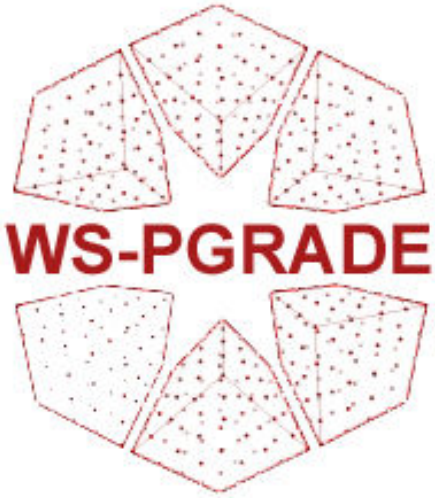
Parameter sweep job submission to Eucalyptus clouds

Peter Kacsuk, Attila Marosi
MTA SZTAKI
kacsuk@sztaki.hu



www.eu-egee.org

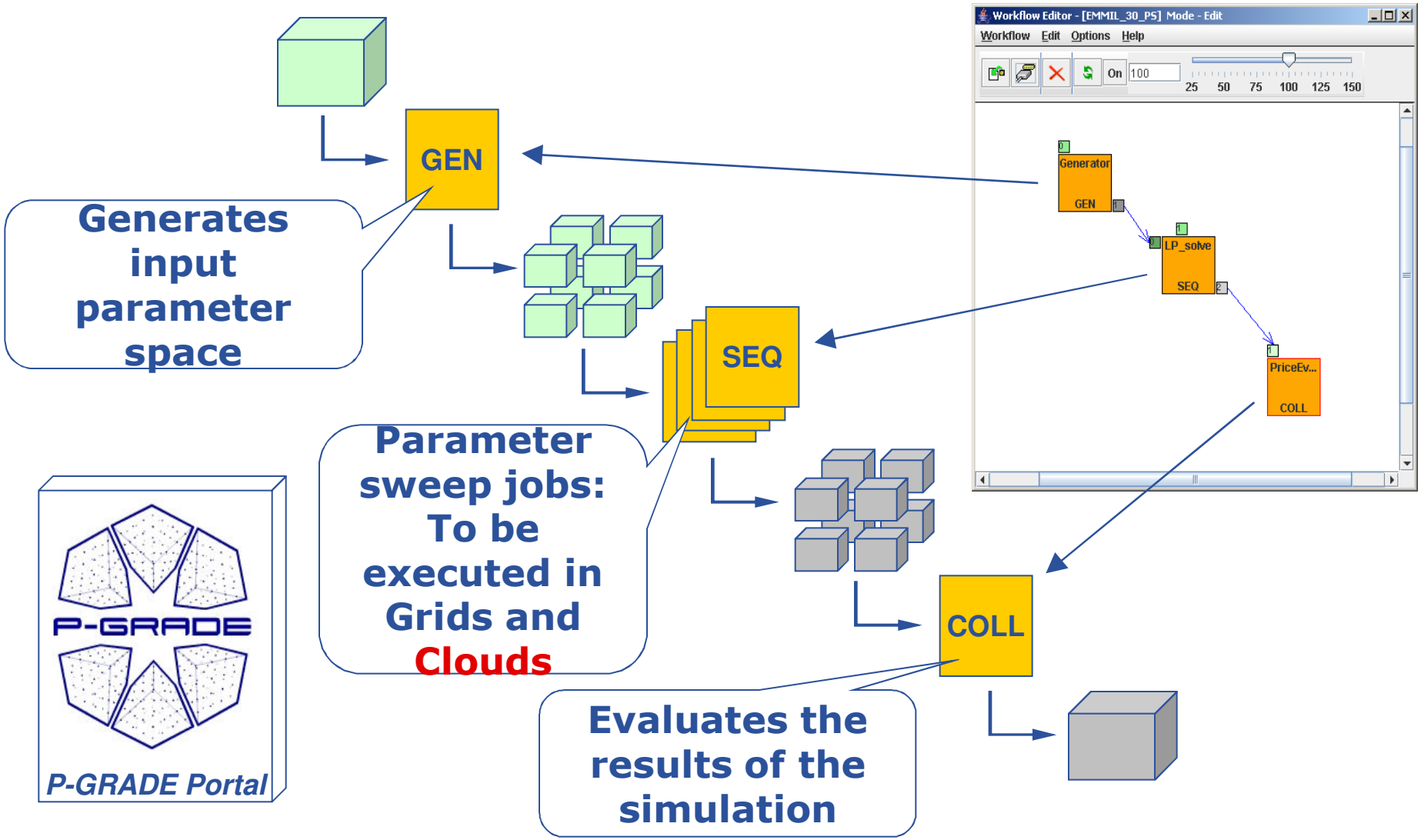
5th EGEE User Forum
Uppsala, 12-15 April 2010

e-infrastructure

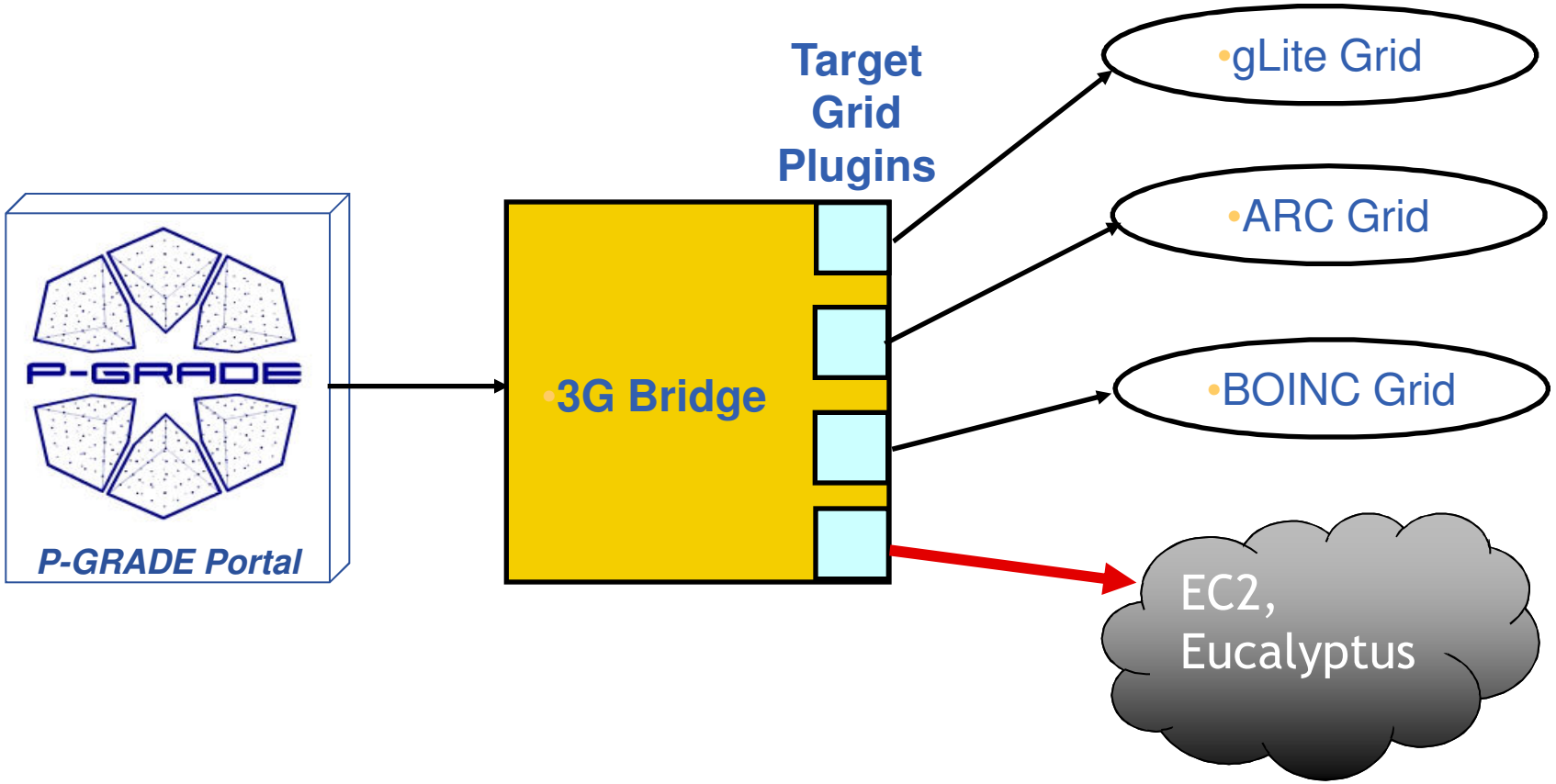
Outline

- Principles of parameter sweep job submission by P-GRADE portal
- Principles of parameter sweep job submission to various grids by 3G Bridge
- Variants of creating 3G Bridge cloud plugins
- Conclusions

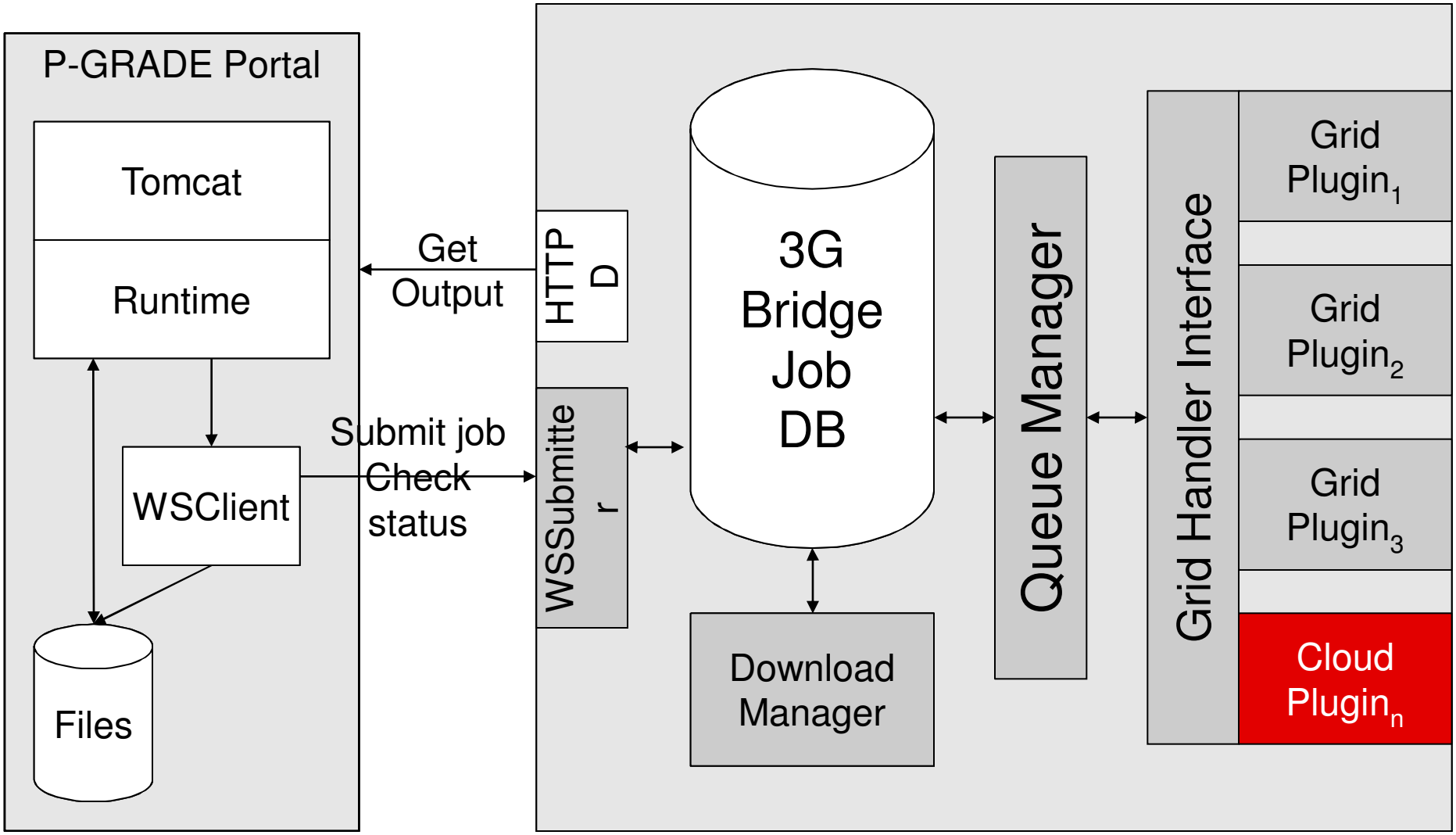
Parameter study workflow execution support in P-GRADE portal



PS job execution in Grids by P-GRADE



P-GRADE portal → 3G Bridge



Problems of creating cloud plug-in

- **The cloud plug-in needs to do three things at once**
- **It is not enough to submit the job (*job delegation - 1*), first a resource needs to be allocated (*resource management - 2*)**
- **There is no job manager on the allocated resource to submit the task to, and thus no** job scheduling (3)

Tasks to solve the problems:

1. **Cloud resource management**
2. **Job submission**
3. **Scheduling logic**

We investigated three variants of solving these problems

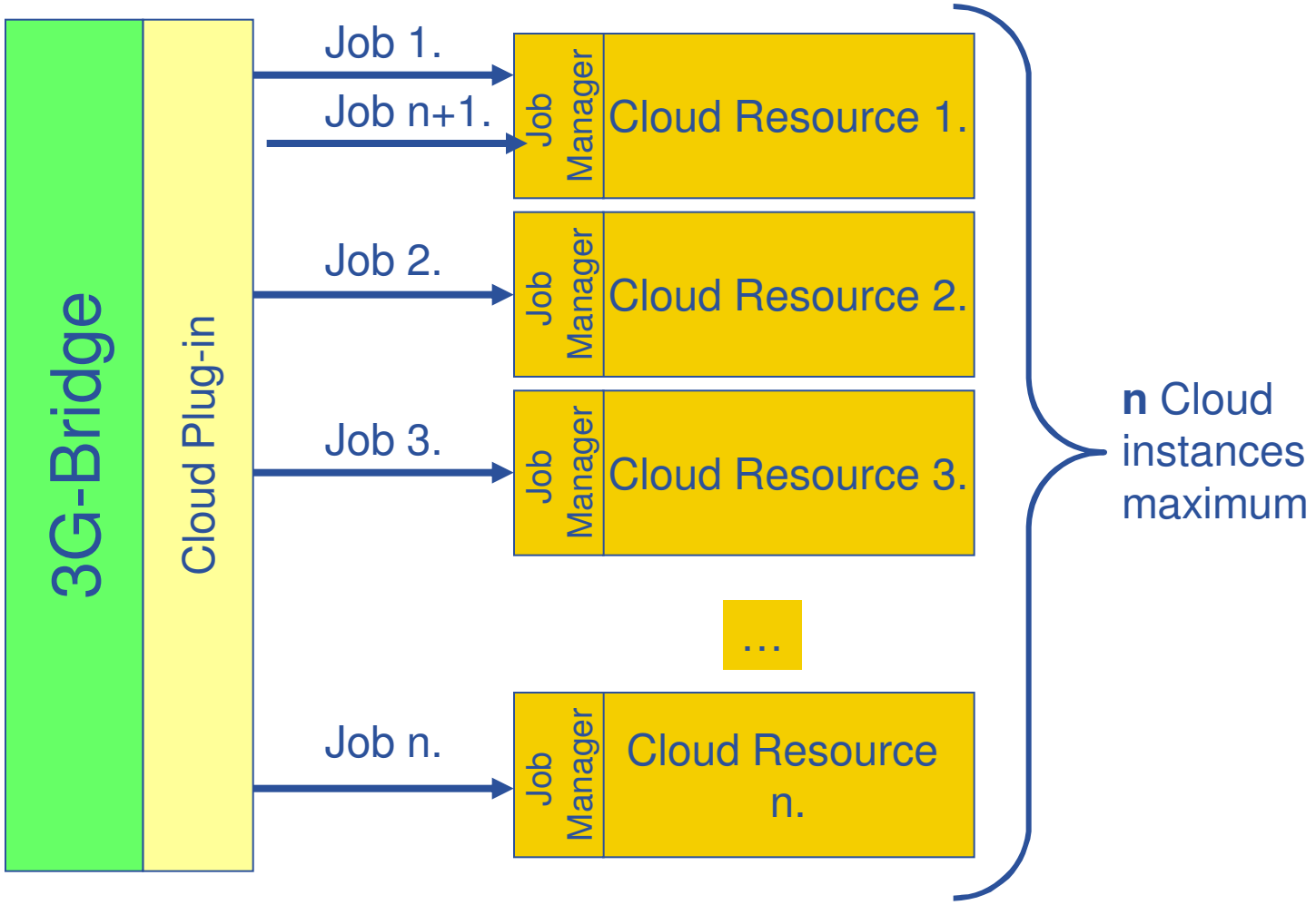
Variant I.

- **Allocate a cloud resource (linux instance) for each job**
- **Submit the job simply by copying it to the remote cloud resource and execute there**
  - no job manager required
- **A single plug-in implements all functionality**
- **This variant has been implemented first to gain experience with clouds**
- **Advantages:**
  - simple to implement
  - good for proof of concept
- **Drawbacks:**
  - not effective

Variant II.

- Set an upper limit for the maximum number of Cloud resources utilizable by the 3G-Bridge (“n”) – for safety
- Use a VM image that contains a job manager (e.g. Condor)
- Submit jobs using the job manager
- Schedule jobs between maximum n instances (resources) by the cloud plug-in

Variant II.



The plug-in manages the resources but: How to schedule tasks to instances ?

Variant II. – Scheduling example

```
start_and_submit:
    if (m==0) and (k<n) then
        instanceId = start_instance()
    else
        instanceId = find_min(j(0)..j(k-1))
    submit_job(instanceId, jobId)

stop:
    for i in 0..k-1 do
        if (t(i) < timestamp("-20 minutes")) and (j(i) == 0)
            stop_instance(i)
```

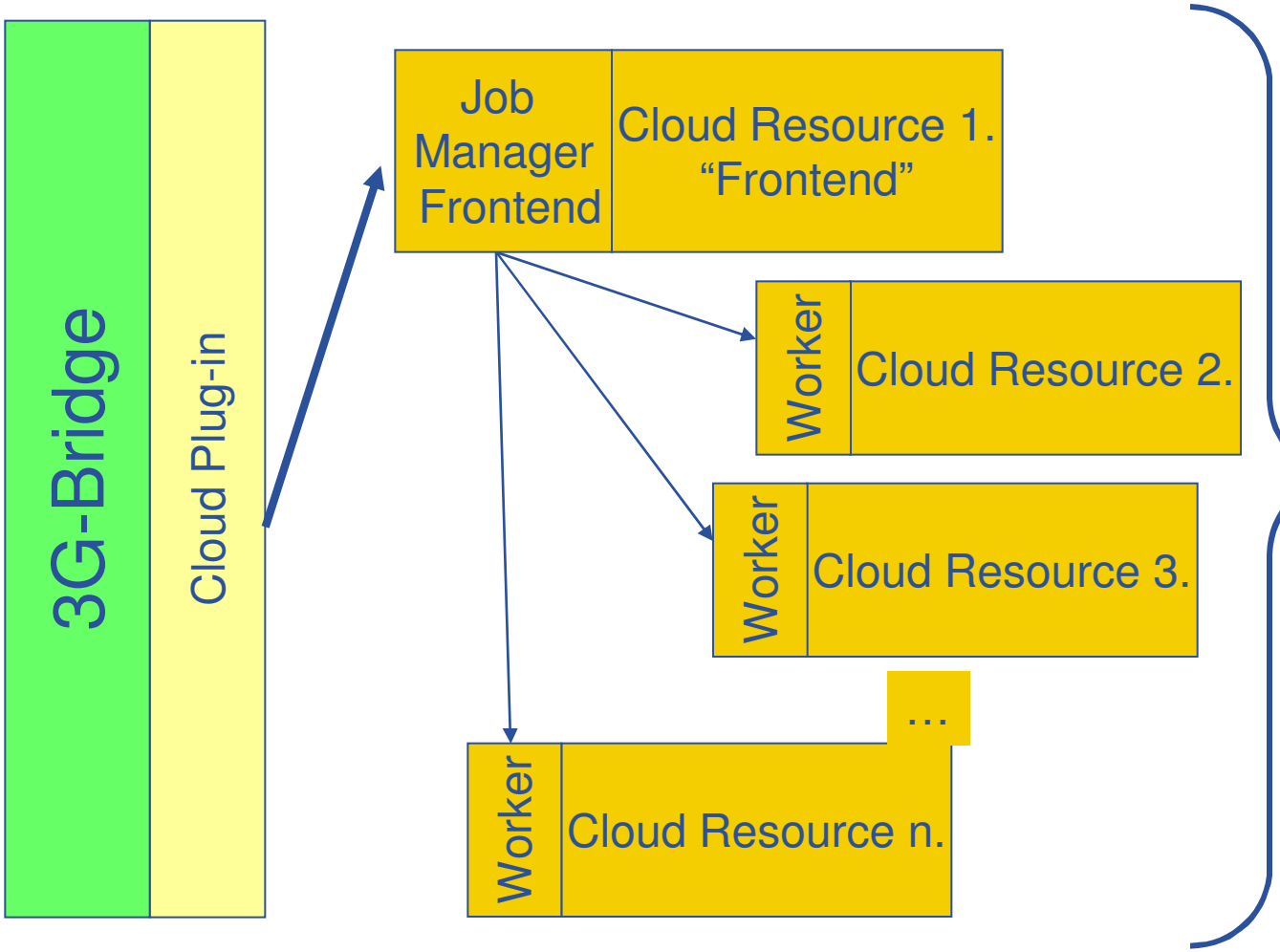
m: number of free instances
k: total number of running instances
n: maximum number of instances
j(x): number of jobs queued on instance x
t(x): timestamp of last execution on instance x

- We have no information how long a job is going to run, we can only estimate the load of any instance by the number of jobs queued on it
- We shutdown an instance if it has no work queued and did not execute tasks in the last 20 minute (20 chosen arbitrarily)

Variant II.

- **Benefits**
  - Still simple to implement
  - Uses reliable method for task submission (e.g. Condor WS API)
  - Controllable number of cloud resources
- **Drawbacks**
  - A single plug-in manages both the resource allocation and job submission
  - Special prepared VM image is required
  - Job Manager overhead, e.g.: condor is around 200Mb
  - Nodes are separated from each other
  - ***No way to reschedule job from a node to another***
    - Some nodes might become overcommitted while others are empty

Variant II/b.



The plug-in manages the resources !!!

Jobs are submitted to a frontend node

n Cloud instances maximum

Variant II/b. – Scheduling example

```
start_and_submit:
    if (tj / k > q) and (k<n)
        start_worker()
    submit_job_to_frontend(jobID)

stop:
    if (tj / (k-1) < q) and
        (t < timestamp("-20 minutes"))
    stop_worker()
```

m: number of free instances
k: total number of running instances
n: maximum number of instances
j(x): number of jobs queued on instance x
t: timestamp of last job submission
tj: total number of jobs in the cloud
q: prefered maximum job number per worker

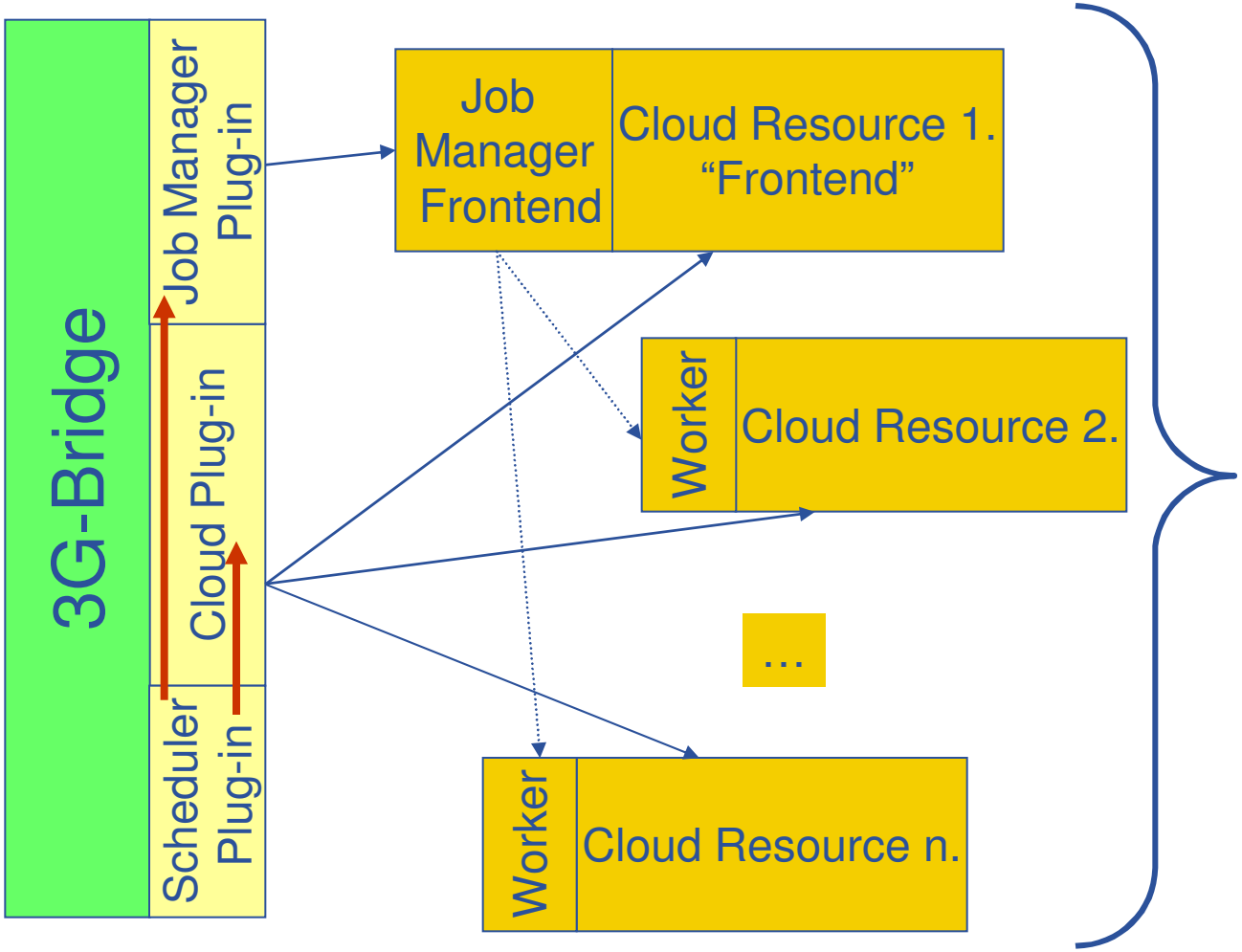
- **The job manager schedules task between its workers**
- **The plug-in needs only to start/stop instances for the job manager**

Variant II/b.

- **Benefits**
  - The job manager balances tasks between the cloud resources
  - No resource gets overcommitted while others starve, task can migrate
  - We can execute parallel (MPI, PVM, Map-Reduce, Hadoop) applications..

- **Drawbacks**
  - Two special prepared VM Images are required
    - One for the workers and one for the Frontend
  - The cloud instances need to communicate between each other
  - Still a single plug-in manages everything

Variant III.



Job manager plug-in handles job submission

Cloud **plug-in** manages **resource allocation**

Scheduler plug-in implements logic to utilize together **job management** and **resource allocation plugins** (can be more of each used together !!!)

Variant III: Benefits and drawback

Benefits

- **Separates job submission, resource management and scheduling logic**
  - Task management plug-in (e.g. Condor) can be used on its own
  - Scheduling logic allows to build a pool of resources (“meta-cloud”) from different clouds and local/remote resources **as long there is a resource allocator/manager plug-in for the resource**
  - Different resource management plug-ins can be implemented
    - E.g.: EC2/Eucalyptus, VM manager
  - The implemented plug-ins can be used at other places, e.g. VM support for BOINC
    - VM resource manager with a task delegation plug-in

Drawback

- **Requires modifications in 3G-Bridge itself**

Current status of implementation

- ***Variant I.* has been successfully implemented**
- **Implementation of *Variant II.* is in progress**
  - Relative little effort to implement
  - Can be simply reworked to get the Resource Management and Task Delegation components for Variant III.
- **We use Eucalyptus Public Cloud**
- **We investigate Condor WebService API for task submission**
  - stable version of Condor (7.4.1, 7.4.0, 7.2.5, etc) segfaults when using the WS API on the Eucalyptus Public Cloud
    - 2 bug reports submitted to Condor developers

Thank you