DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

ITEA

INFORMATION TECHNOLOGY

FOR EUROPEAN ADVANCEMENT

(ITEA2 09011)

Power capping in SLURM

First prototype results and feedback

Green days @Lille, November 2013 | Matthieu Hautreux <matthieu.hautreux@cea.fr>

Power capping in SLURM
Agenda

Background & Objectives

Prototype design & First results

Feedback & Next steps

Power capping in SLURM

Background & Objectives

Last decades feedback

- An ever growing requirement for both computational power and storage resources
  - For scientific and industrial usages
- An increasing pressure on the ability to efficiently manage the associated power
  - For cost and/or environmental reasons

Power management features of SLURM 2.6.x

- Power saving logic
  - Enabling to shutdown idle nodes when necessary
  - Enabling to bring back shutdown nodes when required
- Power monitoring logic
  - Enabling to get the amount of required power per node (Watts)
    - Using in-band IPMI or external retrieval mechanism
  - Enabling to estimate jobs power consumptions (Joules)
- Dynamic frequency scaling logic
  - Enabling to modify the max allowed frequency per job
  - Thus enabling users to search for power efficient jobs

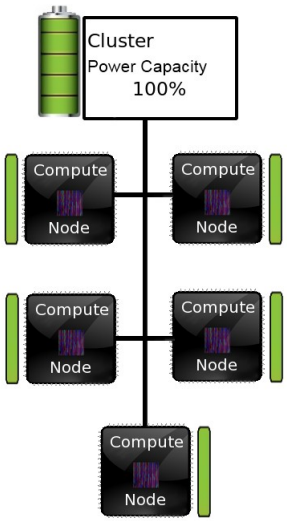
Dynamic Power capping in SLURM

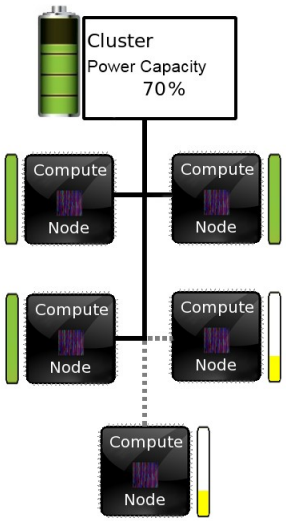
- Provide a mechanism to cap the amount of available power
  - Limiting the amount of usable resources on the clusters
- Provide a way to plan power cuts to limit the available power in advance
  - Helping to adapt to dynamic energy provisioning/pricing systems
- Work partially funded by the H4H-Perfcloud ITEA2 project
  - http://www.h4h-itea2.org

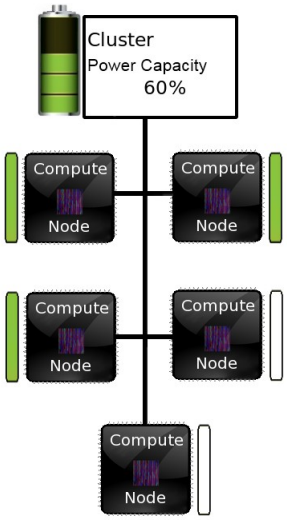
Power capping in SLURM
Background & Objectives

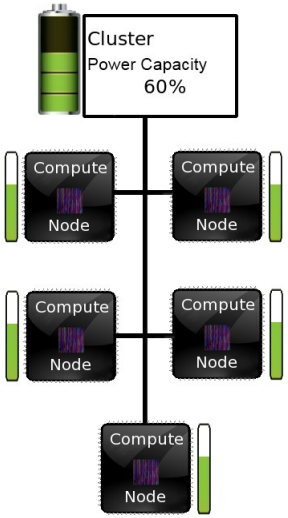
Operating clusters in dynamic power budgets

- Cycling through different states
  - Running without constraint
  - Coping with the power constraint
    - Leaving some nodes idle
  - Minimizing the power constraint
    - Shutting down nodes to use more nodes in a same budget
  - Optimizing despite the constraint
    - Using more nodes at a lower frequencies







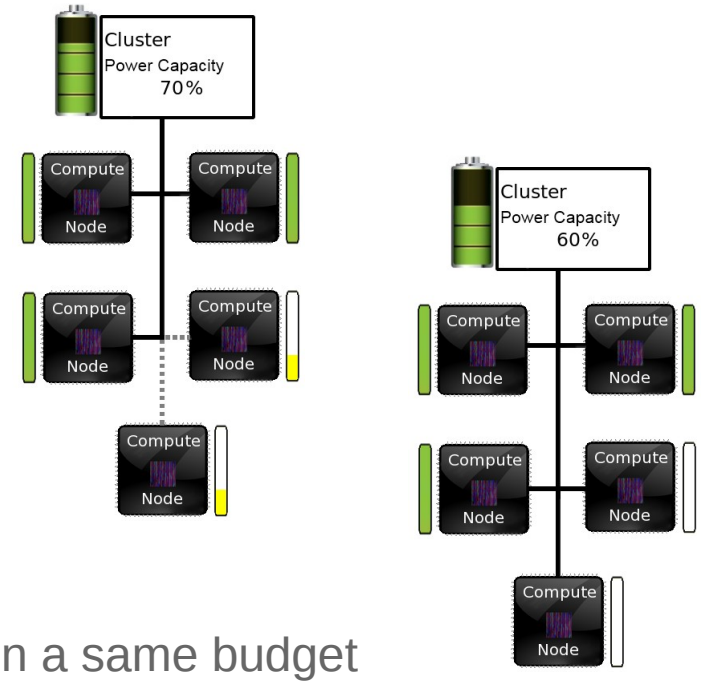


Power capping in SLURM

Prototype design & First results

Current prototype focus

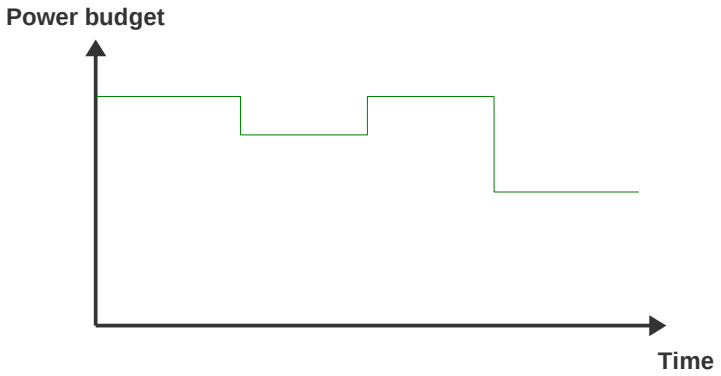
- Cope with the energy constraint
  - Leaving some nodes idle



- Minimize the constraint consequences
  - Shutting down nodes to use more nodes in a same budget
  - Linking the capping logic with the « power saving » logic of SLURM
    - Automatic shutdown of idle nodes after a long inactive period

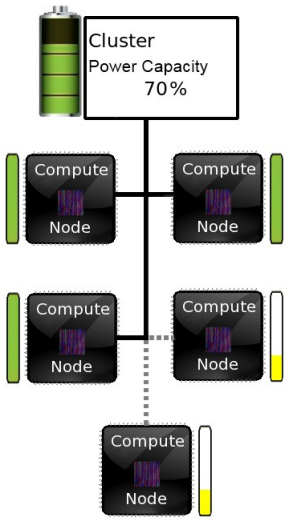
Current prototype focus

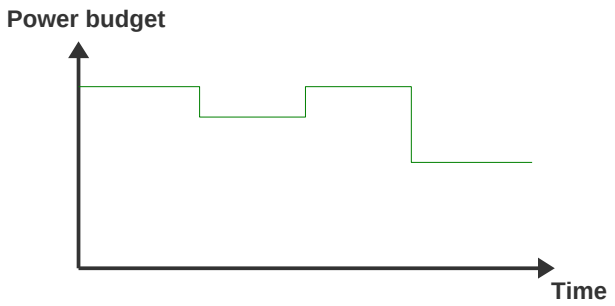
- Provide fine-grained power budget schedule
  - Using a default cap to respect in all cases
    - Potentially all the necessary power for the nodes
  - Scheduling additional « power cuts » on demand
    - Through the reservation system of SLURM



SLURM parameters addition

- On a node basis
  - MinWatts : the idle consumption in Watts
  - MaxWatts : the max consumption in Watts
  - PowerSavedWatts : the max consumption in Watts of a «power saved » node
  - DownWatts : the max consumption of a down node
- On a controller basis
  - PowerCap : the power cap to respect in Watts
- A new « power reservation » logic
  - Adding a « Watts » parameter to identify the amount of power to reserve
    - Between the start and end times
  - Similar to the « license reservation »
    - No nodes attached (LICENSE_ONLY)

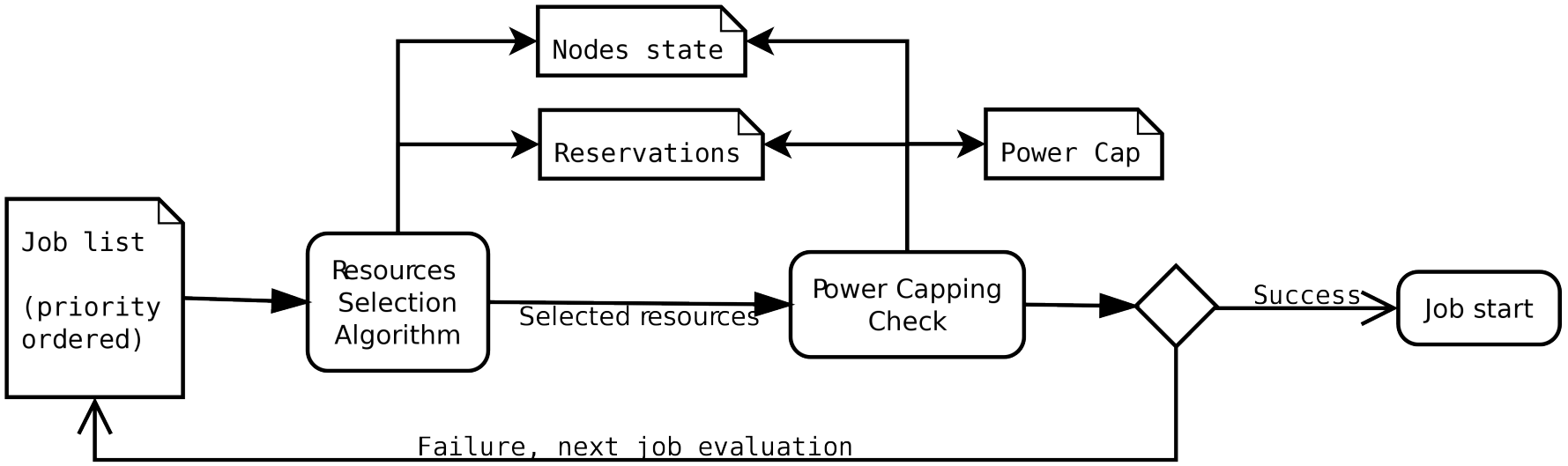




Algorithm

- For each job scheduling attempt
  - Let SLURM elect the best set of nodes
    - According to its own policy (topology, priorities, ...)
  - Evaluate the max amount of power required to operate the cluster adding the job
    - Considering the current/next state of each node
  - Evaluate the power cap that must be applied to the job
    - Taking into account the power cap of the cluster
    - And the overlapping « power reservations »
  - Let the job pending if the cap is lower than the required power amount
    - And block the resources for direct scheduling
      - The backfill logic will decide to start lower priority jobs

Algorithm


Nodes state
Reservations
Power Cap
Job list
(priority ordered)
Resources Selection Algorithm
Selected resources
Power Capping Check
Success
Job start
Failure, next job evaluation

Example 1

- Basic usage of the power capping logic

```
[mat@leaf0 utilit]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
physical     up   infinite      1       idle leaf0
virtual*     up   infinite    256  idle leaf[1000-1255]
[mat@leaf0 utilit]$ scontrol show powercap
MinWatts=116150 CurrentWatts=116150 PowerCap=INFINITE
AdjustedMaxWatts=244150 MaxWatts=244150
[mat@leaf0 utilit]$
```

```
[mat@leaf0 utilit]$ srun -n 10 -N 10 sleep 100 >/dev/null &
[1] 23819
[mat@leaf0 utilit]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
physical     up   infinite      1   idle leaf0
virtual*     up   infinite     10    mix leaf[1000-1009]
virtual*     up   infinite    246   idle leaf[1010-1255]
[mat@leaf0 utilit]$ scontrol show powercap
MinWatts=116150 CurrentWatts=121150 PowerCap=INFINITE AdjustedMaxWatts=244150
MaxWatts=244150
[mat@leaf0 utilit]$
```

Example 1 (cont'd)

```
[mat@leaf0 utilit]$ scontrol update powercap=121000

[mat@leaf0 utilit]$ scontrol show powercap
MinWatts=116150 CurrentWatts=116150 PowerCap=121000 AdjustedMaxWatts=244150
MaxWatts=244150

[mat@leaf0 utilit]$ srun -n 10 -N 10 sleep 100 >/dev/null &
[1] 25975
srun: Required power not available now
srun: job 5 queued and waiting for resources

[mat@leaf0 utilit]$ srun -n 10 -N 9 sleep 100 >/dev/null &
[2] 25978
[mat@leaf0 utilit]$ squeue
       JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           5   virtual    sleep      mat PD       0:00     10 (PowerNotAvail)
           6   virtual    sleep      mat  R       0:05      9 leaf[1000-1008]
[mat@leaf0 utilit]$
```

Example 1 (cont'd)

```
[mat@leaf0 utilit]$ scontrol show powercap
MinWatts=116150 CurrentWatts=120650 PowerCap=121000 AdjustedMaxWatts=244150
MaxWatts=244150

[mat@leaf0 utilit]$ squeue
       JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           5   virtual    sleep              mat PD 0:00     10 (PowerNotAvail)
[2]+  Done                          srun -n 10 -N 9 sleep 100 > /dev/null

[mat@leaf0 utilit]$ scontrol update powercap=INFINITE

[mat@leaf0 utilit]$ srun: job 5 has been allocated resources
[mat@leaf0 utilit]$ squeue
       JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           5   virtual    sleep              mat  R 0:14     10 leaf[1000-1009]
```

Example 2

- Leveraging the power saving logic

```
[mat@leaf0 utilit]$ scontrol show powercap
MinWatts=2230 CurrentWatts=116150 PowerCap=INFINITE AdjustedMaxWatts=244150 MaxWatts=244150

[mat@leaf0 utilit]$ scontrol update powercap=121000
[mat@leaf0 utilit]$ scontrol show powercap
MinWatts=2230 CurrentWatts=116150 PowerCap=121000 AdjustedMaxWatts=244150 MaxWatts=244150

[mat@leaf0 utilit]$ srun -n 10 -N 10 sleep 100 >/dev/null &
[1] 1629
srun: Required power not available now
srun: job 2 queued and waiting for resources

[mat@leaf0 utilit]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
physical     up   infinite      1   idle leaf0
virtual*     up   infinite    256   idle leaf[1000-1255]

[mat@leaf0 utilit]$ # waiting for nodes   to be shutdown
```

Example 2 (cont'd)

- Nodes entering power save state release additional power
  - Enabling the start of the pending job

```
[mat@leaf0 utilit]$ srun: job 2 has been allocated resources
[mat@leaf0 utilit]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
physical     up   infinite      1   idle leaf0
virtual*     up   infinite    107   idle~ leaf[1010-1116]
virtual*     up   infinite     10    mix leaf[1000-1009]
virtual*     up   infinite    139   idle leaf[1117-1255]
[mat@leaf0 utilit]$ scontrol show powercap
MinWatts=2230 CurrentWatts=73535 PowerCap=121000 AdjustedMaxWatts=143035
MaxWatts=244150
[mat@leaf0 utilit]$
```

Example 3

- Capping in advance using power reservations

```
[mat@leaf0 utilit]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
physical     up   infinite      1   idle leaf0
virtual*     up   infinite    256   idle leaf[1000-1255]
[mat@leaf0 utilit]$ scontrol show powercap
MinWatts=116150 CurrentWatts=116150 PowerCap=INFINITE AdjustedMaxWatts=244150 MaxWatts=244150

[mat@leaf0 utilit]$ scontrol create res FLAG=LICENSE_ONLY
starttime=now+20minutes duration=3 Watts=123150 Users=root
Reservation created: root_1

[mat@leaf0 utilit]$ scontrol show res
ReservationName=root_1 StartTime=2013-10-02T23:57:26 EndTime=2013-10-03T00:00:26
Duration=00:03:00
   Nodes= NodeCnt=0 CoreCnt=0 Features=(null) PartitionName=virtual Flags=LICENSE_ONLY
   Users=root Accounts=(null) Licenses=(null) Watts=123150 State=INACTIVE

[mat@leaf0 utilit]$ scontrol show powercap
MinWatts=116150 CurrentWatts=116150 PowerCap=INFINITE AdjustedMaxWatts=244150 MaxWatts=244150
[mat@leaf0 utilit]$
```

Example 3 (cont'd)

- Jobs not respecting overlapping future caps are blocked
- Jobs ending before the constraining future caps are executed
- Jobs respecting overlapping future caps are executed as well

```
[mat@leaf0 utilit]$ srun --immediate -t 40 -n 10 -N 10 sleep 100 >/dev/null
srun: Force Terminated job 2
srun: error: Unable to allocate resources: Required power at least partially reserved
[mat@leaf0 utilit]$ srun --immediate -t 15 -n 10 -N 10 sleep 60       &
[1] 10135
[mat@leaf0 utilit]$ scontrol show powercap
MinWatts=116150 CurrentWatts=121150 PowerCap=INFINITE AdjustedMaxWatts=244150 MaxWatts=244150
[1]+  Exit 1                  srun --immediate -t 20 -n 10 -N 10 sleep 60

[mat@leaf0 utilit]$ srun -t 20 -n 10 -N 10 sleep 60 &
[1] 12270
[mat@leaf0 utilit]$ srun: Required power at least partially reserved
srun: job 12 queued and waiting for resources
[mat@leaf0 utilit]$ srun --immediate -t 30 -n 10 -N 9 true >/dev/null

[mat@leaf0 utilit]$ scontrol delete reservation=root_1
[mat@leaf0 utilit]$ srun: job 12 has been allocated resources
[1]+  Done                    srun -t 20 -n 10 -N 10 sleep 60
[mat@leaf0 utilit]$
```

Power capping in SLURM

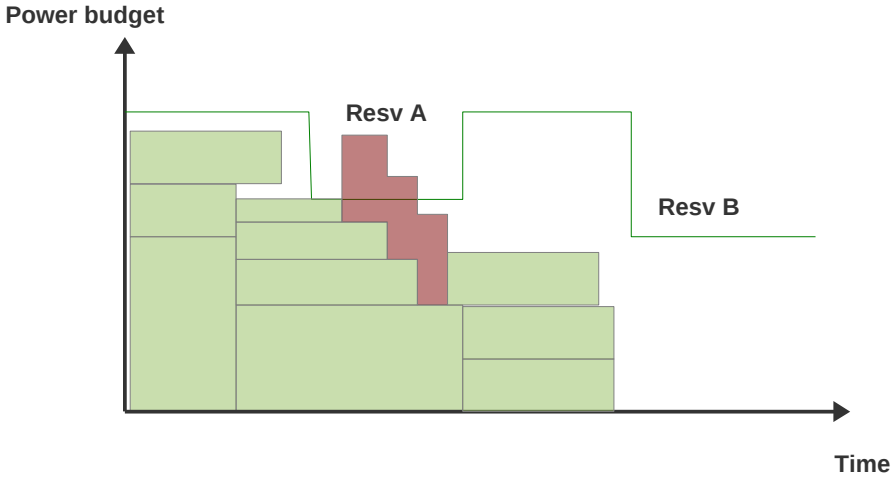
Feedback & Next steps

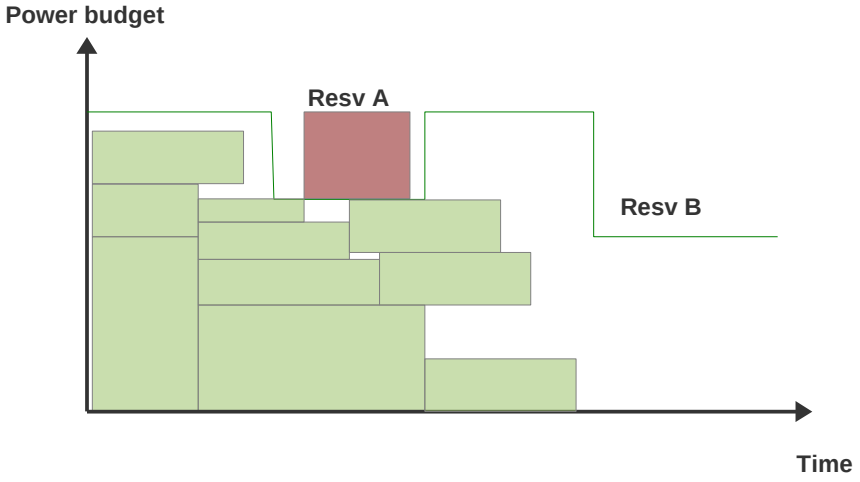
Scheduling algorithm

- The backfilling behavior is altered by the new power constraint
  - Resources can be available but not the requested power
  - Power hungry jobs are thus automatically skipped
    - Resulting in starvation of power hungry (large) jobs during « cuts »
    - The system becomes a « First-Fit » against the power resource
- Possible enhancement
  - Addition of an eligible start time for power blocked jobs
    - Computed based on the release of nodes by terminating jobs
    - Used in the backfilling logic to place lower priority jobs ending before that
  - Need to validate if such a notion is then sufficient
    - No more « first-fit » effect ?

Advanced power reservation

- Current logic enables to use the power of a particular reservation
  - Limited to the users associated to the reservation
    - Must be by default associated to admin users only
  - This creates temporary power respect anomalies
    - Should it be kept like that ?
      or using a power reservation should just be forbidden ?
      or treated separately from the rest of the available power ?





Nodes power consumptions

- Nodes power consumptions are hard to determine
  - *Idle* nodes can be used outside of slurm and use more power than assumed
  - *Down* nodes can also
    - Be used outside of slurm for nonreg tests (~MaxWatts)
    - Be idle but unreachable (~MinWatts)
    - Simply be shutdown (~0 Watts)
  - *Power saved* nodes can also be in different states
    - We need to at least be able to select between MinWatts or ~0 Watts in conf
- The effective power consumption differs from the assumed value
  - Shared nodes or nodes used with tampered frequencies can be in intermediate states between Min/MaxWatts
    - Need to find a way to have a table freq <-> Watts on a per core basis to compute a better estimation of the CurrentMaxWatts of the nodes

Nodes power consumptions

- Ensuring a global cap by ensuring local caps has limitations
  - less accurate but more deterministic
    - Applications do not permanently reach the highest consumptions of the nodes
  - But we do need determinism to avoid overconsumption

Next steps

- Improve the backfilling logic when respecting power caps
  - Avoid « first-fit the power » effect
- Evaluate with real workloads and use cases
  - To get confidence in the system
- Study scheduling alternatives
  - Not only checking the power constraint against the proposed solution
    - But looking for the best power candidate among multiple sched solutions
- Study the integration of the dynamic frequency scaling logic
  - To better estimate the required power in partial usage of the resources
  - Certainly using hardware power capping support to get determinism

Thank you for your attention

Questions ?

Commissariat à l'énergie atomique et aux énergies alternatives
Centre DAM Ile-de-France | Bruyères-le-Châtel 91297 Arpajon Cedex
T. +33 (0)1 69 26 40 00 |
Etablissement public à caractère industriel et commercial | RCS Paris B 775 685 019