
Model Parallel Workers
Partitioned Model States

Practical

Introduction to Linux and Connecting to LISA

Install UNIX tools on your laptop

- *Windows*
- MobaXterm (http://mobaxterm.mobatek.net)

- *Mac OSX*
- Terminal (pre-installed in /Applications/Utilities)
- XQuartz (http://www.xquartz.org)

- *Linux*
- You are already well equipped!

Remote Login (SSH) to SURFsara

- Authentication on (some) SURFsara systems

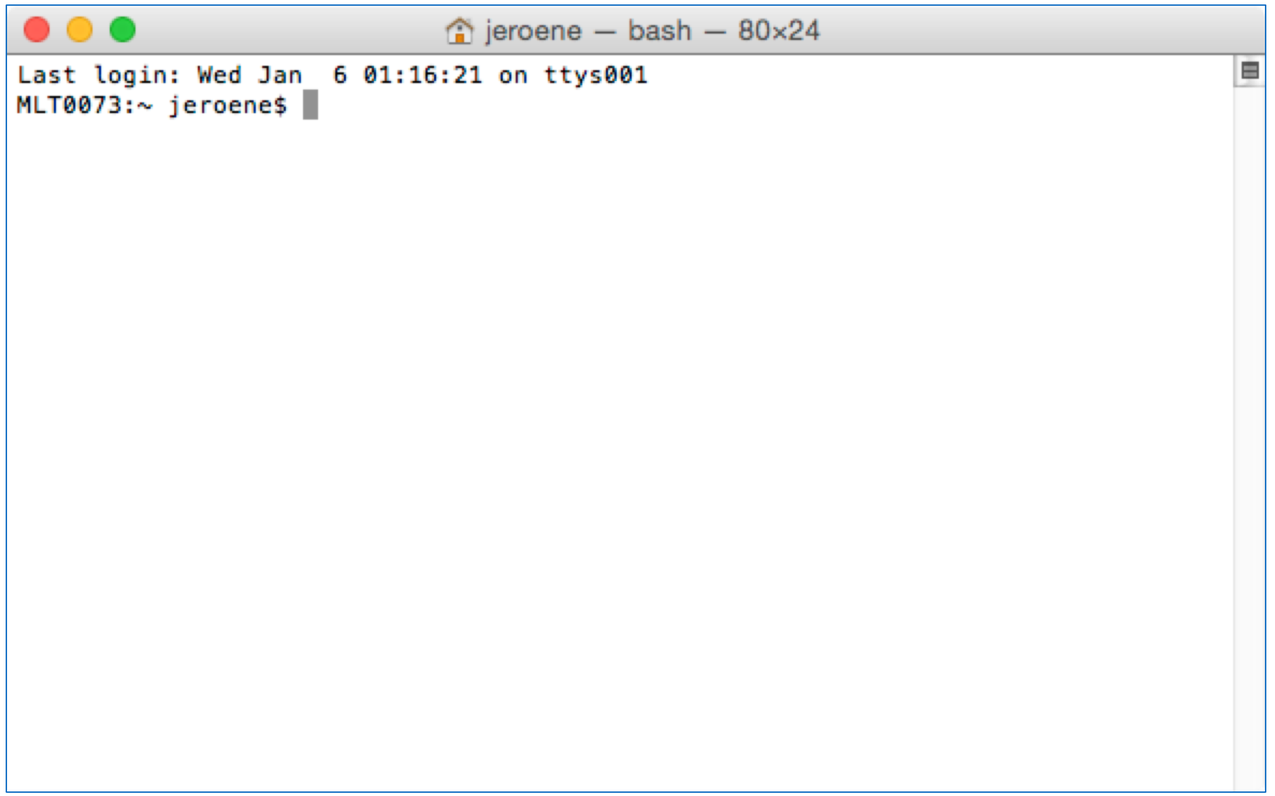
- *Lisa & Cartesius*
- username/password
- key pair

- *HPC Cloud*
- key pair (only!)

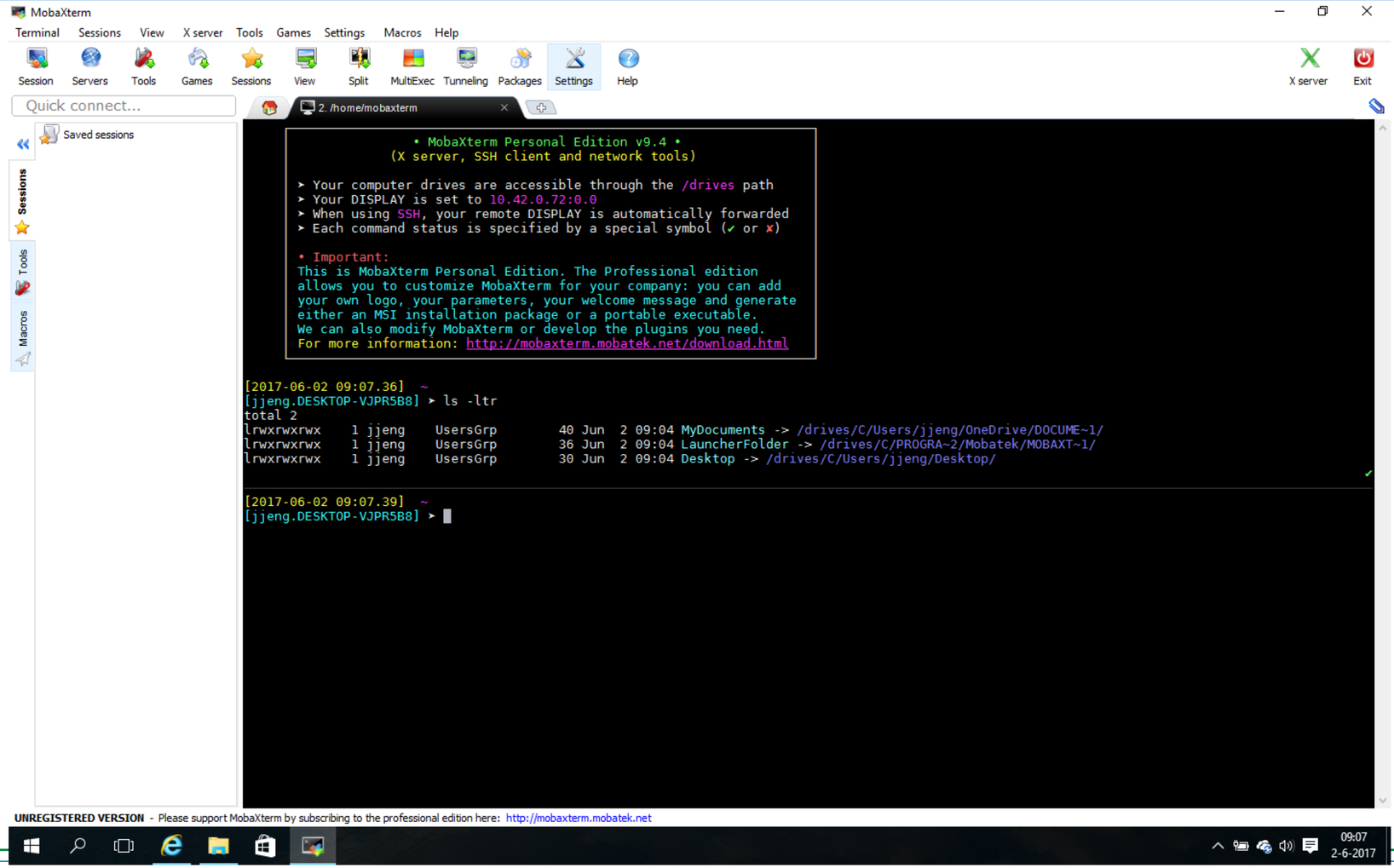
- *Grid*
- username/password (local clusters)
- Grid certificate (for other national and international clusters)

- Get your own demo login: https://goo.gl/forms/zqydoG4cRmaX7DTR2

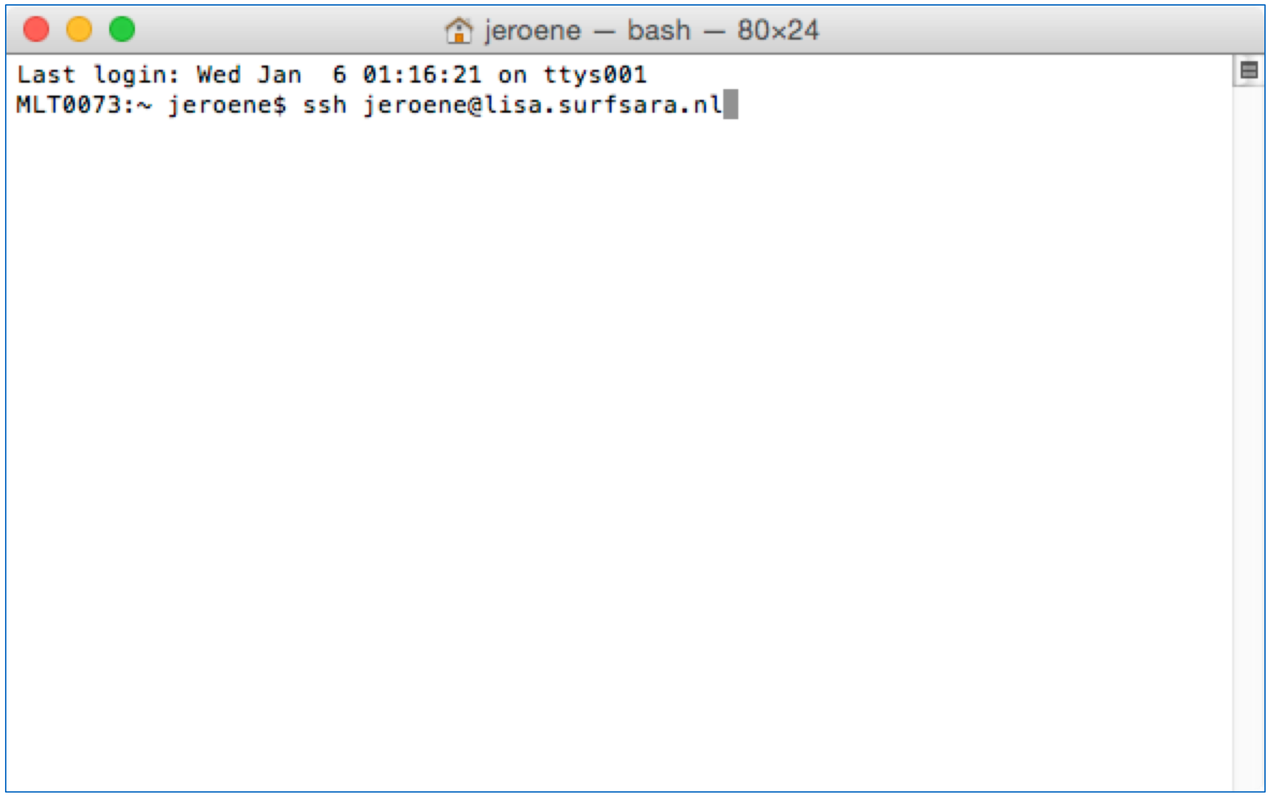
Terminal – OSX



Terminal – Windows (with MobaXterm)

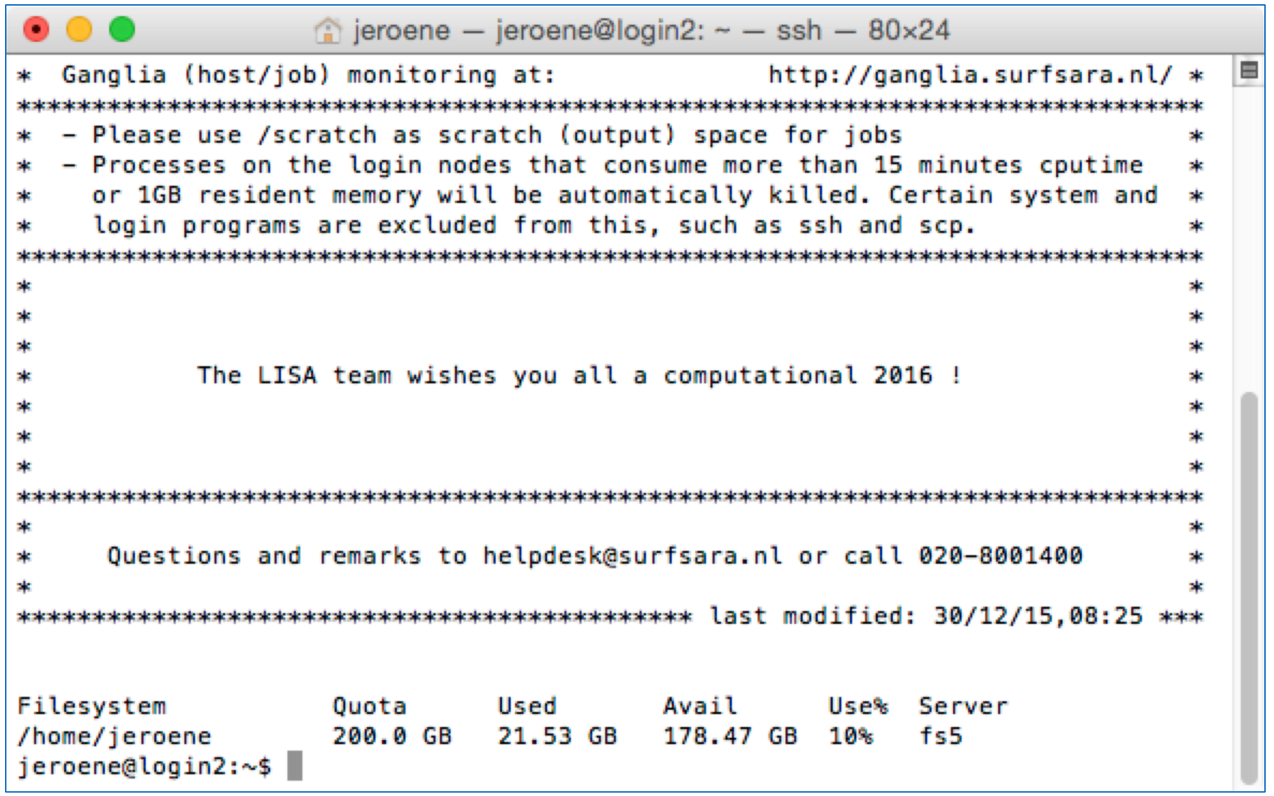


Login to Lisa – with username (2)

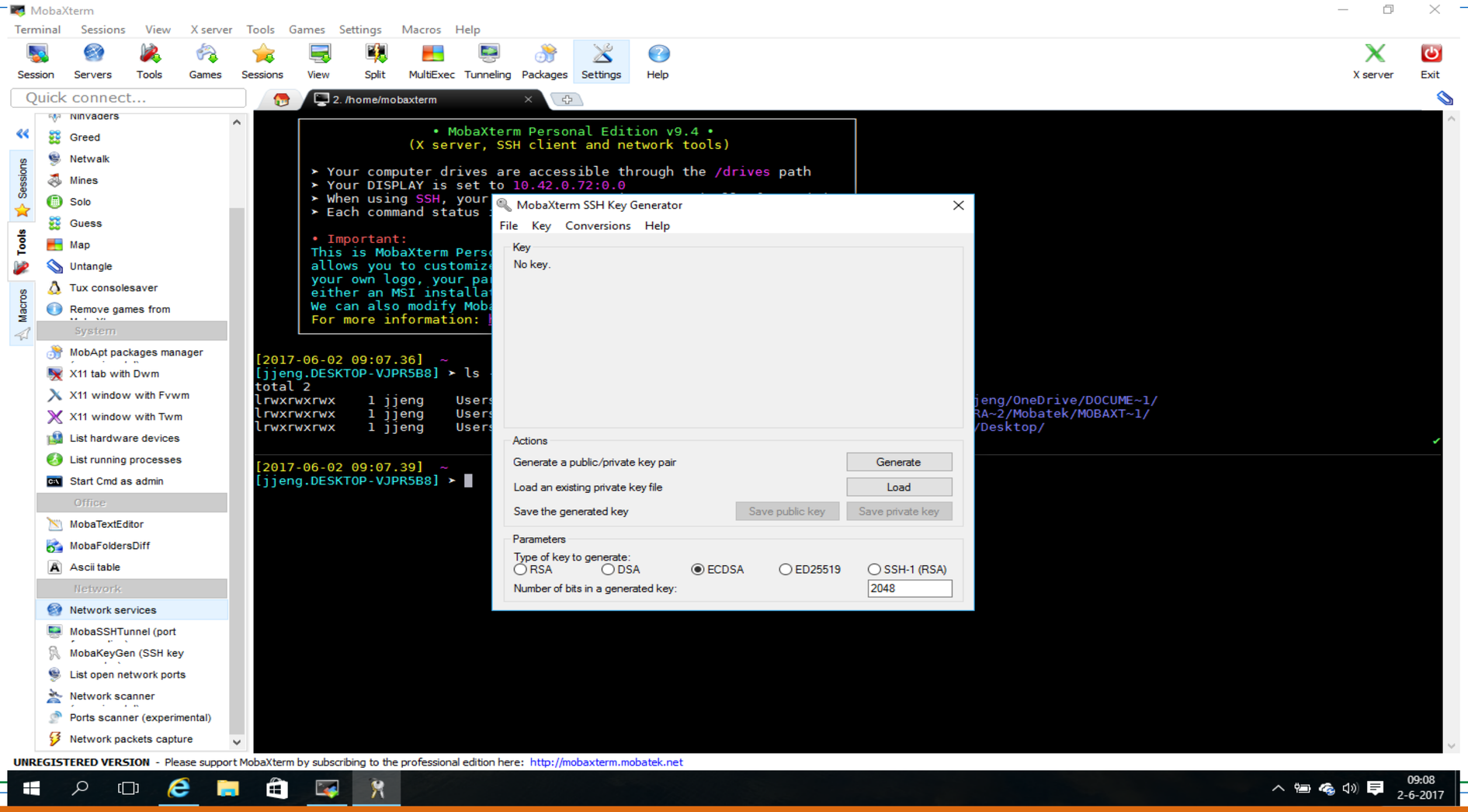


Login to Lisa – with username (3)

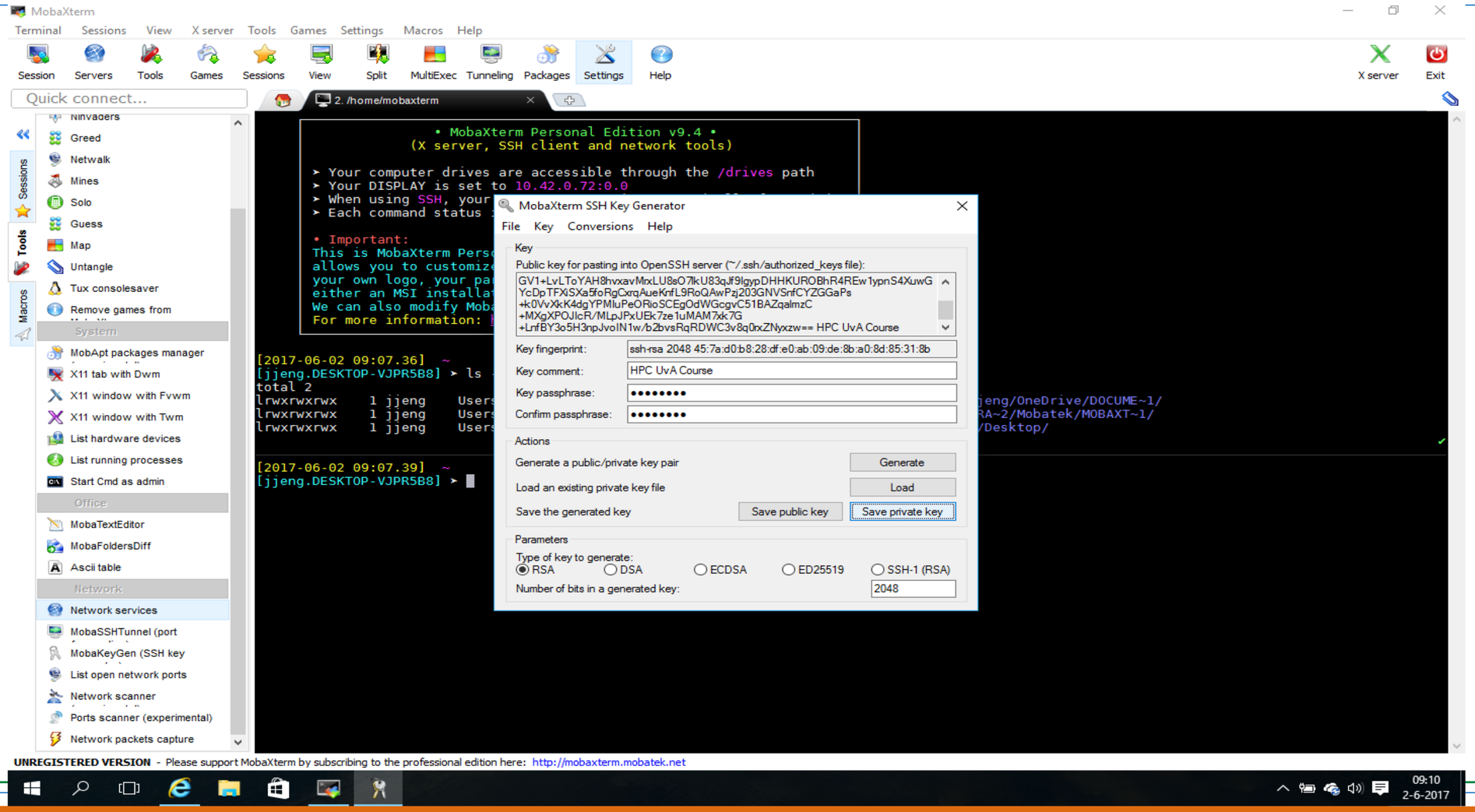
Login to Lisa – with username (4)



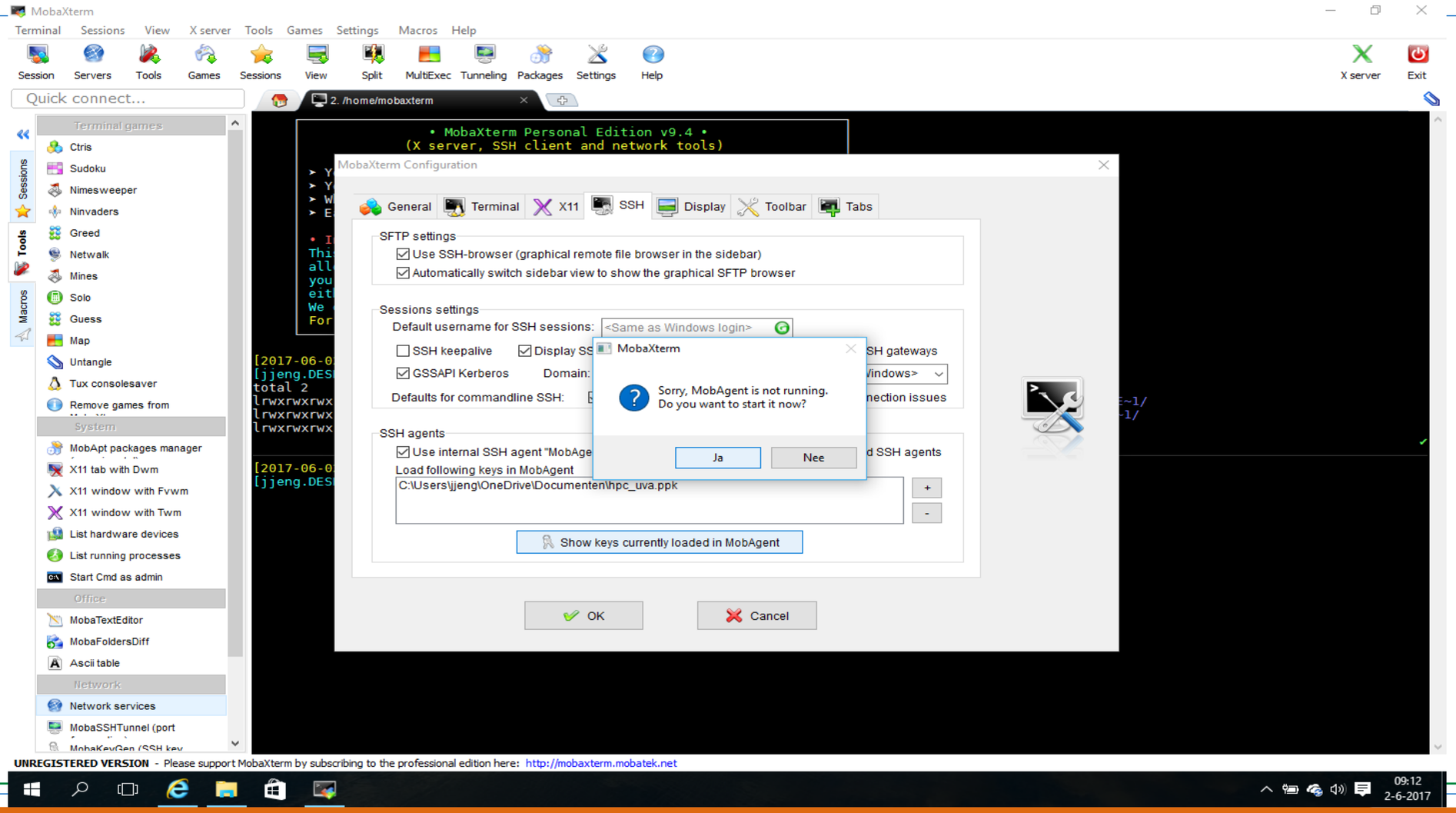
Create key pair – Windows (1)



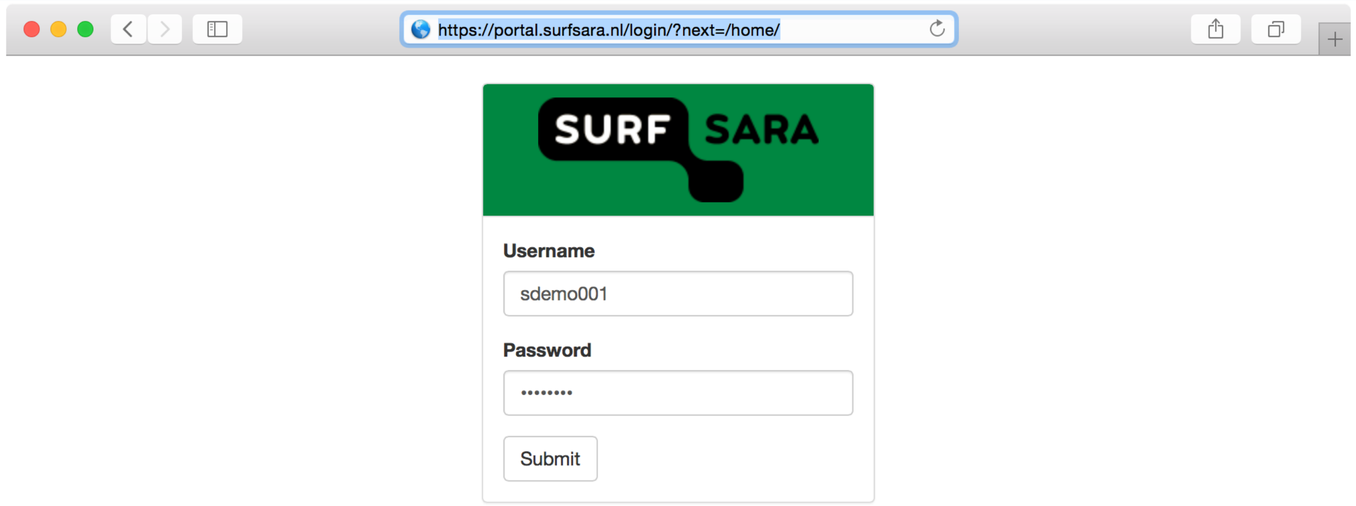
Create key pair – Windows (2)



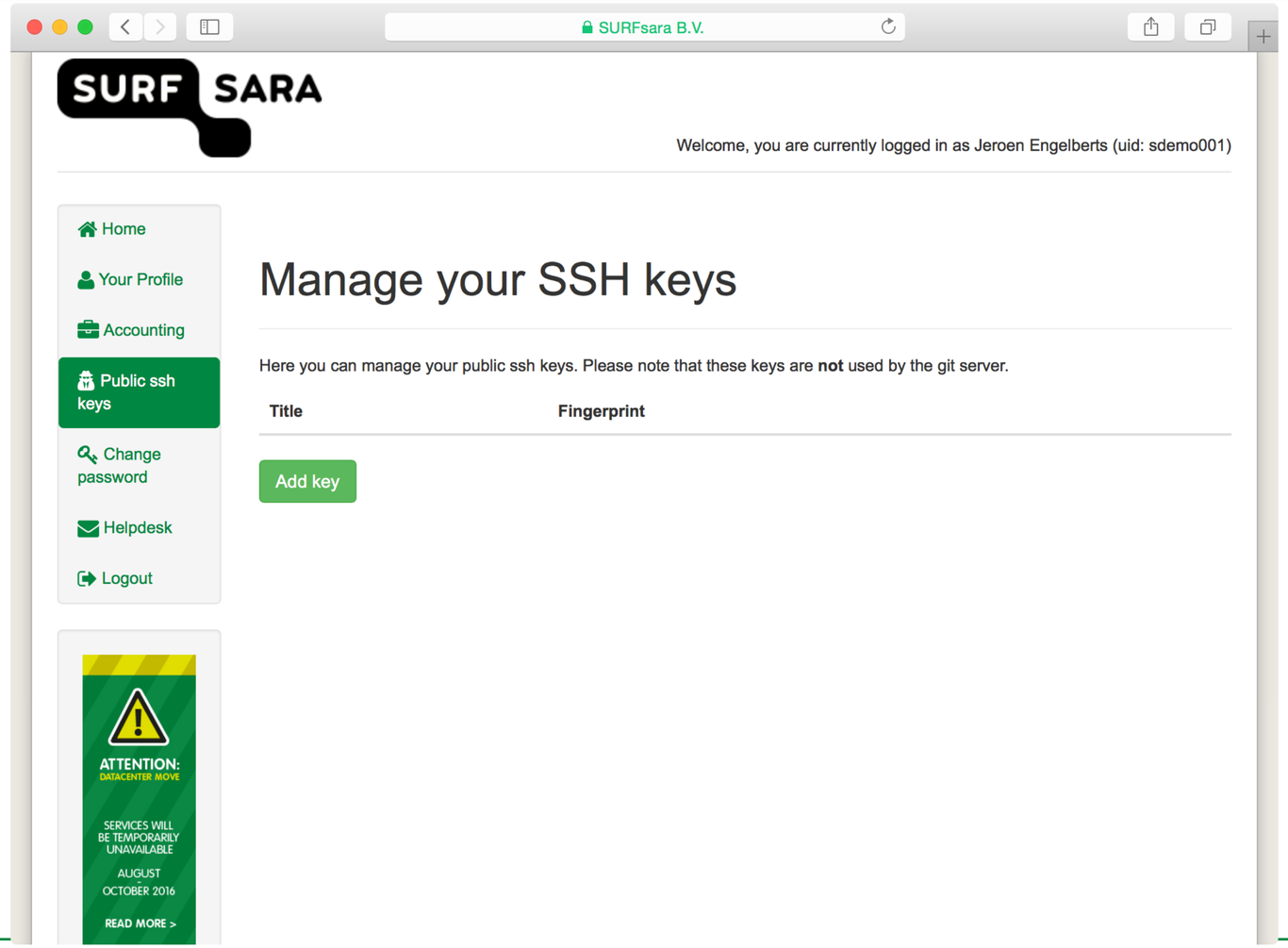
Start MobAgent – Windows



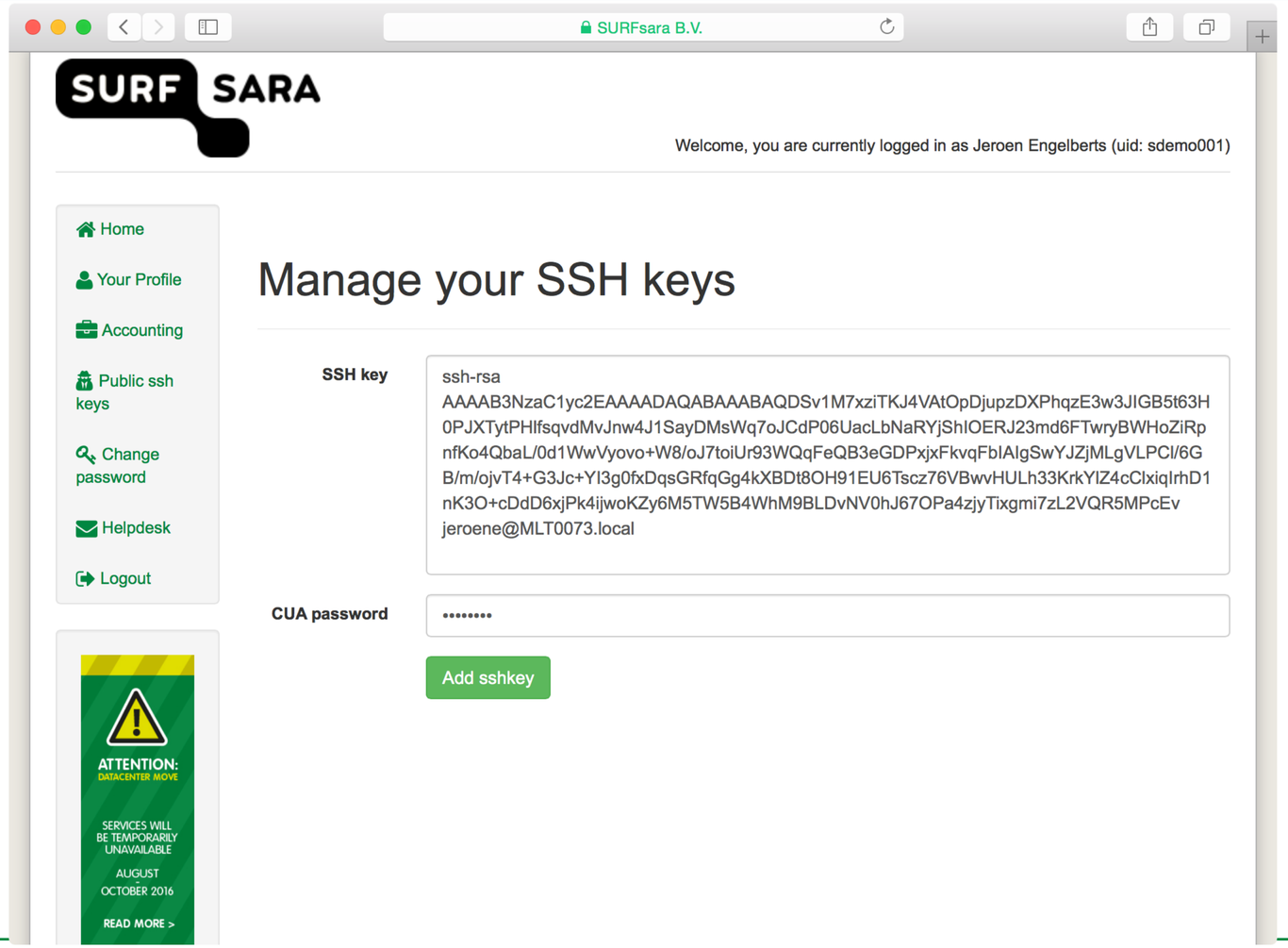
Login to Lisa – with key pair (3)



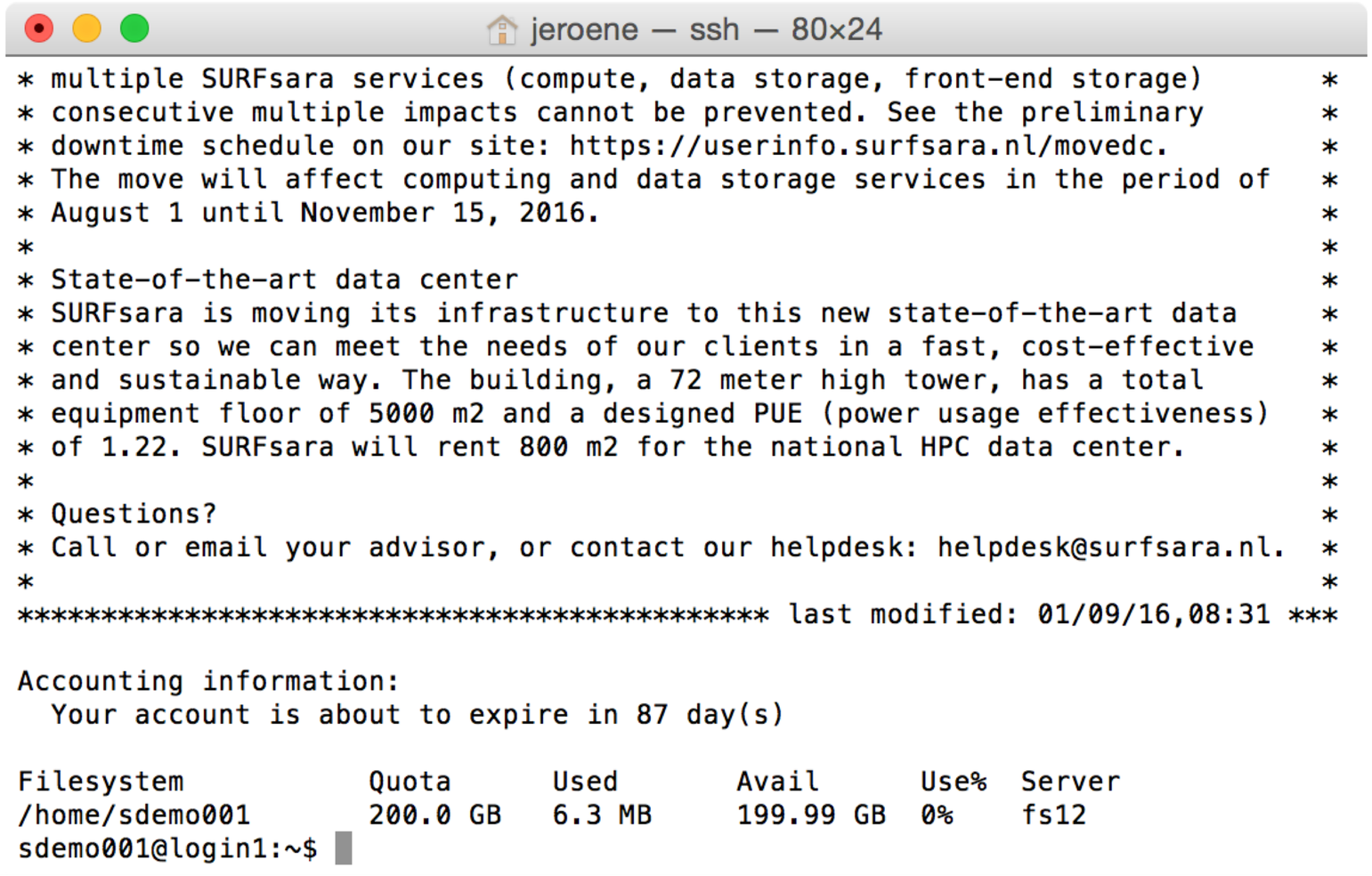
Login to Lisa – with key pair (4)



Login to Lisa – with key pair (5)



Login to Lisa – with key pair (6)



Create key pair and copy to LISA – Linux

ssh-keygen

ssh-copy-id -i ~/.ssh/id_rsa.pub username@lisa.surfsara.nl

-t ed25519

Practical

Connecting to the GPU landing node
PBS/Maui scheduler
First GPU job

Connect to LISA. Get GPU status in a node

Connecting to the gpu landing node
$ ssh login-gpu
$ ssh username@login-gpu.lisa.surfsara.nl

GPU status
$ nvidia-smi

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 384.111                 Driver Version: 384.111                  |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 108...  On   | 00000000:3B:00.0 Off |                  N/A |
|  0%   30C    P8    15W / 250W |      0MiB / 11172MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  GeForce GTX 108...  On   | 00000000:5E:00.0 Off |                  N/A |
|  0%   27C    P8    15W / 250W |      0MiB / 11172MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  GeForce GTX 108...  On   | 00000000:B1:00.0 Off |                  N/A |
|  0%   30C    P8    15W / 250W |      0MiB / 11172MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  GeForce GTX 108...  On   | 00000000:D9:00.0 Off |                  N/A |
|  0%   27C    P8    16W / 250W |      0MiB / 11172MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

LISA batch environment

```
$ qstat -q

$ qstat -u $(whoami)
```

http://physics.princeton.edu/it/faq/pbs-maui-cmds.html

Simple job script

```
#PBS -S /bin/bash
#PBS -lwalltime=00:05:00
#PBS -lnodes=1:ppn=12
#PBS -lmem=250G

cat /proc/cpuinfo | grep -e processor -e 'model name'

nvidia-smi
hostname
```

LISA batch environment

```
qsub -q gpu first_script.sh
qstat -u $(whoami)
```

Practical

LISA software environment
Installing Tensorflow Horovod Keras
Submitting a benchmark job

LISA software environment

```
damian@login-gpu1:~$ module av NCCL
------------------------------------------------------------ /hpc/sw/modules/modulefiles/libraries ------------------------------------------------------------
NCCL/2.0.5-CUDA-9.0.176
damian@login-gpu1:~$ module show NCCL
-------------------------------------------------------------------
/hpc/sw/modules/modulefiles/libraries/NCCL/2.0.5-CUDA-9.0.176:

module load     GCC/6.4.0-2.28
module load     CUDA/9.0.176
prepend-path    LD_LIBRARY_PATH /hpc/sw/NCCL/2.0.5/lib
prepend-path    LD_RUN_PATH     /hpc/sw/NCCL/2.0.5/lib
prepend-path    C_INCLUDE_PATH  /hpc/sw/NCCL/2.0.5/include
prepend-path    INCLUDE_PATH    /hpc/sw/NCCL/2.0.5/include
prepend-path    SARA_INCLUDE_PATH       /hpc/sw/NCCL/2.0.5/include
prepend-path    SARA_LIBRARY_PATH       /hpc/sw/NCCL/2.0.5/lib
-------------------------------------------------------------------
damian@login-gpu1:~$ module load NCCL
damian@login-gpu1:~$ module list
Currently Loaded Modulefiles:
 1) libgfortran/32/1(default)   9) eb/3.6.1(default)              17) gompi/2017b                                  25) Tcl/8.6.7-GCCcore-6.4.0
 2) stdenv/1.3(default)        10) GCCcore/6.4.0                   18) FFTW/3.3.6-gompi-2017b                       26) SQLite/3.20.1-GCCcore-6.4.0
 3) licenses/1.0(default)      11) binutils/2.28-GCCcore-6.4.0     19) ScaLAPACK/2.0.2-gompi-2017b-OpenBLAS-0.2.20  27) GMP/6.1.2-GCCcore-6.4.0
 4) oldwheezy/1.0(default)     12) GCC/6.4.0-2.28                  20) foss/2017b                                   28) libffi/3.2.1-GCCcore-6.4.0
 5) moab/default               13) numactl/2.0.11-GCCcore-6.4.0    21) bzip2/1.0.6-GCCcore-6.4.0                    29) Python/2.7.14-foss-2017b
 6) surfsara/1.1(default)      14) hwloc/1.11.7-GCCcore-6.4.0      22) zlib/1.2.11-GCCcore-6.4.0                    30) CUDA/9.0.176
 7) EasyBuild/3.6.1            15) OpenMPI/2.1.1-GCC-6.4.0-2.28    23) ncurses/6.0-GCCcore-6.4.0                    31) cuDNN/7.0.5-CUDA-9.0.176
 8) compilerwrappers           16) OpenBLAS/0.2.20-GCC-6.4.0-2.28  24) libreadline/7.0-GCCcore-6.4.0                32) NCCL/2.0.5-CUDA-9.0.176
```

LISA software environment

(one time)
$ echo "module load eb" >> ~/.bash_profile

$ source ~/.bash_profile
Or logout/login

Modules needed for a GPU run:
$ module load Python/2.7.14-foss-2017b cuDNN/7.0.5-CUDA-9.0.176 OpenMPI/2.1.1-GCC-6.4.0-2.28 NCCL

$ export LD_LIBRARY_PATH=/hpc/sw/NCCL/2.0.5/lib:/hpc/eb/Debian9/cuDNN/7.0.5-CUDA-9.0.176/lib64:/hpc/eb/Debian9/CUDA/9.0.176/lib64:$LD_LIBRARY_PATH

Installing Tensorflow / Horovod / Keras

Tensorflow GPU
$ pip install tensorflow-gpu --user --upgrade --force-reinstall --no-cache

Horovod installation https://github.com/uber/horovod/blob/master/docs/gpus.md
$ CC=mpicc CXX=mpicxx HOROVOD_NCCL_HOME=/hpc/sw/NCCL/2.0.5 HOROVOD_GPU_ALLREDUCE=NCCL pip install --no-cache-dir --user horovod

Sanity check:
$ echo "localhost slots=4" > machine_file
$ mpirun -np 4 -hostfile machine_file python -c "import horovod.tensorflow as hvd; hvd.init(); print(hvd.rank())”

Keras installation
$ pip install keras --user --no-cache

Tensorflow performance

- The execution of an individual op (for some op types) can be parallelized on a pool of intra_op_parallelism_threads.

- Nodes that perform blocking operations are enqueued on a pool of inter_op_parallelism_threads available in each process.

For LISA GPU: --num_intra_threads 4 --num_inter_threads 3

Full job script

```
#!/bin/sh
#PBS -lwalltime=24:00:00
#PBS -lnodes=2:ppn=12
#PBS -lmem=250GB

module load eb
module load Python/2.7.14-foss-2017b
module load cuDNN/7.0.5-CUDA-9.0.176
module load OpenMPI/2.1.1-GCC-6.4.0-2.28
module load NCCL

export LD_LIBRARY_PATH=/hpc/sw/NCCL/2.0.5/lib:/hpc/eb/Debian9/cuDNN/7.0.5-CUDA-
9.0.176/lib64:/hpc/eb/Debian9/CUDA/9.0.176/lib64:$LD_LIBRARY_PATH

rm machine_file
cat $PBS_NODEFILE > machine_file
sed 's/$/ slots=4/' machine_file > machinefile
uniq machinefile | cat > machine_file
rm machinefile

OMP_NUM_THREADS=12 HOROVOD_FUSION_THRESHOLD=33554432 mpirun -np 8 --map-by ppr:4:node -hostfile machine_file -x
NCCL_P2P_DISABLE=0 --rank-by core --display-map -x HOROVOD_FUSION_THRESHOLD -x OMP_NUM_THREADS python
~/horovod/examples/tensorflow_mnist.py --variable_update horovod --num_intra_threads 4 --num_inter_threads 3 --horovod_device cpu
```

Submit the job

qsub –q gpu full_job_script.sh

And monitoring can be done by
qstat –u $(whoami)
qstat –f jobid
ssh rXXnX
nvidia-smi
top

Horovod code explained

```
hvd.init() #World initialization

mnist = learn.datasets.mnist.read_data_sets('MNIST-data-%d' % hvd.rank(), one_hot=True) #Each worker
gets its own shard

D_solver = hvd.DistributedOptimizer(tf.train.AdamOptimizer(0.001 * hvd.size())).minimize(D_loss,
var_list=theta_D, global_step=global_step)
G_solver = hvd.DistributedOptimizer(tf.train.AdamOptimizer(0.001 * hvd.size())).minimize(G_loss,
var_list=theta_G, global_step=global_step) #distributed optimizers

config.gpu_options.visible_device_list = str(hvd.local_rank()) #visible devices list

        hvd.BroadcastGlobalVariablesHook(0), #hooks
        tf.train.StopAtStepHook(last_step=20000 // hvd.size()), #since we are doing data parallelism, an epoch
is locally epoch/workers
```

Extra notes about Keras performance

```
import keras.backend as K

    K.set_image_dim_ordering('tf')

    import horovod.keras as hvd

    config = tf.ConfigProto()

    config.intra_op_parallelism_threads = params.intraop
    config.inter_op_parallelism_threads = params.interop

    os.environ['KMP_BLOCKTIME'] = str(1)
    os.environ['KMP_SETTINGS'] = str(1)
    os.environ['KMP_AFFINITY'] = 'granularity=fine,compact'
    # os.environ['KMP_AFFINITY'] = 'balanced'
    os.environ['OMP_NUM_THREADS'] = str(params.intraop)

    K.set_session(tf.Session(config=config))
```

Extra notes about Horovod performance

One of the unique things about Horovod is its ability to interleave communication and computation coupled with the ability to batch small *allreduce* operations, which results in improved performance. We call this batching feature Tensor Fusion. Tensor Fusion works by attempting to combine all the tensors that are ready to be reduced at given moment of time into one reduction operation.

The fusion buffer size can be tweaked using the HOROVOD_FUSION_THRESHOLD environment variable:
$ HOROVOD_FUSION_THRESHOLD=33554432 mpirun -np 4 -x HOROVOD_FUSION_THRESHOLD python train.py

Setting the HOROVOD_FUSION_THRESHOLD environment variable to zero disables Tensor Fusion:
$ HOROVOD_FUSION_THRESHOLD=0 mpirun -np 4 -x HOROVOD_FUSION_THRESHOLD python train.py

You can tweak time between cycles (defined in milliseconds) using the HOROVOD_CYCLE_TIME environment variable:
$ HOROVOD_CYCLE_TIME=3.5 mpirun -np 4 -x HOROVOD_FUSION_THRESHOLD python train.py

Questions?