Calendario prossimi incontri

~~Lunedi~~	~~10/4~~				
~~Mercoledi~~	~~12/4~~				
Giovedi	13/4	9-11	E	Datacenters	(lezione)
Lunedi	17/4	11-13	E	Extra Red	(seminario)
Mercoledi	19/4	9:30-13		Gita a S. Piero	

Cloud Computing (**LAB**)

HANDS ON GFogBrain

Giuseppe Bisicchia
giuseppe.bisicchia@phd.unipi.it

Dipartimento di Informatica, Università di Pisa

Cosa faremo?

- Affronteremo il problema del *placement* di microservizi sul Continuum Cloud-IoT.
- Svilupperemo una strategia che ci consentirà di capire dove *«piazzare»* i microservizi rispettandone i requisiti e stimandone il **costo monetario**, l'**energia consumata** e la **CO2** prodotta da ogni **piazzamento**.

L’approccio dichiarativo

- *Un **piazzamento** è **valido** se ogni servizio è piazzato in un **nodo compatibile**.*

- *Un **servizio** è **piazzato** in un **nodo compatibile** se…*

Dalla logica proposizionale … al Prolog

Regole *X è divertente AND X è carino ➔ X è interessante*

Fatti *alberto è carino*
bruno è carino
bruno è divertente

Dalla logica proposizionale … al Prolog

Regole — *X è divertente AND X è carino ➔ X è interessante*

Fatti
alberto è carino
bruno è carino
bruno è divertente

Goal — *chi è un W interessante?*

Dalla logica proposizionale … al Prolog

Regole — *X è divertente AND X è carino ➔ X è interessante*

Fatti
alberto è carino
bruno è carino
bruno è divertente

Goal — *chi è un W interessante?*

```
interessante(X) :- carino(X), divertente(X).

carino(alberto).
carino(bruno).
divertente(bruno).

:-interessante(W).
```

Il motore di inferenza

```
interessante(X) :- carino(X), divertente(X).

carino(alberto).
carino(bruno).
divertente(bruno).
```

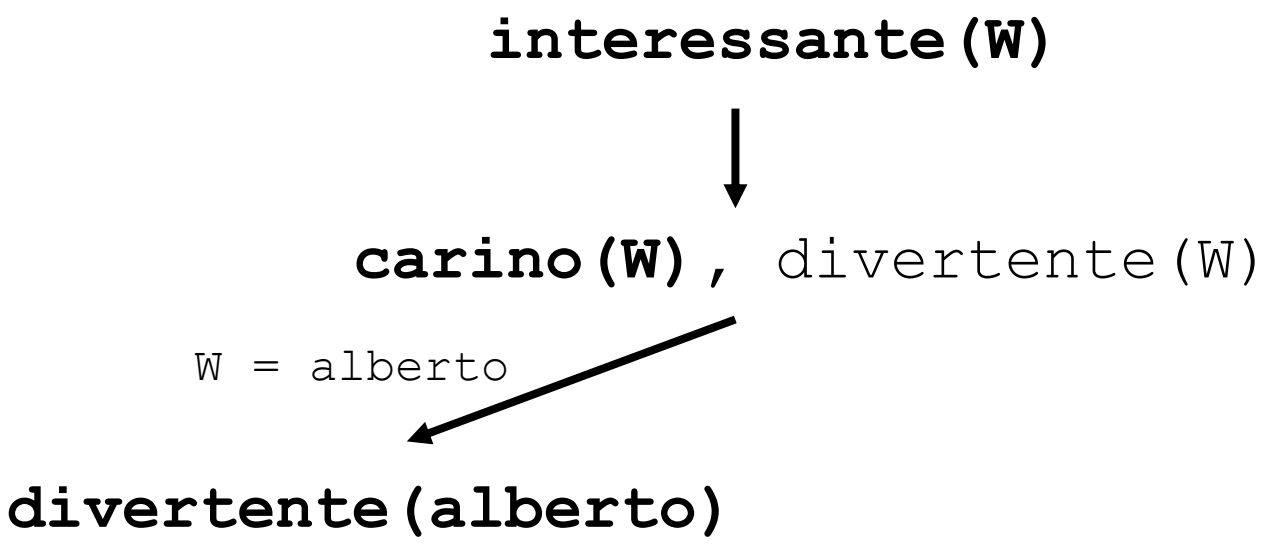
interessante(W)

```
interessante(X) :- carino(X), divertente(X).

carino(alberto).
carino(bruno).
divertente(bruno).
```

interessante(W)

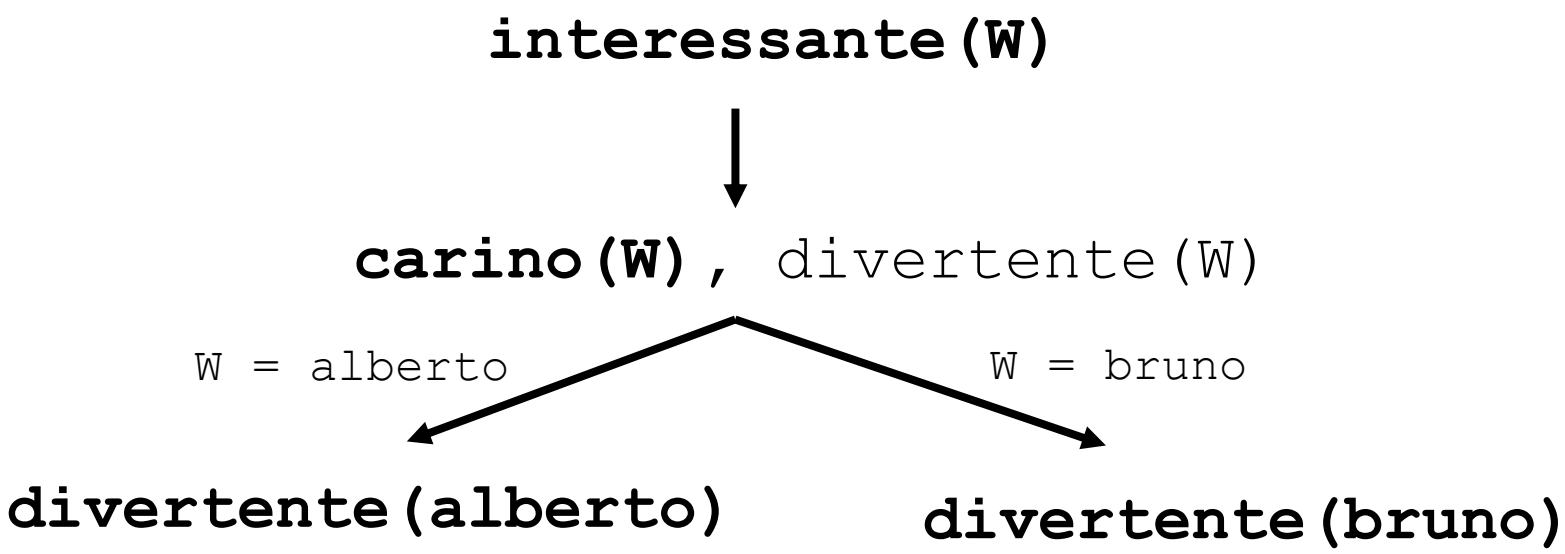
carino(W), divertente(W)

```
interessante(X) :- carino(X), divertente(X).

carino(alberto).
carino(bruno).
divertente(bruno).
```



failure

```
interessante(X) :- carino(X), divertente(X).

carino(alberto).
carino(bruno).
divertente(bruno).
```



failure

success

W = bruno

Prolog Cheat Sheet

- Variabile: simbolo che inizia con lettera maiuscola.
  - E.g., `Var`
- Fatto: `fact(…).` -> informazione sempre vera.
  - E.g., `persona(alberto).`
- Regola: `Head(…) :- Body.` -> se *Body* vero allora *Head* vero/quando chiami *Head* esegui *Body*.
  - E.g., `essere_vivente(X) :- persona(X).`
- Operazione Matematica: `Res is X op Y` -> Res = X op Y.
  - E.g., `Next is Current + 1`

SWISH Prolog

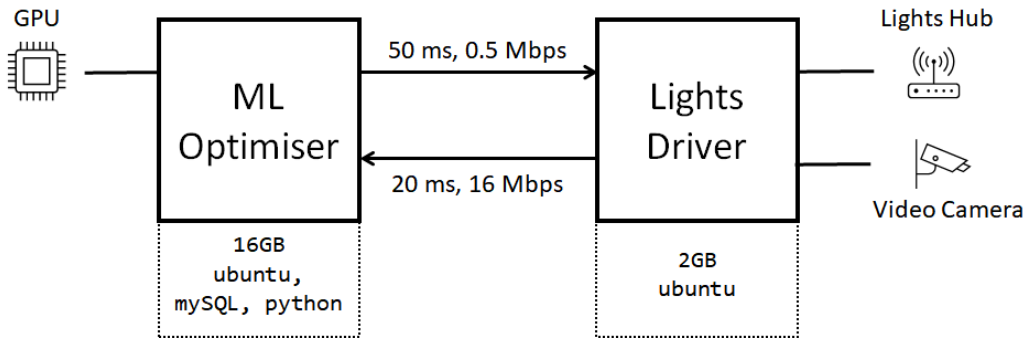
Durante il laboratorio useremo una versione online di Prolog disponibile al seguente link:

https://swish.swi-prolog.org/

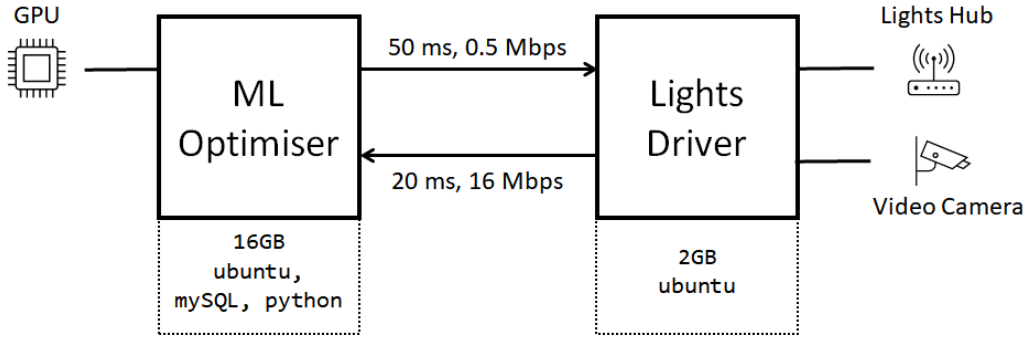
O alternativamente:

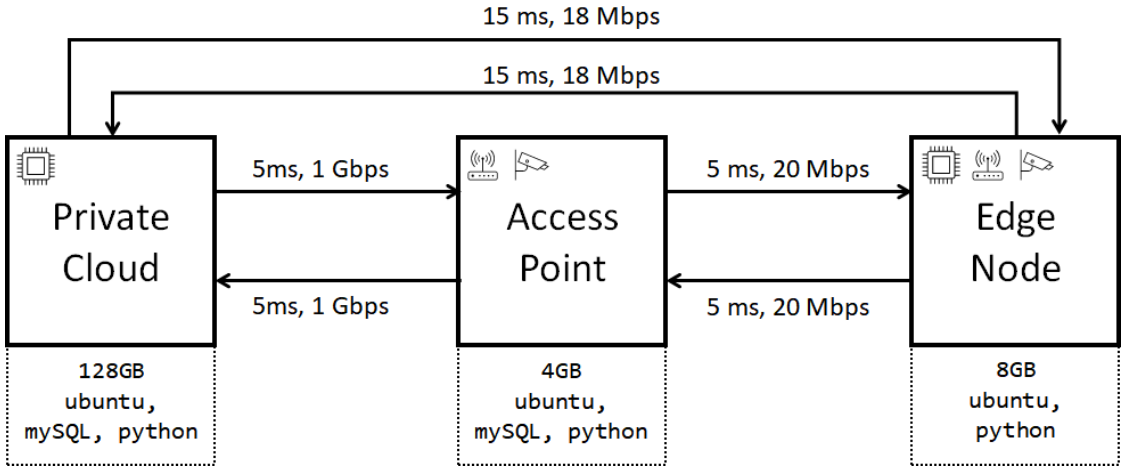
https://legalmachinelab.unibo.it/logicalenglish/

Il nostro obiettivo

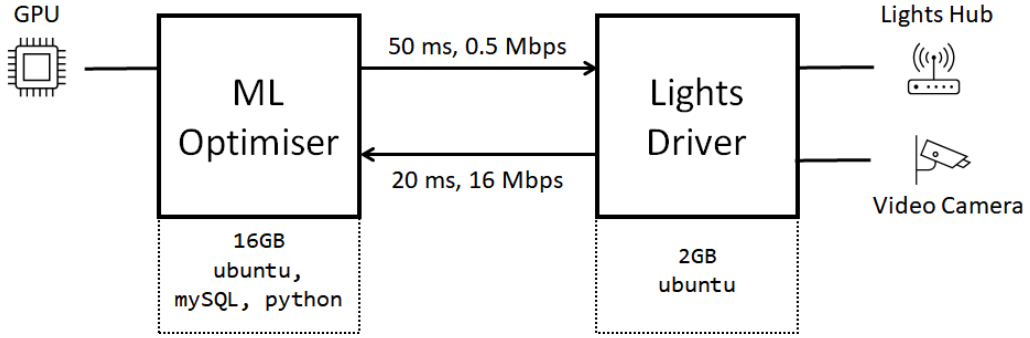


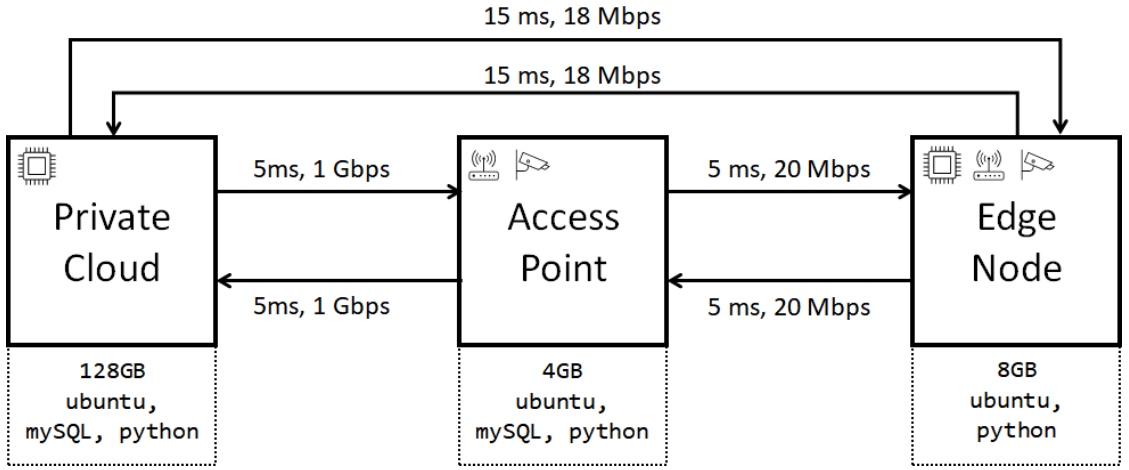
Il nostro obiettivo





Il nostro obiettivo





Id	Placement	Emissions	Cost	Energy Cons.
P_1	on(lightsDriver, edgenode), on(mlOptimiser, privateCloud)	0.29 $kgCO_2$	0.0356 €/h	0.60 kWh
P_2	on(lightsDriver, accesspoint), on(mlOptimiser, privateCloud)	0.32 $kgCO_2$	0.0316 €/h	0.63 kWh

Modelliamo un’applicazione

- Un’applicazione è definita da un identificatore «AppId» e da una lista di identificatori di servizi.

```
application(AppId, [Sid_1, ..., Sid_k]).
```

Modelliamo un’applicazione

- Un’applicazione è definita da un identificatore «AppId» e da una lista di identificatori di servizi.
- Un servizio è definito da un identificatore, una lista di requisiti software (e.g., [ubuntu, java]), un valore numerico rappresentante l’hardware (e.g., RAM) richiesto e una lista di dispositivi IoT necessari (e.g., [gpu]).

```
application(AppId, [Sid_1, ..., Sid_k]).

service(SId, [SwReq_1, ..., SwReq_n], HwReq, [IoTReq_1, ..., IoTReq_n]).
```

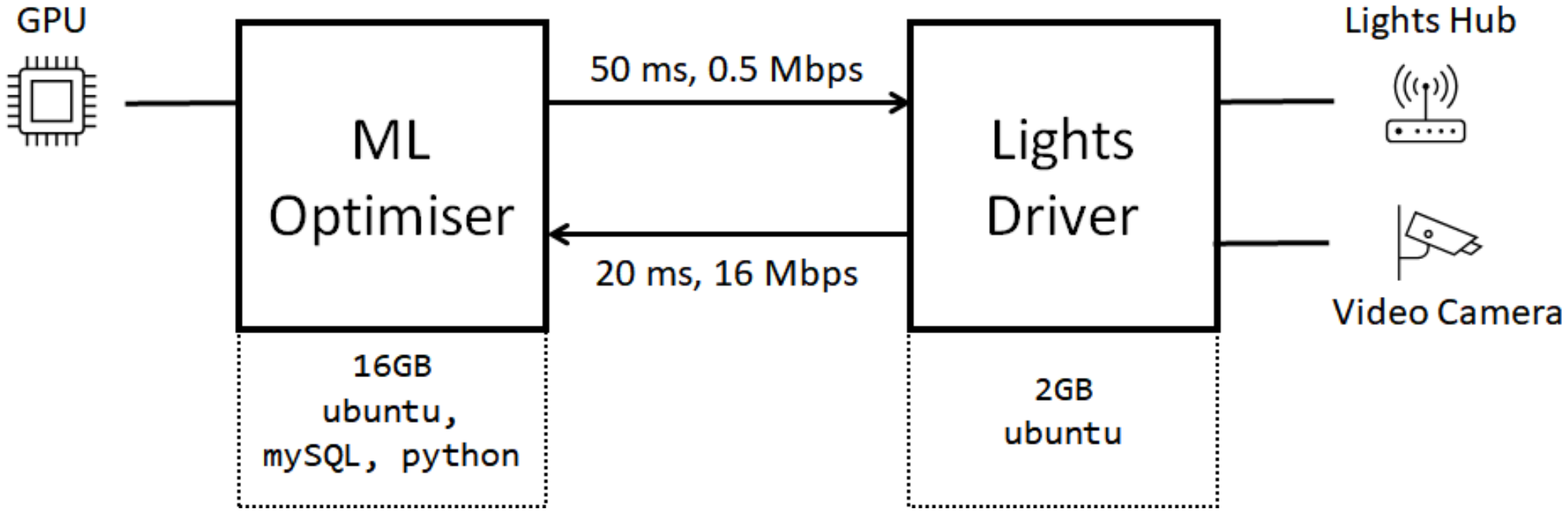
Modelliamo un’applicazione

- Un’applicazione è definita da un identificatore «AppId» e da una lista di identificatori di servizi.
- Un servizio è definito da un identificatore, una lista di requisiti software (e.g., [ubuntu, java]), un valore numerico rappresentante l’hardware (e.g., RAM) richiesto e una lista di dispositivi IoT disponibili (e.g., [gpu]).
- Ad ogni interazione (i.e., s2s) tra due servizi (SId_1 e SId_2) associamo la massima latenza accettabile e la minima banda richiesta.

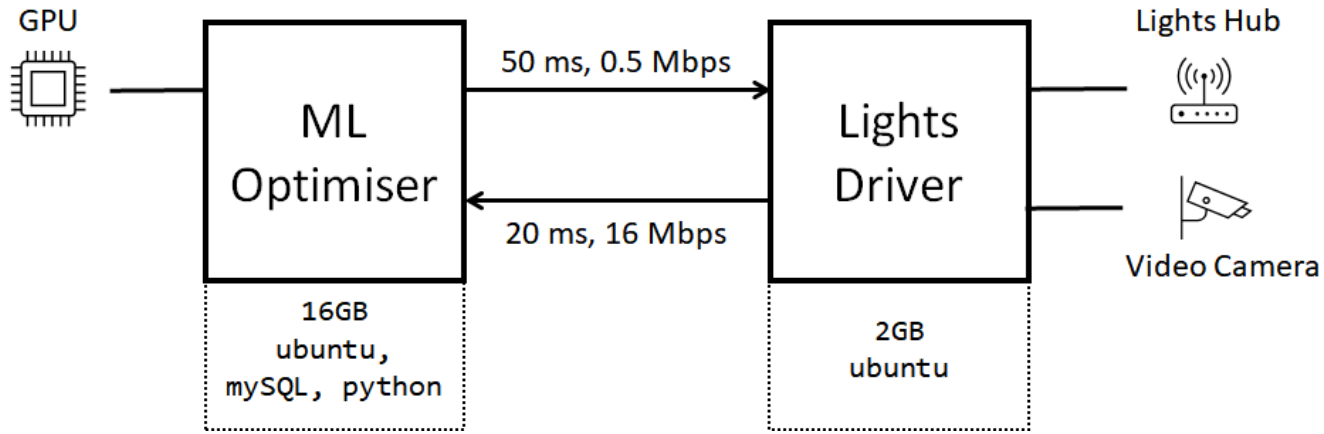
```
application(AppId, [Sid_1, ..., Sid_k]).

service(SId, [SwReq_1, ..., SwReq_n], HwReq, [IoTReq_1, ..., IoTReq_n]).

s2s(SId_1, SId_2, MaxLatency, MinBandwidth).
```

Esercizio: Modelliamo un'applicazione

Modelliamo in Prolog l'applicazione seguente come una serie di fatti.



Esercizio: Modelliamo un'applicazione




```
application(lightsApp, [mlOptimiser, lightsDriver]).
    service(mlOptimiser, [mySQL, python, ubuntu], 16, [gpu]).
    service(lightsDriver, [ubuntu], 2, [videocamera, lightshub]).
    s2s(mlOptimiser, lightsDriver, 50, 0.5).
    s2s(lightsDriver, mlOptimiser, 20, 16).
```

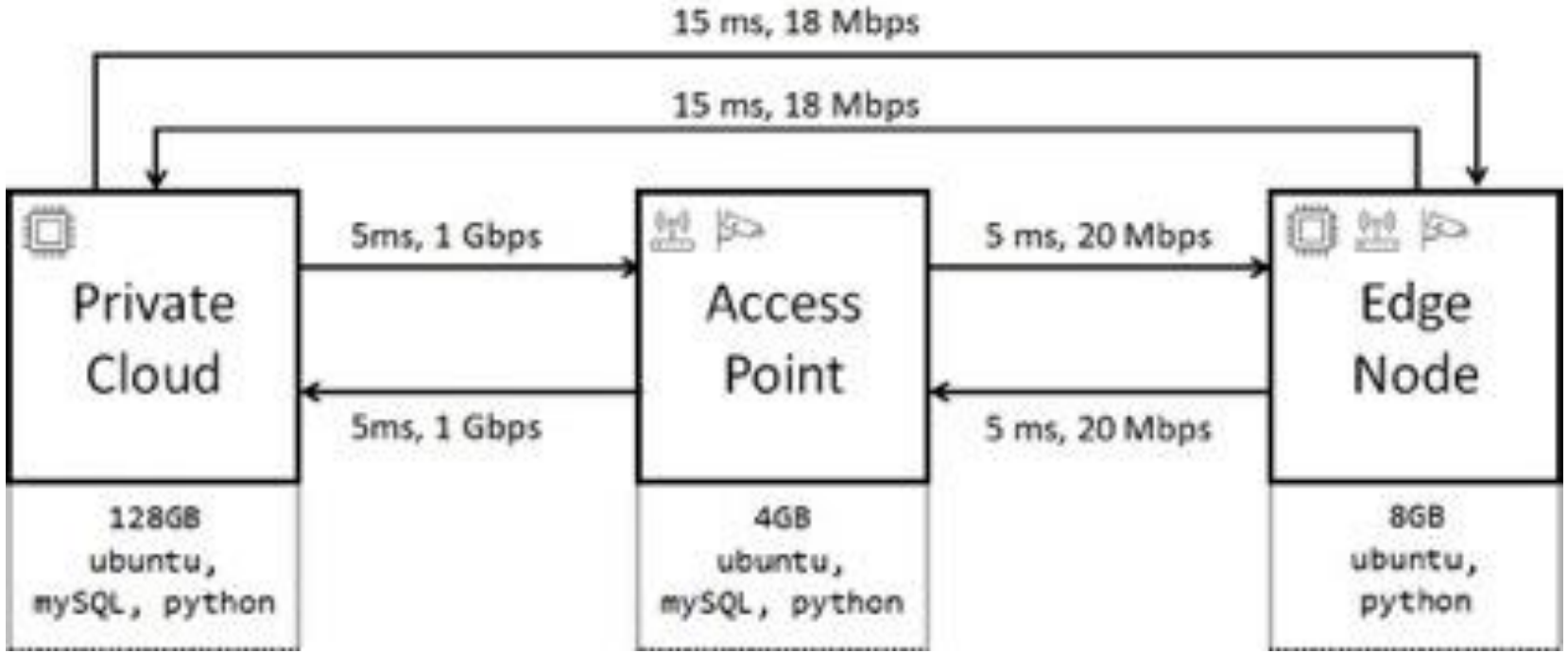
Esercizio: Modelliamo il Continuum

- Un’infrastruttura è definita da un insieme di nodi e di link tra nodi.
- Un nodo è definito da un identificatore, una lista di software disponibili (e.g., [ubuntu, java]), un valore numerico rappresentante l’hardware disponibile (e.g., RAM) e una lista di dispositivi IoT necessari (e.g., [gpu]).
- Ad ogni link tra due nodi (NId_1 e NId_2) associamo la latenza e la banda misurate.

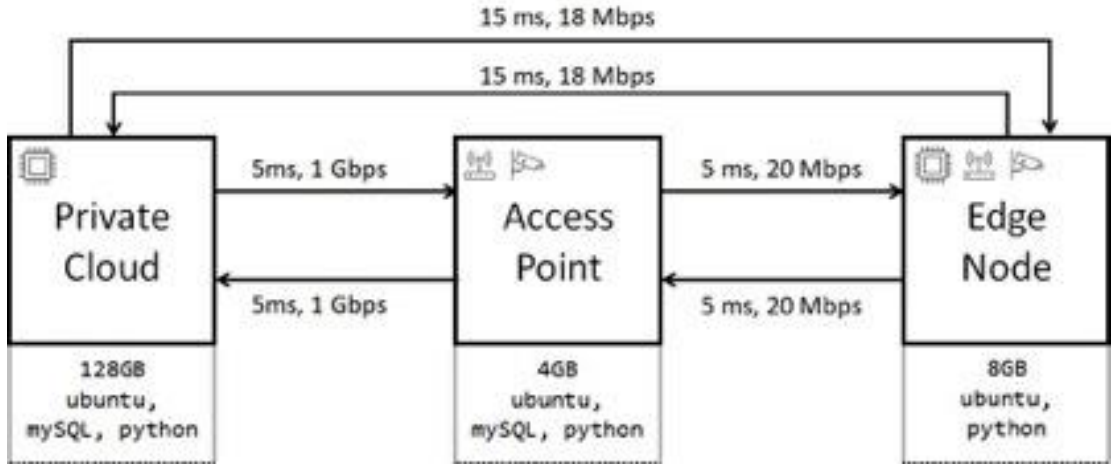
```
node(NId, [SwCap_1, ..., SwCap_n], FreeHw, [IoTCap_1, ..., IoTCap_n]).

link(NId_1, NId_2, FeaturedLatency, FeaturedBandwidth).
```

Esercizio: Modelliamo il Continuum

Modelliamo in Prolog l'infrastruttura seguente come una serie di fatti.



Esercizio: Modelliamo il Continuum




```
node(privateCloud,[ubuntu, mySQL, python], 128, [gpu]).
node(accesspoint,[ubuntu, mySQL, python], 4, [lightshub, videocamera]).
node(edgenode,[ubuntu, python], 8, [gpu, lightshub, videocamera]).

link(privateCloud, accesspoint, 5, 1000).
link(accesspoint, privateCloud, 5, 1000).
link(accesspoint, edgenode, 5, 20).
link(edgenode, accesspoint, 5, 20).
link(privateCloud, edgenode, 15, 18).
link(edgenode, privateCloud, 15, 18).
```

Esercizio: nodeOk

- Una volta definito il nostro modello possiamo iniziare a implementare il nostro *placer*.
- Una delle decisioni più importanti che dobbiamo prendere è specificare quando è possibile piazzare un servizio S su un particolare nodo N.
- Scrivete un predicato nodeOk, che dato l'identificatore S di un servizio ed N di un nodo è vero se i requisiti del servizio S possono essere soddisfatti dal nodo N.
- Provate poi a verificare sull'interprete possibili piazzamenti.

Esercizio: nodeOk

```
nodeOk(S,N) :-
  node(N,SWCaps,HWCaps,IoTCaps),
  service(S,SWReqs,HWReqs,IoTReqs),
  subset(SWReqs, SWCaps),
  subset(IoTReqs,IoTCaps),
  HWReqs < HWCaps.
```

Generiamo un placement

```
placement(Services,P) :-
   placement(Services,[],([],[]),P), allocatedResources(P,Alloc).

placement([S|Ss],P,(AllocHW,AllocBW),Placement) :-
   nodeOk(S,N,P,AllocHW), % checks sw, hw, IoT and cumulative hw
   reqs of s
   linksOk(S,N,P,AllocBW), % checks lat and cumulative bw reqs of s
   with s' in P
   placement(Ss,[on(S,N)|P],(AllocHW,AllocBW),Placement).
placement([],P,_,P).
```


MA DOV'E' IL
Green
Computing
???


THE UNITED STATES OF AMERICA
ONE DOLLAR
FEDERAL RESERVE NOTE
THIS NOTE IS LEGAL TENDER FOR ALL DEBTS, PUBLIC AND PRIVATE
WASHINGTON, D.C.

Costo di un placement

- Prima di parlare delle stime energetiche e dell'emissioni dei nostri placement consideriamo il problema di stimare il costo di un particolare placement.
- Le tecniche e i ragionamenti che affronteremo per questo problema ci saranno utili per affrontare gli aspetti green.

Costo di un placement

- Ogni qualvolta vogliamo aggiungere nuove informazioni da considerare nel nostro placement dobbiamo prima di tutto comprendere come cambia il nostro modello.

```
application(AppId, [Sid_1, ..., Sid_k]).

service(SId, [SwReq_1, ..., SwReq_n], HwReq, [IoTReq_1, ..., IoTReq_n]).

s2s(SId_1, SId_2, MaxLatency, MinBandwidth).
```

```
node(NId, [SwCap_1, ..., SwCap_n], FreeHw, [IoTCap_1, ..., IoTCap_n]).

link(NId_1, NId_2, FeaturedLatency, FeaturedBandwidth).
```

Costo di un placement

- Nel caso del costo di un placement vogliamo sapere quanto ci costa utilizzare un particolare nodo. Dobbiamo quindi aggiungere un'ulteriore informazione associata ad ogni nodo: quanto ci costa utilizzare un'unità hardware del nodo.

```
application(AppId, [Sid_1, ..., Sid_k]).
service(SId, [SwReq_1, ..., SwReq_n], HwReq, [IoTReq_1, ..., IoTReq_n]).
s2s(SId_1, SId_2, MaxLatency, MinBandwidth).
```

```
node(NId, [SwCap_1, ..., SwCap_n], FreeHw, [IoTCap_1, ..., IoTCap_n]).
cost(NId, Cost).
link(NId_1, NId_2, FeaturedLatency, FeaturedBandwidth).
```

Costo di un placement

- Adesso, dopo aver trovato un placement valido dobbiamo calcolare il costo di quel placement.
- Generiamo quindi tutti i placement validi con il predicato placements/2 e per ognuno di essi calcoliamo il costo con il predicato hourlyCost/2.

```
placements(A,Placements) :-
   findall((Cost,P), (placement(A,P), hourlyCost(P,Cost)), Ps),
   sort(Ps,Placements).

hourlyCost([on(S,N)|P],NewCost) :-
    hourlyCost(P,OldCost),
    service(S,_,HW,_), cost(N,C),
    NewCost is OldCost + C * HW.
hourlyCost([],0).
```

Diventiamo Green

Recap

Id	Placement	Emissions	Cost	Energy Cons.
P_1	on(lightsDriver, edgenode), on(mlOptimiser, privateCloud)	0.29 $kgCO_2$	0.0356 €/h	0.60 kWh
P_2	on(lightsDriver, accesspoint), on(mlOptimiser, privateCloud)	0.32 $kgCO_2$	0.0316 €/h	0.63 kWh

Come cambia il modello?

- Dobbiamo modellare l’impatto ambientale che ha un determinato placement.
- Lo faremo considerando l’energia consumata e la CO2 prodotta.

Come cambia il modello?

- Nel nostro modello andremo a considerare, per ogni nodo:
  - PUE (Power usage effectiveness);
  - Il profilo energetico (i.e., quanta energia consuma un particolare nodo per un particolare carico di lavoro)
  - Il mix energetico (i.e., la percentuale di energia impiegata per ogni tipo di fonte)

PUE

Per ogni kWh speso in computazioni, 0.9 sono spesi per funzionalità non-IT (e.g., raffreddamento, networking).

```
node(privateCloud,[ubuntu, mySQL, python], 128, [gpu]).
pue(privateCloud,1.9).
```

Profilo Energetico

Una funzione (possibilmente) non-lineare sulla percentuale di carico L.

```
node(privateCloud,[ubuntu, mySQL, python], 128, [gpu]).
pue(privateCloud,1.9).
energyProfile(privateCloud,L,E) :- E is 0.1 + 0.01*log(L).
```

Profilo Energetico

Dobbiamo conoscere anche la quantità totale di hardware presente sul nodo (150), non solo quella attualmente disponibile (128)

```
node(privateCloud,[ubuntu, mySQL, python], 128, [gpu]).
pue(privateCloud,1.9).
energyProfile(privateCloud,L,E) :- E is 0.1 + 0.01*log(L).
totHW(privateCloud,150).
```

Mix Energetico

Percentuale di energia ricevuta da ogni tipo di sorgente.

```
node(privateCloud,[ubuntu, mySQL, python], 128, [gpu]).
pue(privateCloud,1.9).
energyProfile(privateCloud,L,E) :- E is 0.1 + 0.01*log(L).
totHW(privateCloud,150).
energySourceMix(privateCloud,[(0.3,solar), (0.7,coal)]).
```

Mix Energetico

Per ogni fonte di energia dobbiamo conoscere il suo impatto ambientale in termine di CO2 (in kg/kWh).

```
emissions(gas, 0.610).
emissions(coal, 1.1).
emissions(onshorewind, 0.0097).
emissions(offshorewind, 0.0165).
emissions(solar, 0.05).
```

Consumo Energetico

Possiamo stimare l'energia consumata di un servizio S su un nodo N come il prodotto del PUE del nodo per il profilo energetico del nodo N, dove il carico è dato dalla percentuale di risorse richieste dal servizio S.

Esercizio: Impatto Ambientale

Adesso possiamo scrivere un predicato che dato un servizio S ed un nodo N stima l'impatto ambientale nel piazzare S su N in termini di energia consumata.

```
consumedEnergy(S,N,E) :- ???
```

Esercizio: Impatto Ambientale

```
consumedEnergy(S,N,E) :-
     service(S, _, ReqHW, _),
     totHW(N,TotHW), pue(N,PUE),
     Load is 100 * (TotHW - ReqHW) / TotHW,
     energyProfile(N,Load,ETemp),
     E is ETemp * PUE.
```

CO2 Prodotta

Per quanto riguarda la CO2 prodotta, applichiamo la seguente formula

$$I_s = E_s \times \sum_i p_i \mu_i \ [\mathrm{kgCO_2}]$$

Dove E_s è l'energia consumata, p_i la percentuale in cui è presente la fonte energetica i nel mix energetico e μ_i le emissioni della fonte i.

C02 Prodotta

```
hardwareEmissions([(P,S)|Srcs],Energy,Carbon) :-
     hardwareEmissions(Srcs,Energy,CarbSrcs),
     emissions(S,MU), CarbS is P * MU * Energy,
     Carbon is CarbS + CarbSrcs.
hardwareEmissions([],_,0).
```

GFogBrain

- Ora abbiamo tutto quello che ci serve per stimare l'impatto ambientale di ogni placement.
- Andiamo a modificare la funzione placements/2 in modo da considerare le nuove stime.

```
placements(A,Placements) :-
   findall((C,Cost,E,P), (gFogBrain(A,P,E,C), hourlyCost(P,Cost)), Ps),
   sort(Ps,Placements).

gFogBrain(A,P,Energy,CarbonEmissions) :-
    application(A,Services), placement(Services,[],([],[]),P),
    allocatedResources(P,Alloc), footprint(P,Alloc,Energy,CarbonEmissions).
```

Footprint di un placement

```
footprint(P,(AllocHW,AllocBW),Energy,Carbon) :-
    deploymentNodes(P,Nodes),
    hardwareFootprint(Nodes,P,AllocHW,0,HWEnergy,0,HWCarbon),
    networkFootprint(AllocBW,BWEnergy,BWCarbon),
    Energy is HWEnergy + BWEnergy,
    Carbon is HWCarbon + BWCarbon.

deploymentNodes(P,Nodes) :-
    findall((N,FreeHW), distinct((member(on(_,N),P), node(N,_,FreeHW,_))), Nodes).
```

hardwareFootprint

```
hardwareFootprint([(N,HW)|Ns],AllocHW,Energy,Carbon) :-
    hardwareFootprint(Ns,AllocHW,EnergyNs,CarbonNs),
    hardwareEnergy(N,HW,AllocHW,EnergyN),
    energySourceMix(N,Sources), hardwareEmissions(Sources,EnergyN,CarbonN),
    Energy is EnergyN+EnergyNs, Carbon is CarbonN+CarbonNs.
hardwareFootprint([],_,0,0).
```

Esercizio: Priorità diverse

Modificate l'algoritmo in modo che i placement vengano ordinati in base al seguente ordine di priorità: energia consumata, CO2 prodotta e costo.

```
placements(A,Placements) :-
   findall((E,C,Cost,P), (gFogBrain(A,P,E,C), hourlyCost(P,Cost)), Ps),
   sort(Ps,Placements).
```

Esercizio: What-if Analysis

Provate a modificare il nostro modello (e.g., profilo energetico dei nodi, requisiti dei servizi) e vedete come cambiano i placement e le stime.

Documentazione

- https://www.swi-prolog.org/pldoc/doc_for?object=manual
- https://prolog.readthedocs.io/_/downloads/en/latest/pdf/

More details

Stefano Forti, Giuseppe Bisicchia, Antonio Brogi
Declarative Continuous Reasoning in the Cloud-IoT Continuum,
Journal of Logic and Computation (2022)

Stefano Forti, Antonio Brogi
Green Application Placement in the Cloud-IoT Continuum,
Practical Aspects of Declarative Languages (PADL 2022), LNCS, vol 13165.