Optimised scheduling mechanisms for Virtual Machine deployment in Cloud infrastructures

Michael Gienger, HLRS

26th Workshop on Sustained Simulation Performance

Stuttgart, 10th and 11th of October 2017

Agenda

- Background
- Motivation
- Approach
  - General approach
  - Individual building blocks
- Evaluation
- Summary

BACKGROUND

CLOUD COMPUTING IN A NUTSHELL

What is Cloud Computing ?

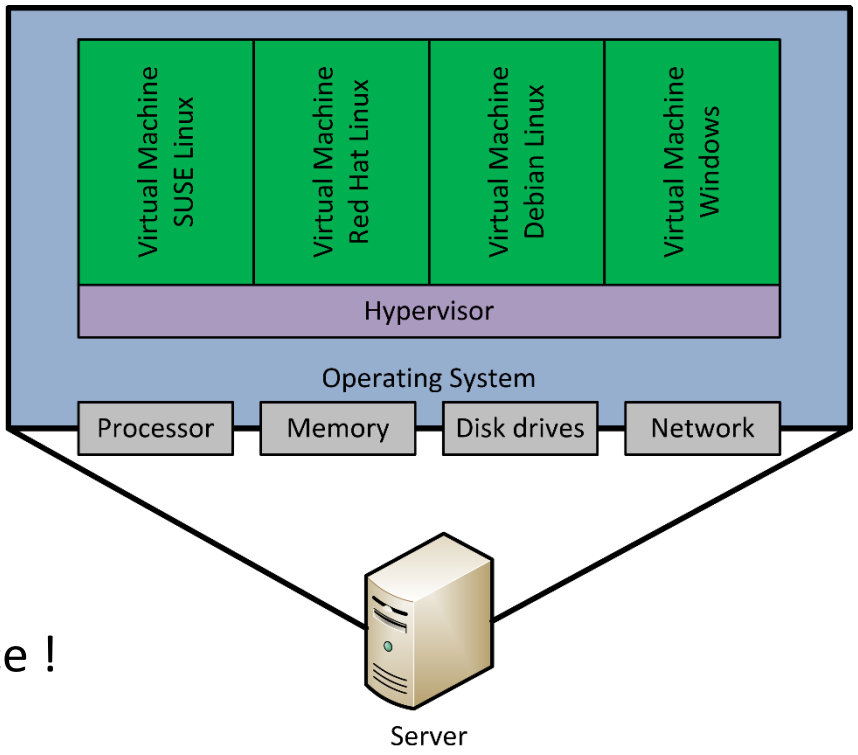
- There is NO common answer !
- The concept IS very popular and flexible
  - Web hosting
  - Data backup
  - Software development
  - Service provisioning
  - and many, many more…
- Cloud Computing IS NOT defined clearly
  - Related areas have jumped on the wagon
  - Rebranded offerings to “Cloud”

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

National Institute of Standards and Technology, 2011

Technological baseline

- Service Provisioning
  - Virtual Machines
    - Flexibility
  - Access
    - Simplicity
  - Resource sharing
    - Availability instead of Performance !
- XaaS
  - Infrastructure as a Service
  - Platform as a Service
  - Software as a Service



MOTIVATION

CLOUD COMPUTING SHORTCOMINGS

State-of-the-Art Cloud issues (i)

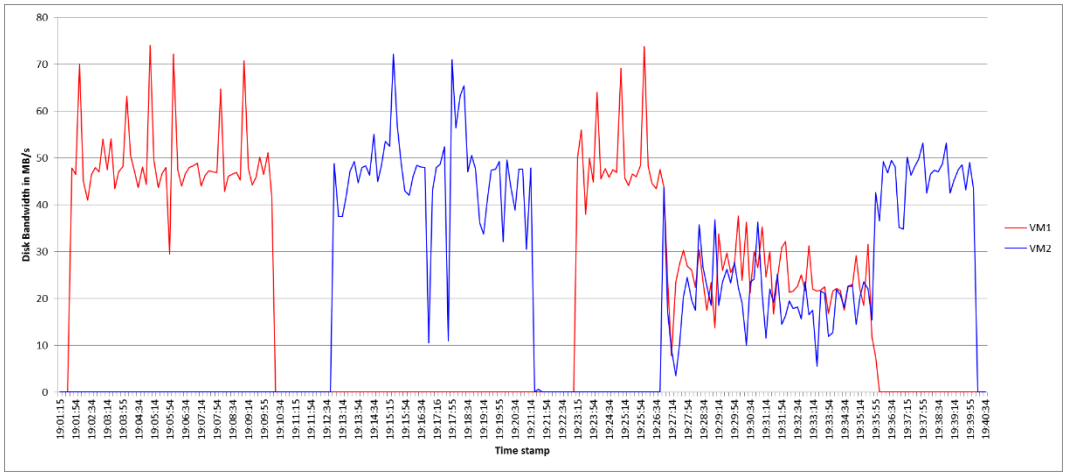
- Virtualisation
  - Performance drop
  - Different hypervisors with different advantages and disadvantages
- Cloud Management
  - All Virtual Machines are treated equally
    - Virtual Machine application domains are not regarded
    - Specific collocation of Virtual Machines is not considered
  - Various, but simple scheduling strategies
    - Random, Memory-bound, Compute-bound, Packing, Striping, First-free, etc.

But there are companies that offer HPC as a Service, with a drawback caused by scheduling and virtualisation technology !

State-of-the-Art Cloud issues (ii)

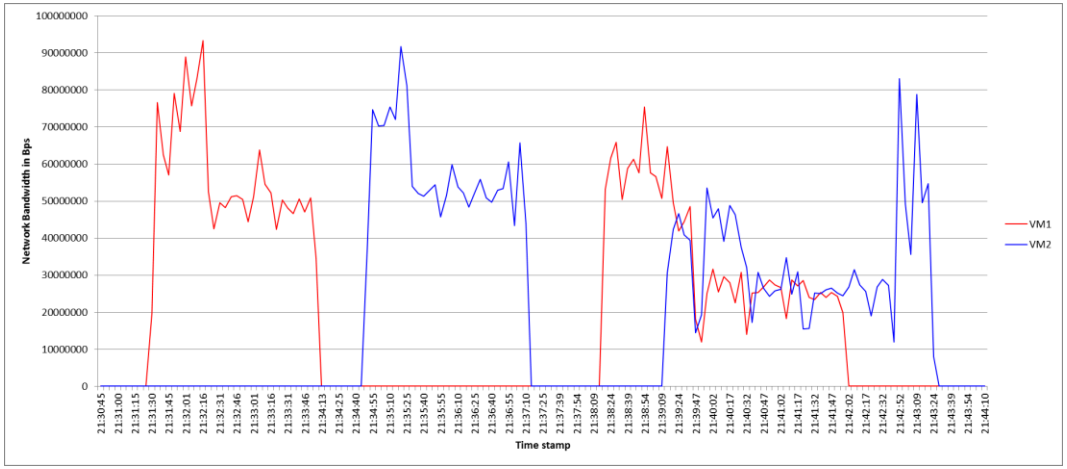
- **Virtual Machines on a single node can influence each other**
  - I/O Benchmark

    Linux *dd*



  - Network Benchmark

    Linux *wget*



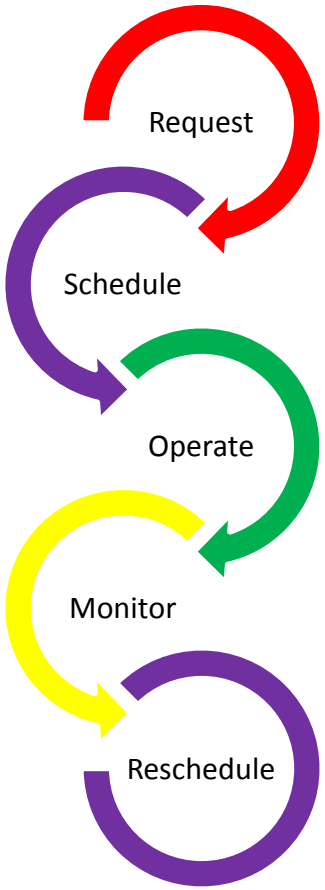
Required improvements

- **Enabling HPCaaS requires optimisation !**
  1. Improve the performance of virtualisation
     - Memory
     - CPU
     - I/O
     - Network
  2. Improve the collocation of Virtual Machines in a Cloud environment
     - Take into account the particular usage of Virtual Machines
     - Collocate contradictory Virtual Machines and their applications

APPROACH

IMPROVE VIRTUAL MACHINE DEPLOYMENT

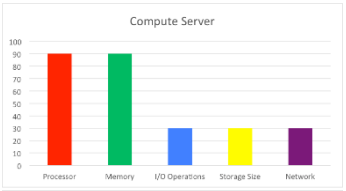
General approach



- **Improved performance relies on sophisticated scheduling strategies**
  - Information about infrastructure usage
  - Virtual Machine application domain
- Infrastructure information needs to be available
  - State-of-the-Art Cloud Middleware builds on CPU and Memory only !
  - I/O, Network, Energy consumption, etc. need to be assessed
- Application profiles are mandatory
  - When booting the Virtual Machine
  - During its lifetime

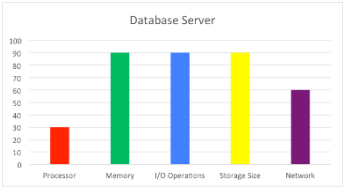
Application characterisation

- Application domains



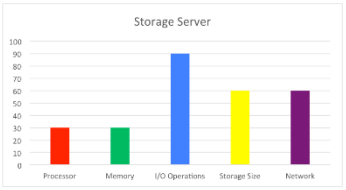
Compute Server

Processor: ↑ Memory: ↑ IO: ↓ Storage: ↓ Network: ↓



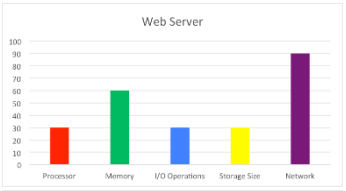
Database Server

Processor: ↓ Memory: ↑ IO: ↑ Storage: ↑ Network: →



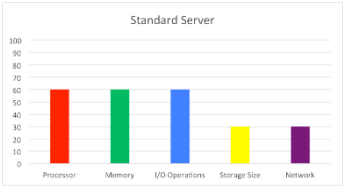
Storage Server

Processor: ↓ Memory: ↓ IO: ↑ Storage: → Network: →



Web Server

Processor: ↓ Memory: → IO: ↓ Storage: ↓ Network: ↑



Standard Server

Processor: → Memory: → IO: → Storage: ↓ Network: ↓

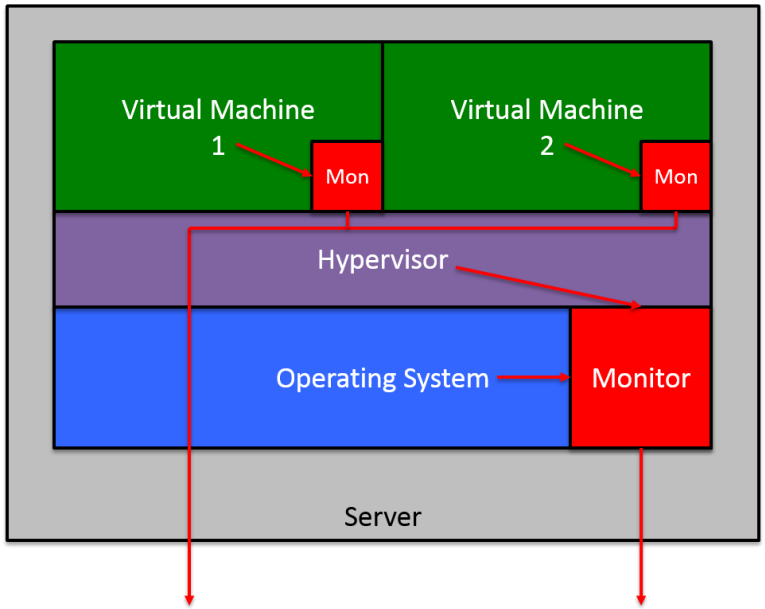
Application requirements retrieval

- Virtual Machine requirements can be obtained twofold
  - **Based on empirical user knowledge and experience**
  - **Automated Virtual Machine profiling**
- Both methods have advantages and disadvantages
  - Empirical knowledge is suitable for initial scheduling, but may be incorrect
  - Profiling requires an initial deployment before an optimal deployment can be determined

Both approaches need to be combined to guarantee suitable **initial**, but also **reactive scheduling** of Virtual Machines !

Infrastructure controlling & monitoring

- Cloud Middleware manages
  - Virtual Machines, Hosts, Networks, etc.
  - Fetches the host information from the hypervisors
- Monitoring information on different levels
  - **Static provisioning: Servers**
    - Infrastructure level
    - Operating system level
      - CPU, Memory, I/O, Network, …
    - Hypervisor
  - **Dynamic provisioning: Virtual Machines**
    - Operating system level
      - CPU, Memory, I/O, Network, …



Cloud scheduling algorithm (i)

- State-of-the-Art algorithms are simple
  - Take into account CPU and Memory of the host
  - Respect generic network or I/O utilisation
    - **But no individual Virtual Machine parameter settings !**
  - Consider particular data centres, zones, clusters and individual hardware
  - Can include manual interaction (such as host selection or blacklists)
- **A more sophisticated algorithm is mandatory**
  - Highly efficient and productive
    - Fast and Robust
    - Correct & Traceable

Cloud scheduling algorithm (ii)

- Describe requirements in conjunction with hardware information in a *M x N* matrix
  - Focus on additional Virtual Machine information
  - Build on deployment costs
    - Quantify and evaluate Virtual Machine costs
    - Normalise values

$$Cost_{VM,Server} = \begin{cases} \infty & Impossible\ combination \\ Cost_{Migration} * (P_1 * Criteria_1 + P_2 * Criteria_2 + \ldots + P_n * Criteria_n) & \\ -Cost_{VM,Server-deployed} & Deployment\ cost \\ 0 & Current\ combination \end{cases}$$

Cloud scheduling algorithm (iii)

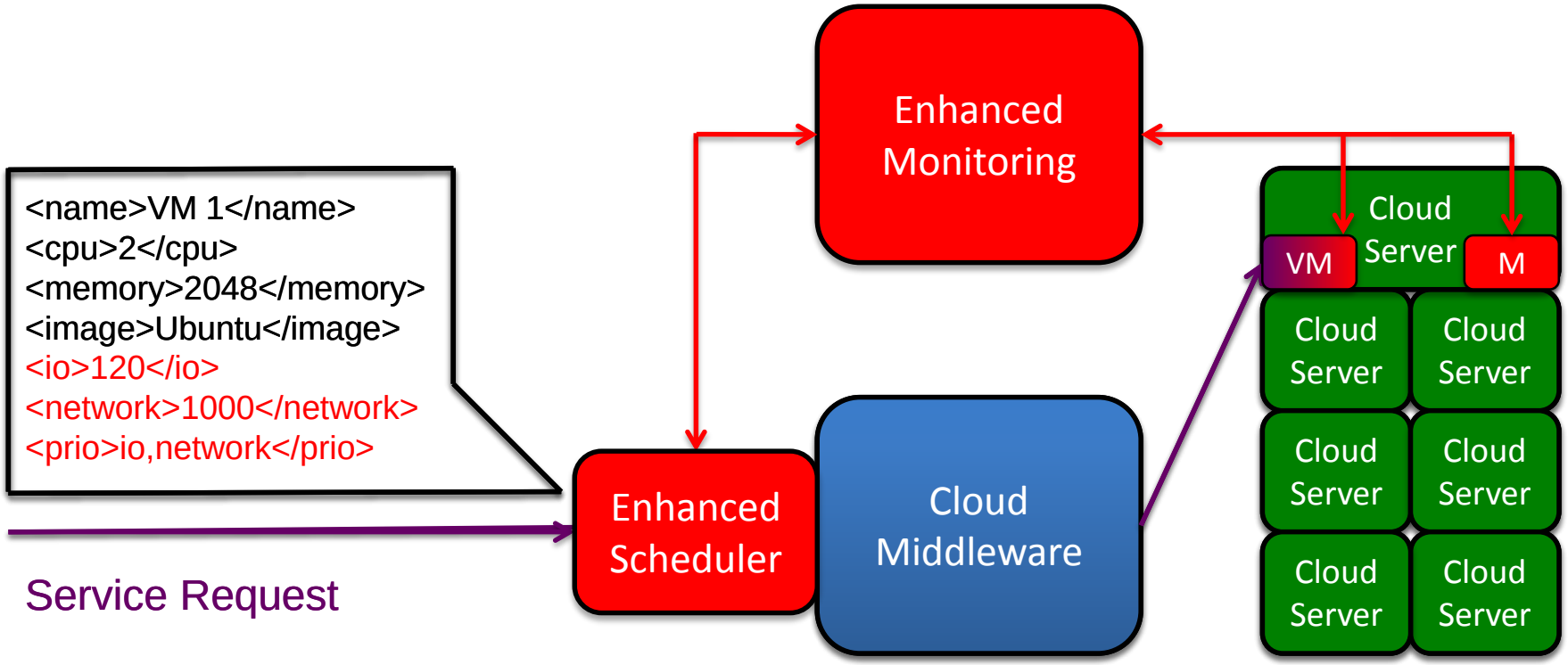
VM / Server combination	Host 1 CPU: 16 cores Memory: 64 GB IO: 100 IOPS Network: 1 Gbps	Host 2 CPU: 24 cores Memory: 128 GB IO: 50 IOPS Network: 20 Gbps
VM 1 CPU: 2 cores Memory: 4 GB IO: 100 IOPS Network: 1 Gbps Priority: IO, Network	**3.1** 3x1 *(100/100)* + 2x0.05 *(1/20)*	∞ (IO)
VM 2 CPU: 1 core Memory: 2 GB IO: 50 IOPS Network: 10 Gbps Priority: Network	∞ (Network)	**2** 3*0.5 *(10/20)* + 1*0.5 *(50/100)*

Cloud scheduling algorithm (iv)

```
I=0;
while VM without Host and I < X do
    while VM parameters allow only one Host do
        select Host;
        update Host with respect to VM.parameters;
    end
    find global minimum;
    select Host;
    update VM costs for all Host cells to ∞;
    update Host with respect to VM.parameters;
    I++;
end
select random Hosts;
update Hosts with respect to VM.parameters;
```

- *O(#VMs x #Hosts x #I)*
- Break point required
  - A perfect solution cannot be guaranteed
  - Random selection of a host
- More complex and error-prone than standard algorithms

Cloud environment integration



Focus on seamless integration, but don‘t break existing infrastructures and their operation models !

EVALUATION

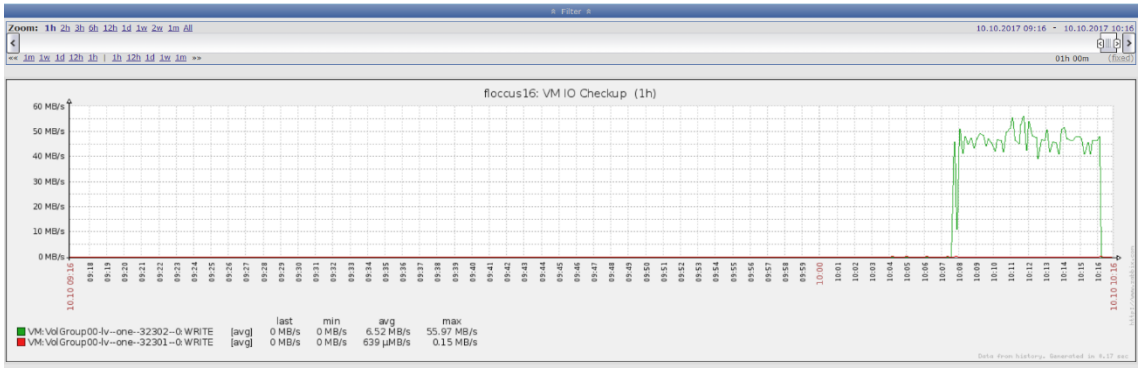
INITIAL RESULTS

Evaluation setup

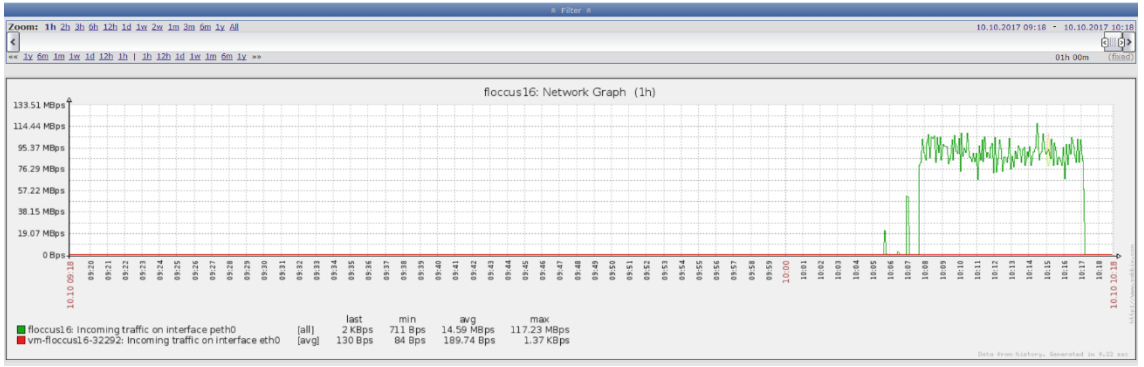
- Cloud infrastructure
  - Cloud controller: OpenNebula
  - Virtual Machine hosts: 2
- Enhanced Monitoring
  - Zabbix Server
  - Zabbix Clients for Servers and Virtual Machines
- Virtual Machines
  - 10 GB size
  - Debian Linux Operating System
  - Synthetic workload benchmarks
  - Deployment based on individual requirements

Collocation of two complementary Virtual Machines (i)

- VM 1
  - I/O intensive
  - Near bare-metal performance



- VM 2
  - Network intensive
  - Near bare-metal performance



Near **native performance** on a single node, although resources are shared (VM packing) !

Collocation of four complementary Virtual Machines (ii)

- VM 1
  - I/O intensive
- VM 2
  - Network intensive
- VM 3
  - CPU intensive
- VM 4
  - Memory intensive

VM	Host	Power consumption
Striping		
VM 1	Server 1	12 W *(VM)* + 125 W *(idle)*
VM 2	Server 2	5 W *(VM)* + 125 W *(idle)*
VM 3	Server 1	60 W *(VM)* + 125 W *(idle)*
VM 4	Server 2	10 W *(VM)* + 125 W *(idle)*
Collocation		
VM 1	Server 1	12 W *(VM)* + 63 W *(idle)*
VM 2	Server 1	5 W *(VM)* + 62 W *(idle)*
VM 3	Server 1	60 W *(VM)* + 63 W *(idle)*
VM 4	Server 1	10 W *(VM)* + 62 W *(idle)*

Drastically reduced energy consumption **by 43 %** compared to VM striping !

SUMMARY

CONCLUSIONS AND FUTURE WORK

Conclusions and Summary

- Clouds build primarily on service availability, not on service performance
  - "Ultimate" performance has not been a key goal
  - **But the application domains open up more and more !**
- Improving the overall Cloud performance is required
  - Support all kinds of services (e.g. HPC as a Service)
- Creating a more efficient scheduling strategy is possible
  - But much more complex
  - And more error-prone
- Seamless integration is important
  - Keep the Cloud approach simple and intuitive !

Future work

- Benchmark the overall setup with real-world workloads
  - Traditional Cloud vs. HPCaaS
  - Evaluate the performance of single applications
  - Quantify the performance of the entire system environment
  - Compare the results against State-of-the-Art scheduling algorithms
- Provide a production-quality implementation
  - Improve the scheduling algorithm performance
- Support different Cloud Middleware

Thank you !

Questions ?

Michael Gienger
High Performance Computing Center Stuttgart
Nobelstrasse 19
70569 Stuttgart
Phone: +49-711-685-63824
Email: gienger@hlrs.de