POLITECNICO
MILANO 1863

Performance Prediction of GPU-based Deep Learning Applications

Danilo Ardagna

danilo.ardagna@polimi.it

ATMOSPHERE
Adaptive, Trustworthy, Manageable, Orchestrated, Secure Privacy-assuring Hybrid, Ecosystem for REsilient Cloud Computing

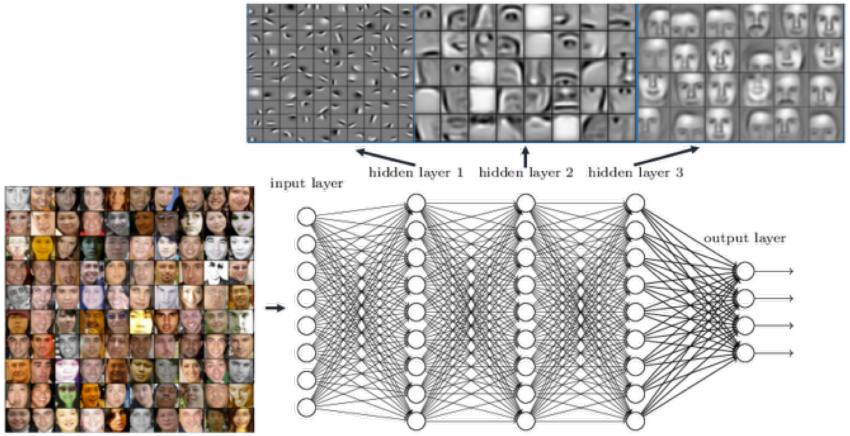
13th RES Users Conference – Zaragoza, 19th September 2019

Outline

- Motivations
- Target scenario and goals
- Performance models and Scheduling problem
- Experimental Results
- Conclusions & Future Works

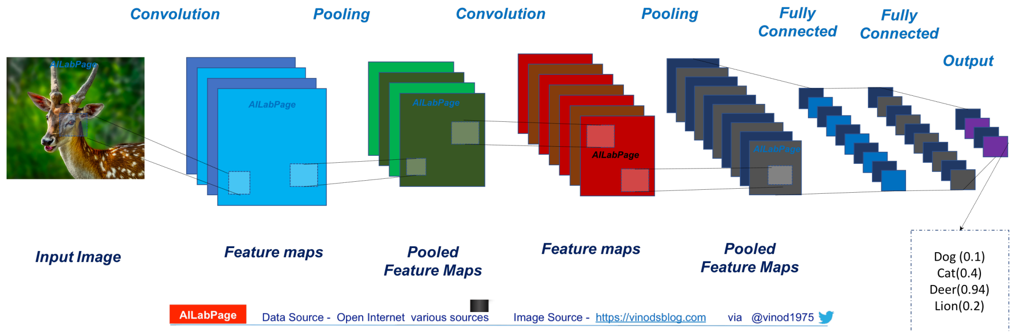
Motivations

- Deep learning is widely used in commonplace activities
- Model learning greatly benefits from GPUs
  - GPUs are **5 to 40x** faster than CPUs
- High-end systems cost up to **400k€** (single node)
- Cloud provider offer **pay-as-you-go** GPU-powered VMs
  - GPU based VMs time unit cost is **5-8x** higher than high-end CPU-only VMs



Motivations

- Deep learning is widely used in commonplace activities



- Model learning greatly benefits from GPUs
  - GPUs are **5 to 40x** faster than CPUs

- High-end systems cost up to **400k€** (single node)

- Cloud provider offer **pay-as-you-go** GPU-powered VMs
  - GPU based VMs time unit cost is **5-8x** higher than high-end CPU-only VMs

Target Scenario

- Deep Learning applications: heterogeneous and irregular computational patterns

+

- Cloud computing: offers flexibility, dynamically adjusting resources as needed

Which configuration should we choose to avoid under and overestimating resources?

Our Goals

How can we predict the execution time of Deep Learning training jobs running on a target cloud configuration?

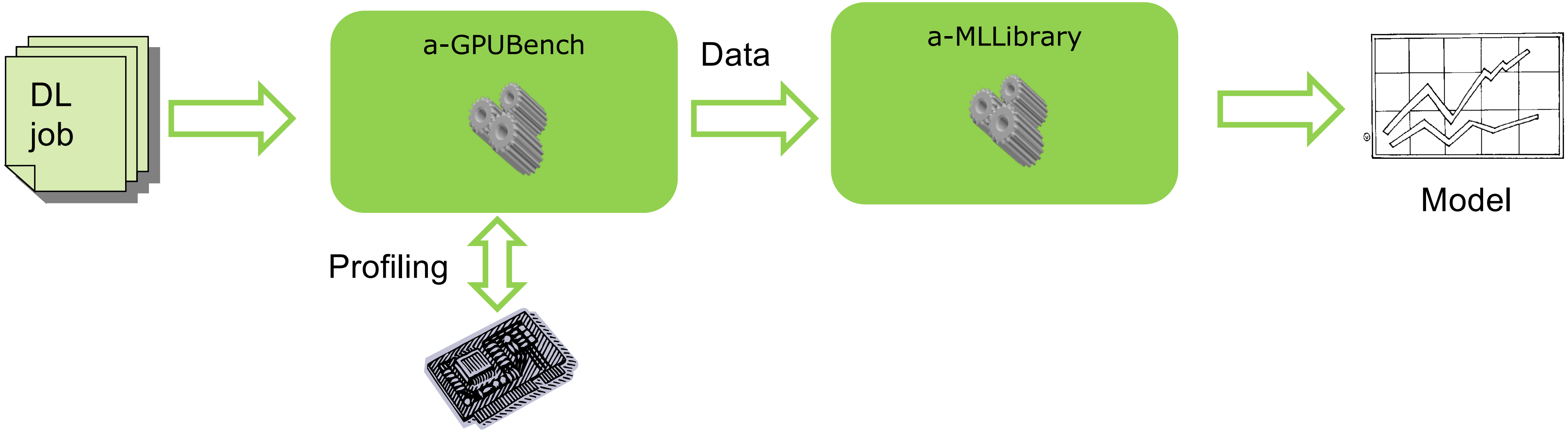
Our Goals

How can we predict the execution time of Deep Learning training jobs running on a target cloud configuration?

ML models to predict execution time given GPU type and number

Online joint capacity planning of on-demand VMs and DL job scheduling

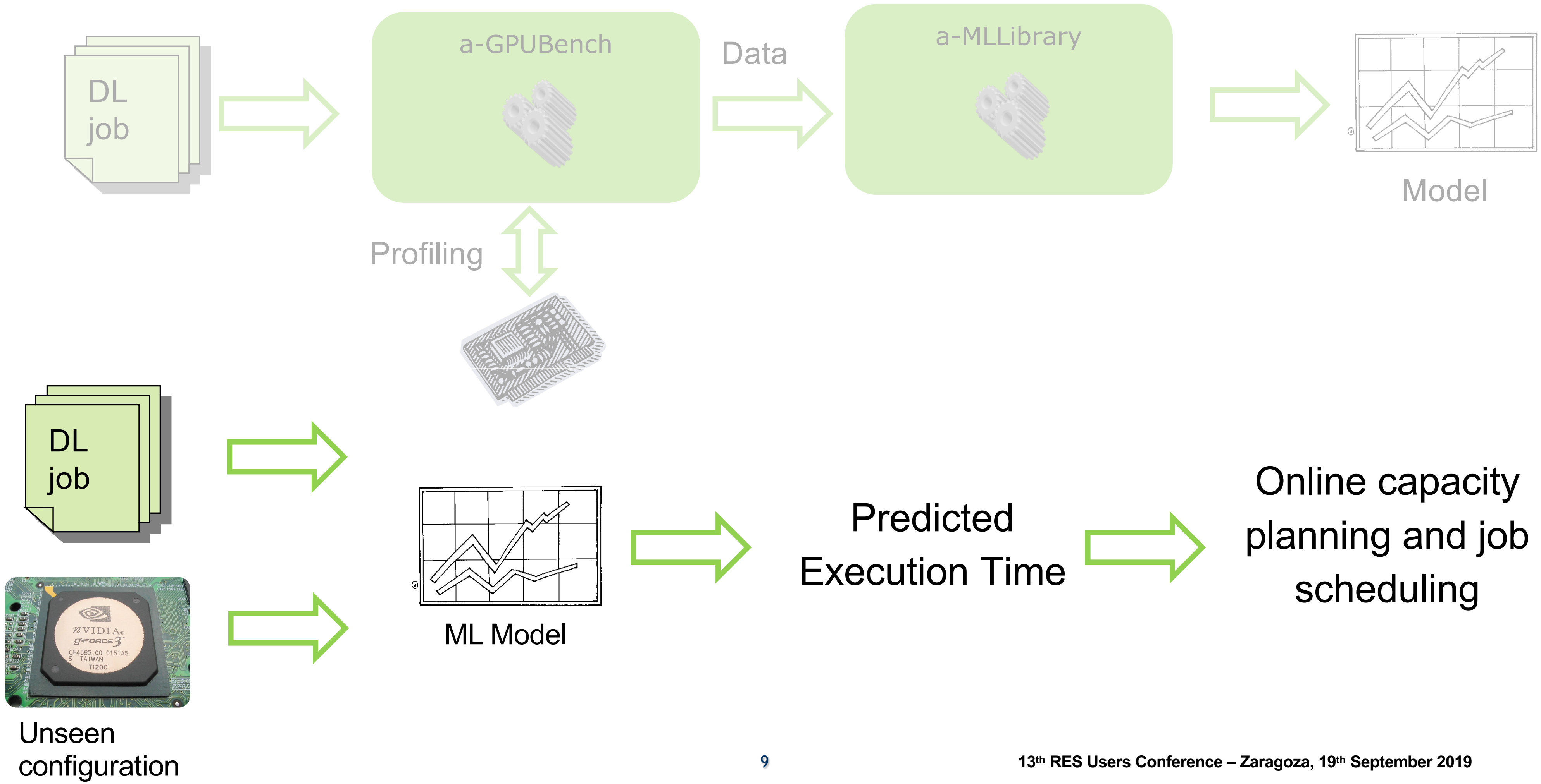
Our Approach: Machine learning



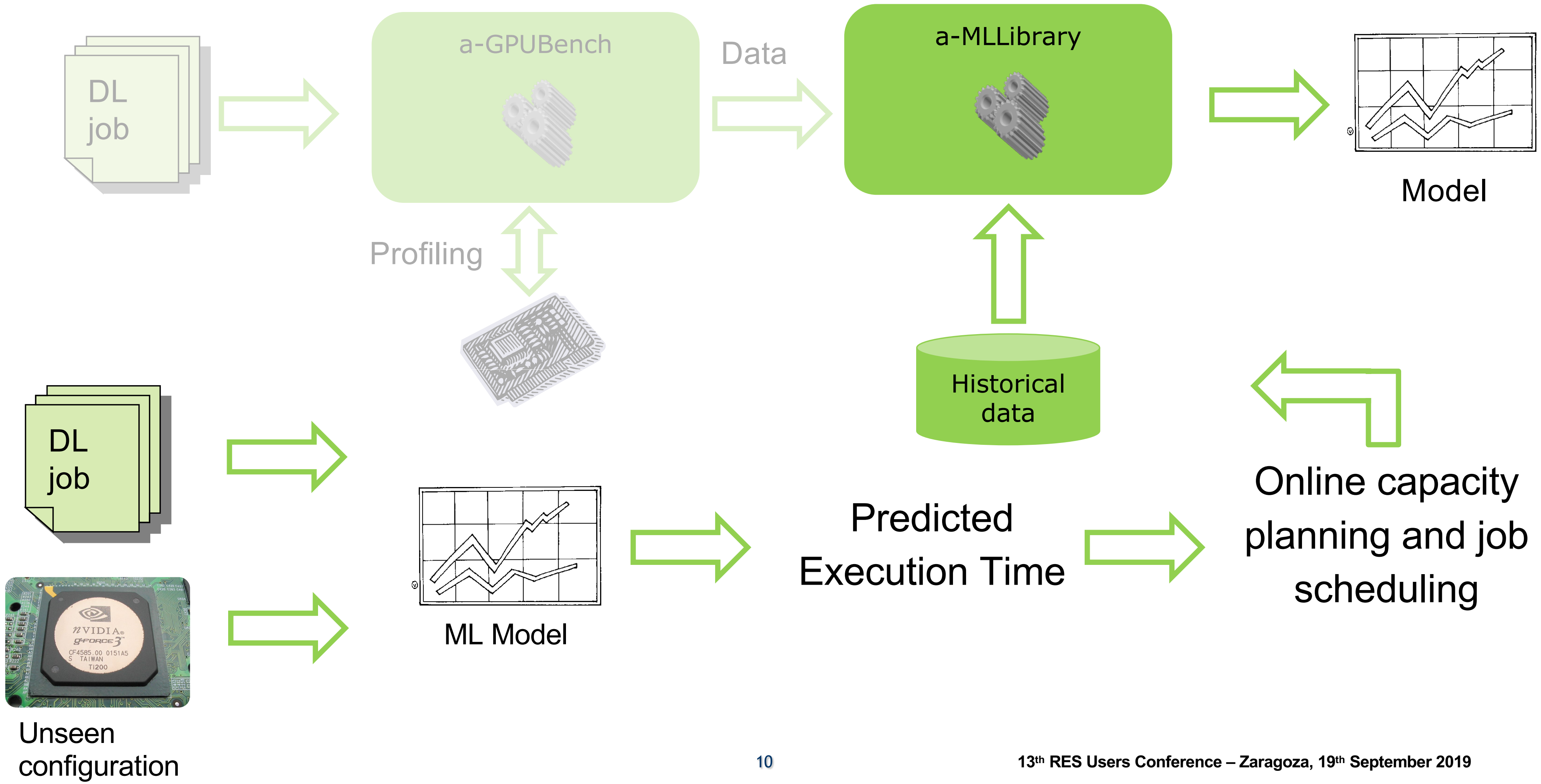
https://github.com/eubr-atmosphere/a-GPUBench

https://github.com/eubr-atmosphere/a-MLLibrary

Our Approach: Machine learning



Our Approach: Machine learning



Our Goals

How can we predict the execution time of Deep Learning training jobs running on a target cloud configuration?

ML models to predict execution time given GPU type and number

Online joint capacity planning of on-demand VMs and DL job scheduling

Our Goals

How can we predict the execution time of Deep Learning training jobs running on a target cloud configuration?

End-to-End Models

Per-Layer Models

Online joint capacity planning of on-demand VMs and DL job scheduling

End-to-End & Per-Layer models

- Linear regression
- Features set:
  - #iterations, batch size, GFlops/s, #GPUs, #threads, disk delay
- Features augmentation: inverse and cross-over terms
- Features selection: Draper and Smith approach

E. Gianniti, L. Zhang, D. Ardagna. **Performance Prediction of GPU-based Deep Learning Application**. Closer 2019 Proceedings. 279-286. Crete, Greece.

- Linear regression
- Feature: operation complexity
  - Derived from architecture and hyper-parameters

$$t_l = \beta_{0l} + \beta_{1l} c_l + \varepsilon_l$$

$$\hat{t}^{\mathrm{CNN}} = I \sum_{l \in L} \hat{t}_l$$

- Benefits: prediction on unseen applications

Layer	Forward	Backward
Conv	$H_f W_f C_{in} C_{out}$	$(2 H_f W_f C_{in} + 1) C_{out}$
FC	$H_{in} W_{in} C_{in} C_{out}$	$2 H_{in} W_{in} C_{in} C_{out}$
Loss	$4 C_{out} - 1$	$C_{out} + 1$
Norm	$5 C_{out} + C_n - 2$	$8 C_{out} + C_n - 1$
Pool	$H_f W_f C_{out}$	$(H_f W_f + 1) C_{out}$
ReLU	$3 C_{out}$	$4 C_{out}$

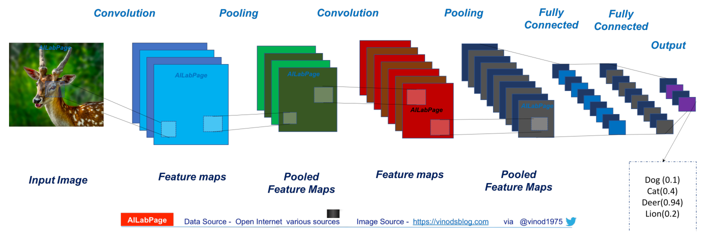
Experimental Settings

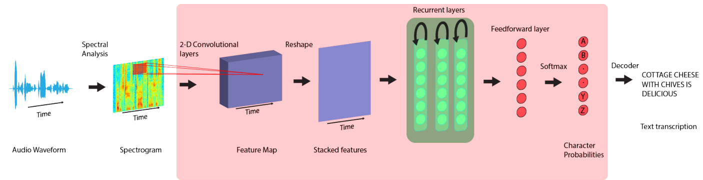
Several well-known Neural networks
- AlexNet
- GoogLeNet
- VGG-16
- ResNet
- Mozilla DeepSpeech

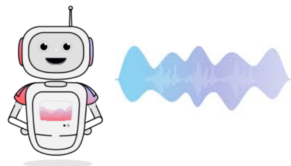
Tensorflow 1.8.0, Pytorch 0.3.1, Caffe 1.0

Training data sets:
- ImageNet dataset (ILSVRC12)
- Common Voice open source corpus

HW configurations:
- Microsoft Azure NC (K80), NV(M60), in-house servers (P600, Geforce GTX 1080Ti)







End-to-End Models Results

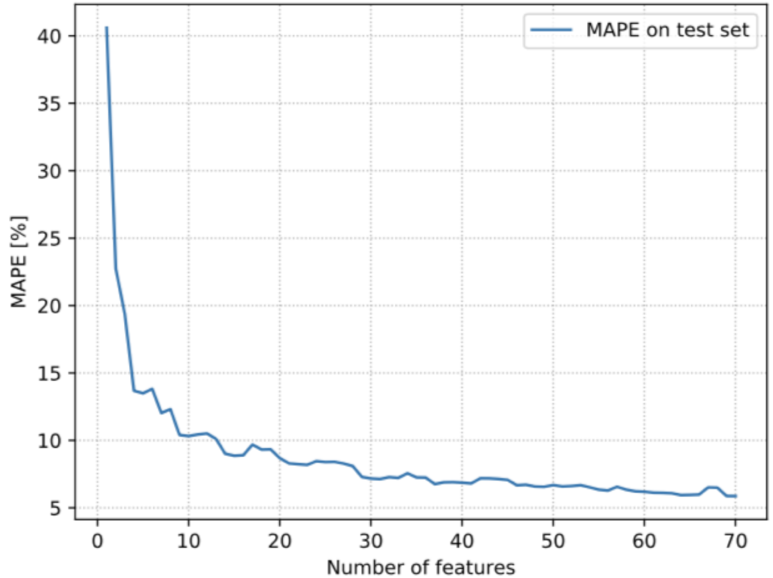
- Performance prediction goals:
  - Extrapolation on batch size
  - Exploitation of new hardware
  - Training of new versions of a CNN

GPUs number Extr.

Network	Framework	GPU Type K80	M60	GTX 1080Ti
AlexNet	PyTorch	7.21	12.18	4.98
	TensorFlow	24.75	17.27	8.77
ResNet-50	PyTorch	5.11	9.04	11.76
	TensorFlow	24.58	18.29	6.54
VGG-19	PyTorch	12.20	15.98	24.13
	TensorFlow	8.84	13.52	13.65

Extr. Inner Modules number

Network	Framework	Max N. IMs	GPU Type M60 1	2	4
ResNet	PyTorch	4	23.51	27.95	17.40
ResNet	PyTorch	5	24.85	25.11	16.75
ResNet	PyTorch	6	$\boxed{26.76}$	20.40	16.63
ResNet	PyTorch	8	17.06	7.93	15.99



Batch Size Extr.

Network	Framework	GPU Type P600 1	P600 2	K80 1	K80 2	K80 3	K80 4	M60 1	M60 2	M60 3	M60 4	GTX 1080Ti 1	GTX 1080Ti 2	GTX 1080Ti 4	GTX 1080Ti 8
AlexNet	PyTorch	11.12	7.85	1.74	3.33	1.81	0.66	6.19	3.49	6.58	0.75	0.43	1.62	1.15	4.16
	TensorFlow	9.83	10.04	2.30	2.61	4.28	2.82	7.19	6.36	6.91	6.96	4.06	5.36	1.14	1.12
ResNet-50	PyTorch	10.64	11.97	0.76	7.83	3.09	4.53	3.60	20.04	9.58	4.64	12.62	11.93	20.63	4.29
	TensorFlow	2.37	14.35	10.25	1.27	1.84	6.83	2.08	2.79	3.07	21.49	0.68	6.44	1.43	12.06
VGG-19	PyTorch	-	-	13.88	21.71	27.63	9.65	10.74	18.54	13.81	7.68	24.98	17.40	2.93	14.06
	TensorFlow	-	-	18.20	0.92	1.16	10.58	7.34	5.06	2.74	6.92	22.88	6.37	24.12	23.56

Comput. Power Extr.

Network	Framework	MAPE
AlexNet	PyTorch	8.28
	TensorFlow	5.08
ResNet-50	PyTorch	18.09
	TensorFlow	10.10

End-to-End Models Results

Batch Size Extr.

Network	Framework	GPU Type and Number					
		P600		K80			
		1	2	1	2	3	4
DeepSpeech	Tensorflow	6.69	5.25	22.46	15.82	3.14	4.41

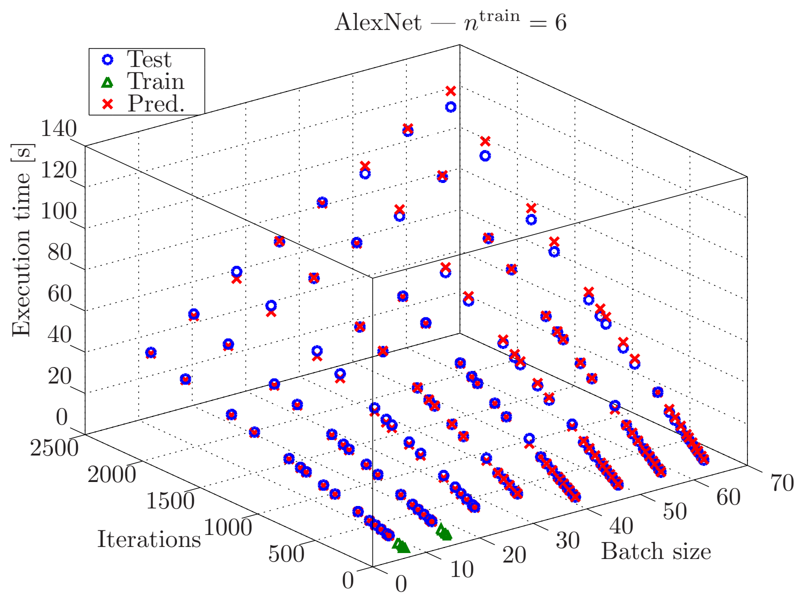
Network	Framework	GPU Type and Number							
		M60				GTX 1080Ti			
		1	2	3	4	1	2	4	8
DeepSpeech	TensorFlow	19.08	12.98	7.09	7.11	5.80	9.63	13.87	5.93

GPUs number Extr.

Network	Framework	GPU Type		
		K80	M60	GTX 1080Ti
DeepSpeech	TensorFlow	11.86	17.49	20.97

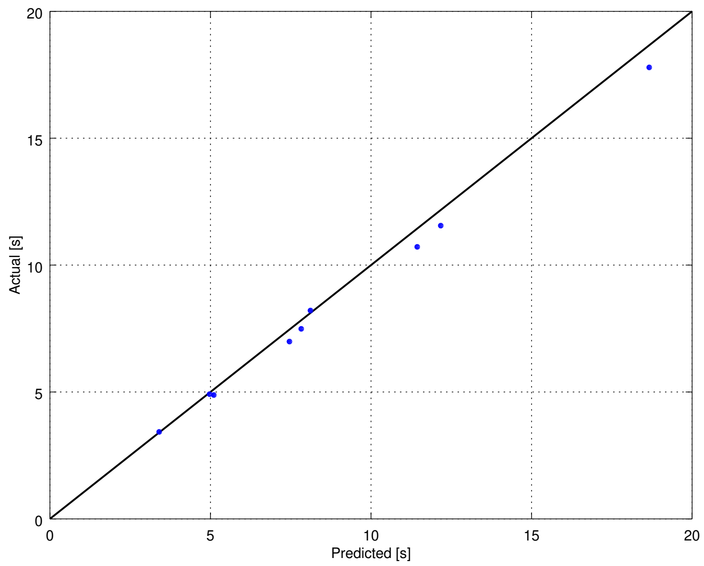
Comput. Power Extr.

Network	Framework	Used Features	
		All	5
Deepspeech	Tensorflow	11.72	10.14

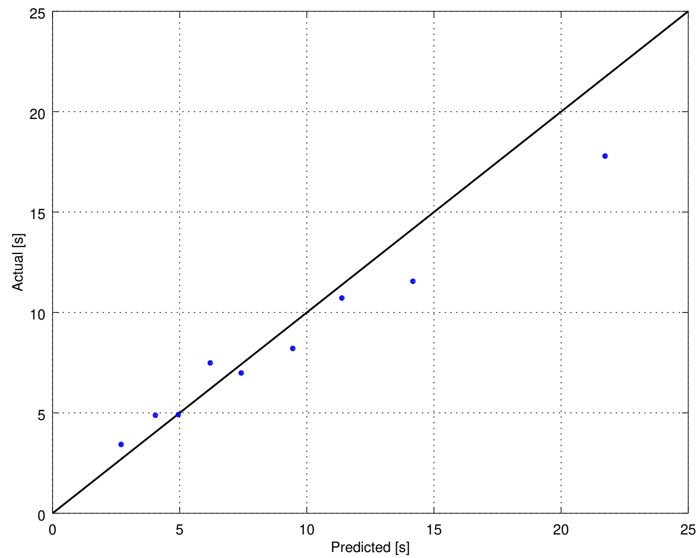


MAPE 3.18%

Per-Layer Models Results



AlexNet model, AlexNet prediction



GoogLeNet model, AlexNet prediction

E. Gianniti, L. Zhang, D. Ardagna. **Performance Prediction of GPU-based Deep Learning Application**. SBAC-PAD 2018 Proceedings. 167-170. Lyon, France.

AlexNet-AlexNet MAPE 4.75%
GoogLeNet-AlexNet MAPE 9.29%

Our Goals

How can we predict the execution time of Deep Learning training jobs running on a target cloud configuration?

Predict execution time given an amount of resources available

Online joint capacity planning of on-demand VMs and DL job scheduling

Reference System and Assumptions

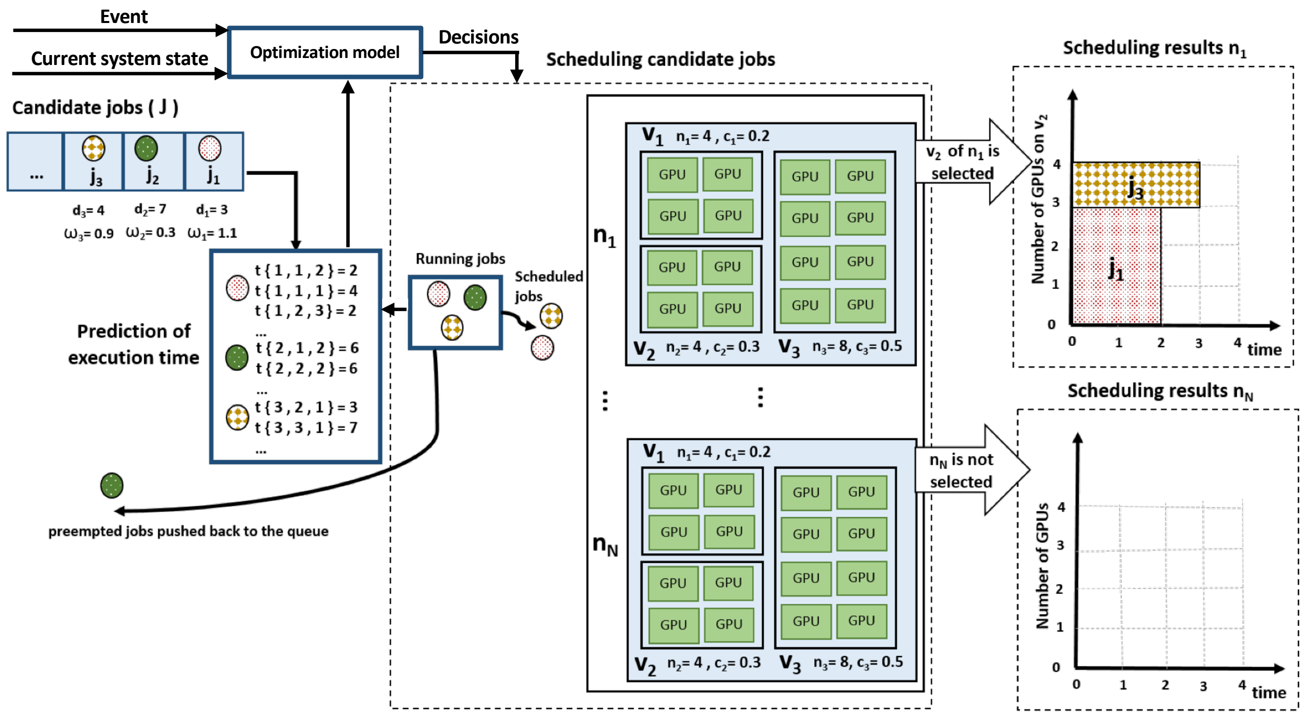
Joint capacity planning of on-demand VMs and DL job scheduling:

1. Decide the number of active nodes
2. Decide the best VM type for each node
3. Partition the GPUs among the jobs
4. Determine an optimized jobs schedule

Online: decisions are taken every time an event occurs

1. A new job is submitted
2. A running job terminates
3. H units of time have elapsed

- Solution Based on MILP
- Several formulations



A. Jahani, M. Lattuada, M. Ciavotta, D. Ardagna, E. Amaldi, L. Zhang. **Optimizing on-demand GPUs in the Cloud for Deep Learning Applications Training**. IEEE ICCCS 2019 Proceedings- To Appear. Rome, Italy.

Experimental campaign

- **Simulation**
  - Ad-hoc event-based **simulator** and a **generator of random instances** implemented in Python
  - Gurobi Optimizer 8.0
  - Jobs inter-arrival times have been generated according to Poisson Distributions with means {30s, 45s}
  - The number of submitted jobs in each instance is set equal to 10 times the number of available nodes
  - Five instances generated for each set of parameters
  - Intel Xeon Silver 4114 server exploiting 12 cores and 32 GB of memory

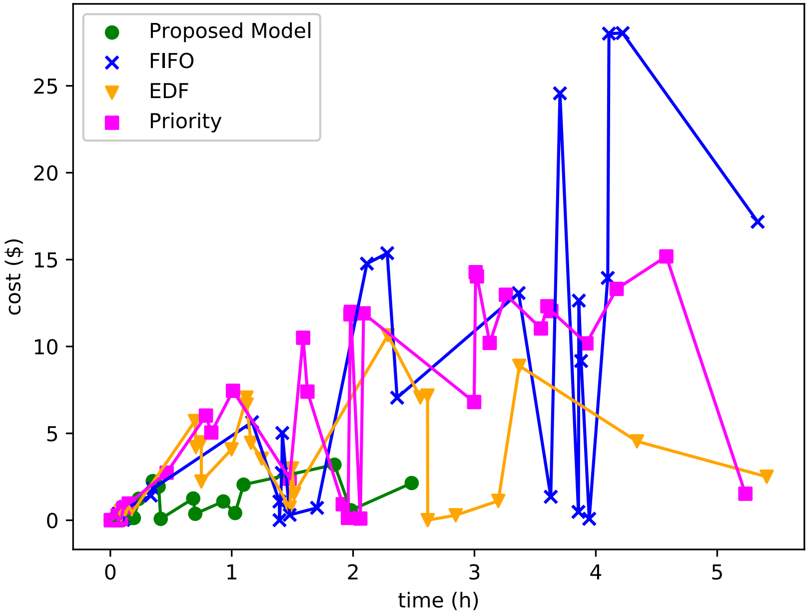
VM type	GPU type	# GPU	Cost($/hour)
Standard NC6	K80	1	0.56
Standard NC12	K80	2	1.13
Standard NC24	K80	4	2.25
Standard NV6	M60	1	0.62
Standard NV12	M60	2	1.24
Standard NV24	M60	4	2.48
In-house server 1	Quadro P600	2	0.11
In-house server 2	GTX 1080Ti	8	1.13
In-house server 3	Quadro P600	8	0.44

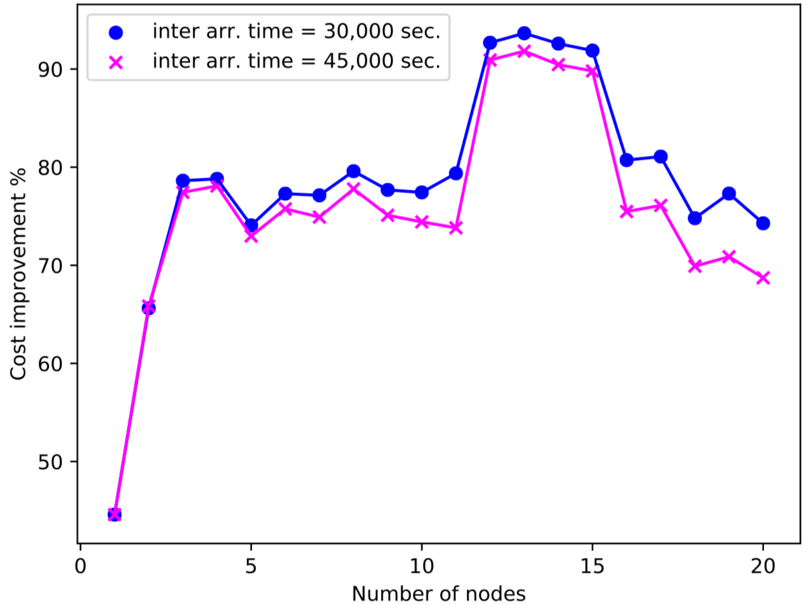
- **Real prototype**
  - Deployed on Microsoft Azure
  - 2 nodes 4 VM types (NC6, NV6, NV12, NV24)
  - Jobs submitted every 300 seconds

Job	Application	N. Inputs	Epochs	Batch Size
JJ1	TensorFlow deepspeech	64	158	4
JJ2	PyTorch vgg19	130000	1	32
JJ3	TensorFlow vgg19	130000	1	32
JJ4	PyTorch alexnet	130000	12	256
JJ5	PyTorch resnet50	130000	3	64
JJ6	TensorFlow resnet50	130000	3	64

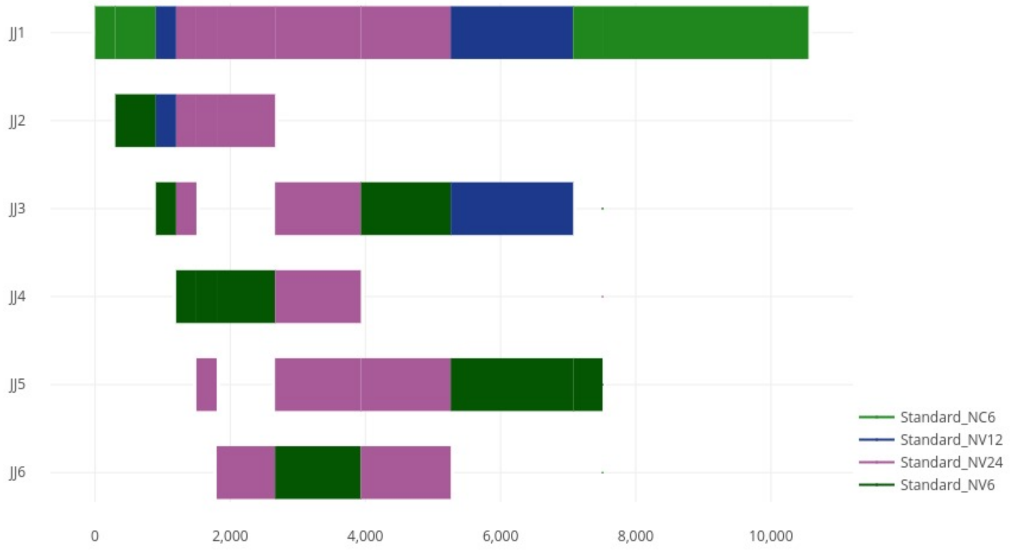
Simulation: comparison against first-principles

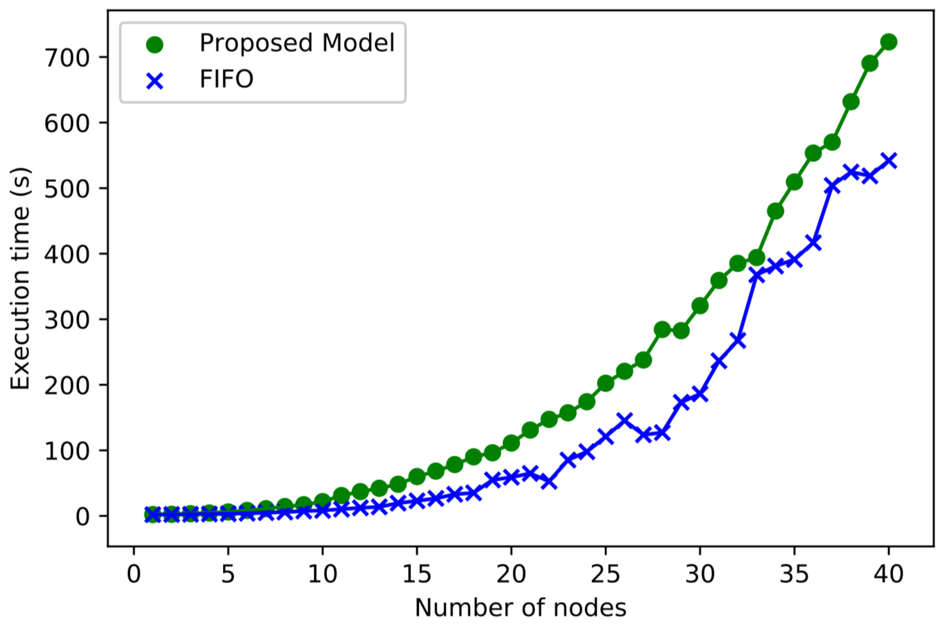
- Comparison against First-in-First-out (**FIFO**), Earliest Deadline First (**EDF**), and Priority Scheduling (**PS**) rules with no preemption and exploiting the model to select the right VM type





Real systems and Scalability





Slot	Predicted cost	Real cost
1	0.09	0.09
2	0.39	0.39
3	0.32	0.32
4	0.53	0.53
5	0.53	0.53
6	1.54	1.88
7	2.25	2.47
8	2.83	2.39
9	3.33	3.68
10	1.28	1.37
11	2.93	3.06
sum	16.02	16.71
Overall Difference		4.12%

Conclusions and Future Work

- Performance models and online capacity planning and scheduling of DL training jobs
- Scheduler scalability
- Disaggregated hardware
- Edge systems and security trade-offs
- Integration of the performance models into the architecture search (e.g., AutoML with performance bounds)

Acknowledgements

Colleagues at Polimi, Università Milano Bicocca,
IBM TJ Watson, and Tabriz University:
Marco Lattuada, Eugenio Gianniti, Li Zhang, Arezoo Jahani,
Federica Filippini, Michele Ciavotta, Edoardo Amaldi

Adaptive, **T**rustworthy, **M**anageable, **O**rchestrated, **S**ecure,
Privacy-assuring **H**ybrid, **E**cosystem for **RE**silient Cloud Computing

- European and Brazilian Research Innovation Action project, within the H2020 program, in the field of Cloud Computing

POLITECNICO
MILANO 1863

Thanks for your attention!