International Tutorial INFOCOMP / DataSys

# High End Requirements and Practice: Advances in Sciences and Computing

The Fifth International Conference on Advanced Communications and Computation

(INFOCOMP 2015 / DataSys)

June 21–26, 2015, Brussels, Belgium

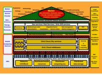
IARIA

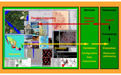
Dr. rer. nat. Claus-Peter Rückemann[1,2,3]

[1] Westfälische Wilhelms-Universität Münster (WWU), Münster, Germany
[2] Leibniz Universität Hannover, Hannover, Germany
[3] North-German Supercomputing Alliance (HLRN), Germany

ruckema(at)uni-muenster.de

GEXI ACT@MAP





# Contents

# Tutorial targets

**Focus:**

- Requirements and implementations of High End Computing resources for advanced scientific computing and resulting from many disciplines and sciences.
- Sustainable means of processing and computing are becoming increasingly important for many disciplines, not only for disciplines, which are spanning long time intervals like geosciences, environmental sciences, and archaeology.
- Illuminating the challenges and processes of computation and implementation also raises questions on the benefits and what the long-term sustainable knowledge and results are when working with advanced and complex application scenarios and what the relative significance of the content is.
- The tutorial shows and discusses real examples of advanced implementations worldwide, introduces in architectures and operation, and tries to discuss consequences and solutions.

# Focus questions

**Some focus questions are:**

- What are the High End Computing and storage requirements?
- How are solutions for requirements implemented in practice?
- What High End Computing system implementations exist in practice?
- Where/what are the emerging challenges?
- What are major demands / motivations / goals from disciplines?
- Are there sustainable data-centric and knowledge-centric long-term approaches?

**It is intended to have a dialogue with the audience on how terms like "long-term", "computing", "knowledge", and "content" may be defined.**

# High End Requirements

**High Performance Computing / Maximum Performance Computing / Supercomputing:**

In High Performance Computing,
supercomputers -i.e., computer systems at the
*upper performance limit of currently feasible processing capacity*-
are employed to solve challenging scientific problems.

# Way (NOT) to go: Hierarchies over experiences

## What others do: "Experts say: Hierarchies are more important than experiences."

Let us take a look on what a virtual, "effective" institution will do.

**"N"ewtoneless**
**"U"niversity**
**"T"echnology**
**"S"ervice**

NUTS' initiative:

- **Buy new high end resources.**
- **Establish glamourous bodies and hierarchies.**
- **Target on procuring the most heterogeneous technologies possible.**

NUTS' strategy:

- **Strictly limit participation to administration when preparing requirements.**
- **Keep away from "best pratice" as everyone can have own best pratice.**
- **Research is only required for justifying technical procurements.**

NUTS' results and recommendations:

- **Short-term scientific results do not require longer research.**
- **Any topic is research.**
- **Focus on promoting the lifted benefits only.**
- **There are no drawbacks – any propagated are from evil doers.**

# Demand for High End Computing and storage resources

**Demands and Motivation: . . . are an important tools for**

- Natural sciences, engineering, social sciences and many others,
- interactive and batch use,
- Electronic Data Processing (EDP): natural sciences, geophysics (seismology, seismics, hydrocarbon geology, physics of the ionosphere etc.) array processing, data copies, memory usage, higher programming languages, batch system, scripting, visualisation (2D, 3D, 4D, . . .),
- Environmental research,
- Scientific Information Systems / Geoscientific Information Systems (GIS),
- Spatial Information Systems,
- Knowledge processing and knowledge discovery,
- . . .

# High Performance Computing / Advanced Scientific Computing

## Overview

- Requirements
  - Fast Central Processing Unit (CPU).
  - Parallel processing.
  - Large memory.
  - Fast Input/Output (I/O).
  - Powerful communication / networks.
- Hardware / resources
- System / software / configuration
- Applications
- Configuration, optimisation, scaling, . . .

## Alternatives?

- High Performance Computing.
- Cluster computing.
- Grid Computing.
- Cloud Computing.

# Parallel computing and the beginning

## von Neumann Arithmetic Logic Unit (ALU):

- Floating point arithmetic, integer arithmetic and I/O possible in parallel,
- Instruction Look Ahead, command-cache (German: Befehlscache),
- Memory Interleaving, that means separate access for address-neighboured Bytes/Bits, e.g., per memory chip there will always only 1 Bit be stored,
- Pipelining referring to command sequence execution (e.g., RISC/Look Ahead),
- further increase of performance by using more CPU.

## Classic taxonomy (Michael Flynn, 1966): Classification instructions/data

Regarding data and instruction stream there are four different types of parallel systems:

| | |
|---|---|
| **SISD** | Single Instruction Single Data<br>*Classical architecture, processor internal parallel, pseudo-parallel* |
| **SIMD** | Single Instruction Multiple Data<br>*Parallel on statement level* |
| **MISD** | Multiple Instruction Single Data |
| **MIMD** | Multiple Instruction Multiple Data<br>*Parallel on program level, SPMD - parallel property of data . . .* |

# Parallel computing: Software

**Different levels can be distinguished on software level:**

- Job: Whole jobs run parallel on different processors. With this scenario there is no or little interaction between the jobs. Results are better computer utilisation and shorter real runtimes. (Example: workstation with several processors and multitasking).
- Program: Parts of a program run on multiple processors. Results are shorter real runtimes. (Example: parallel computer).
- Command: Parallel execution between the phases (instructions) of command execution. Result is accelerated execution of the whole command. (Example: serial computer / single processors).
- Arithmetic, Bit-level: Hardware-parallel of integer arithmetics and Bit-wise parallel, but not necessarily word-wise serial access on memory or vice versa. Result is less clock cycles for working an instruction.

The levels of parallel computing given here can occur in combination, too.

# Parallel computing: Hardware

**Different levels can be distinguished on hardware level:**

Pipelining: Segmentation of operators which are worked consecutively (relevant for vector computers).

Functional units: Different functional independent units for working on (different) operations, e.g., super scalar computers can execute additions, multiplications, and logical operations in parallel.

Processor arrays: Arrays of identical processor elements for parallel execution of (similiar) operations. Example: MasPar computer with 16384 relatively simple processors, systolic arrays for image processing.

Multi processing: Several independent processors with own instruction sets each. Parallel execution is possible up to whole programs or jobs.

# Classification, memory access

**Memory access:**

Shared Memory (competing processes): Memory which is accessed by various processes "concurrently". This means they share the memory. Each process can have access all the data. A serial program can easily run under this circumstances. If there is a small number of processors this can show a first performance increase. Another aspect is the increase of access conflicts (e.g., bank conflicts) with larger numbers of processors. The result is that scalability (here that means performance with number of processors) is not warranteed anymore. For reducing access conflicts very performant bus systems and access managements are necessary. This increases the price of the system, too.

Distributed Memory (communicating processes): Distributed Memory consists of memory parts, which can only accessed from one process over the whole program run. If data from other processes is needed, than this can only be handled by explicit communication between the processes.

# Synchronisation of parallel processes

## Synchronisation of parallel processes:

**Synchronisation:** Prevention of undefined states. This can be achieved with various techniques for syntonising the processes, at various point of time.

**Barrier:** Point inside the program, which all processes have to pass through. A Barrier guarantees, that the single processes wait until all processes in this program part have reached this point. This is e.g., important for having different velocities.

**Deadlock:** Multiple processes each waiting on an event, which can only be triggered by one of the waiting processes. (German: “Verklemmung”).

**Semaphor:** A signal, which is not operated from some central instance, but from single processes. A general problem with accessing common resources: Deadlock inexistence. Semaphores (railway signal in Holland) have been introduced on suggestion of Dijkstra.

# Filesystems

**Filesystems:**

| Filesystem type | Examples |
|---|---|
| **Distributed** | NFS, AFS, NCP, CIFS/SMB, XtreemFS, Ceph, Btrfs, HDFS, Tachyon, ... |
| **Shared** | SAN, CXFS, GFS, Polyserve, StorNext FS, QFS, ... |
| **Parallel** | GPFS, Lustre, PVFS, IBRIX, OneFS, PanFS, NFS/pNFS, ... |

# Utilisation and implementation

## Message Passing Interface (MPI):

- MPI (Message Passing Interface) is a Message Passing library specification.
- MPI is not a programming language.
- Programming interfaces exist for example for C, C++, Fortran77 and Fortran90.
- MPI is suitable for parallel computers (SMP systems), Cluster and heterogeneous networks.
- MPI is quite extensive (much more than over 100 functions).
- MPI is small (6 function are often enough, in order to write efficient programs.
- With MPI the same program is executed on all nodes, but not neccessarily the same program code.

# Message Passing Calls

## Message Passing Interface (MPI), Types of Calls:

MPI procedures have the following terminology:

**local:** If the termination of a procedure depends only from the locally executed process. There is no explicit communication with another process necessary such operations. MPI calls which generate local object or do requests for the status of local objects are called "local".

**non-local:** If the termination of a procedure requires the execution of a MPI procedure in another process. MPI communication is often non-local.

**blocking:** If stepping back from a procedure signals, that the user can reuse the resources used in the procedure call. Any visible change of state of a calling process after a blocking call happens before the stepping back from the blocking procedure is done.

**non-blocking:** If it is allowed to step back from a procedure before the operation, which is triggered by calling the procedure, has terminated and before the user can reuse the resources, e.g., buffers, that have been specified in the call. For example, a non-blocking call can start a receive operation, but the message will earliest be accepted after the calling procedure has been terminated.

**collective:** If all processes in a process group have to call the same procedure.

# SMP, MPP, MPI . . .

## Architecture

- SMP: Symmetric Multi-Processing.
- MPP: Massively Parallel Processing.
- MPI: Message Passing Interface, `http://www-unix.mcs.anl.gov/mpi/index.htm`.
- OMP: OpenMP, „open“ implementation, SMP/MPI, `http://www.openmp.org/`.
- MPICH: MPICH Implementierung, `http://www-unix.mcs.anl.gov/mpi/mpich/`.
- Hybrid: MPI/OpenMP.

. . .

# Beyond MPI and friends: Partitioned Global Address Space

## Partitioned Global Address Space (PGAS) and PGAS Models:

MPI communication will not be sufficient anymore for Exascale systems.

**PGAS:** Partitioned Global Address Space.
`http://www.pgas.org`

**GASPI:** Global Address Space Programming Interface.
GASPI is a PGAS API. It uses a SPMD model.
Currently it is in discussion for Petascale and Exascale systems.

`http://www.gaspi.de/en/project.html`

The three main targets are:

- Scalability,
- Flexibility,
- Fault tolerance.

# Architecture and implementation

## Implementation and components

- Hardware / Computing.
  - MPP (Massively Parallel Processing). — MPP compute nodes
  - SMP (Symmetric Multi-Processing). — SMP compute nodes
- System software.
  - Operating systems. — Login server, admin server
  - Cluster management. — Management server
  - Storage management. — Storage server
  - File management. — File server
- Networks.
  - InifiniBand for I/O.
  - InifiniBand for Message Passing Interface (MPI).
  - NumaLink, Aries, . . .
  - Service networks.
- Parallel filesystems (Lustre). — MDS server, OSS server
- Batch system, scheduling, load balancing. (Moab, Torque, . . .). — Batch server
- Accounting . . .
- Data handling, archive / backup. — Archive / backup server
- Optional Grid, Cloud services level.

# Networks: complexity and security

## Networks: complexity and security

- MPI (InfiniBand) (e.g., access, batch system, scheduling, interactive use of nodes)
- IO (InfiniBand) (e.g., access, batch system, scheduling, interactive use, file systems)
- Ethernet (common ethernet issues)
- Admin (e.g., ssh keys, routing)
- Service (e.g., ssh keys, routing)
- special "links" (e.g., HLRN-Link 10 GbE over 320 km)
- ...

# Configuration management

## Why: Without configuration engine

- lots of scripts and makefiles,
- file deployment via cp/rsync,
- complex structure,
- no standard.

## Why: Without revision control:

- single shared global repository,
- no history/log and poor rollback,
- no coordinated access to repository.

## With revision control, more complex but:

- multiple administrators (system, service),
- concurrent access,
- inhomogeneous mix of architectures and different operating systems,
- multiple services (pbspro, sge, globus, deisa, unicore, batch, login, lustre, ...),
- abstraction of regular system administration tasks (copy/edit/monitor files, run commands, ...).

# Configuration management: Implementation and security

## Example: Cfengine / Subversion (SVN) and security:

- **Decoupling:**
  *Decoupled user commit and internal synchronization*,
  *prevents from DoS attacks*,
  *reduces load*,
- **Repository Access:**
  *secured via public-key authentication and SSH tunnel*,
  *each user and repository gets his/her own public/private keypair*,
  *for each repository one SVN tunnel*.
- **Maintenance/Security:**
  *SVN/Cfengine server is controlled via Cfengine*, no management login required.
- **User management:**
  *generate private/public keypair*, add public key to `authorized_keys` file, deploy keys to user.
- **Project management:**
  add repository name to configuration file, Cfengine triggers a script which creates all necessary components, new repository includes a Cfengine template (optional).
- **Redundancy and failover:**
  services may be split across multiple hosts, tricky to implement for SVN.
- **Further aspects:** virtualisation, hardware requirements, Xen host . . .

# Future of High Performance Computing and parallel computing

## Outlook on some pro an con:

**On the one hand:**

- development in micro electronics and technology,
- higher integration density of micro circuits and chips,
- higher clock rates,
- ...

**On the other hand:**

- Signal velocity is limited: $3 \times 10^8$ m/s (e.g., see the typical construction of older Cray),
- technological problems with reducing sizes of chip structures (currently $< 25\mu$m) (reproduction of chip masks),
- energy density,
- background effects, disturbancies,
- quantum effects,
- atomic size $10^{-10}$ m is the last limitation, that cannot be overcome (so far),
- classical von Neumann architecture will soon reach its limitations,
- increase of computing capacities currently only with parallel processing,
- increasing amount of system complexity on management and software level ...

# Tender Process – How Requirements are Currently "Considered"

## Multi-step cycle of 4-7 years:

### Requirements:

- **Users / disciplines**
  $\Longrightarrow$ request users / disciplines for comments.
- **Infrastructure**
  $\Longrightarrow$ participate infrastructure planners, architects, administration, etc.
- **Legal regulations (non-discrimination / environment / procedures)**
  $\Longrightarrow$ participate lawyers.
- **Technical developments**
  $\Longrightarrow$ information from developers and industry.
- **Future planning**
  $\Longrightarrow$ participate hierarchy.
- ...

**This should be drastically improved by PARTICIPATING experience and knowledge, practically experienced auditing, on-topic users, developers, and industry ...**

# Comparison of High End Systems

**Can High End Systems be compared seriously? Remember:**

- Every HEC / Supercomputing system is unique in it's overall hardware, software stack, and configuration.
- Development cyle is about 5 years.
- Most tests for the bleading edge components have to be done on final, entire systems.

**Extraordinary With Singular Aspects: The Greatest, Biggest, Greenest**

Top500: Top500 list with the "fastest" supercomputers in the world. `http://www.top500.org`. Only standard-benchmark: High Performance Linpack (HPL). (2012-11 Blue Waters/NCSA system opts out of Top500 list due to Linpack.)

Green500: "Ecological" list going for performance in relation to energy consumption. `http://www.green500.org`. Only energy and only in operation.

Graph500: `http://www.graph500.org`.

...

# Complex Systems

**Supercomputing Resources – Examples**

For the further dialog within the tutorial, the tutorial discusses some selected historical and up-to-date High Performance Computing systems and hardware and components used with Advanced Scientific Computing.

- Cray2, JUMP, BSC, Shenzhen, Jaguar, Tianhe, Sequoia, Titan, German supercomputing (HLRB, SuperMUC, JUQUEEN, HLRN, and others) ...
- ⇒ Supercomputing and big data
- ⇒ Operation and infrastructure transition phases
- ⇒ Infrastructures, networks, and architectures
- ⇒ Major long-term and sustainability issues with infrastructures
- ...
- **(All existing supercomputing resources are "individuals" – and different.)**

```
----------------------------------------------------
-- ABOVE EXAMPLES AND OTHER MATERIAL FOR DISCUSSION --
----------- ORIGINALLY ON FOLLOWING PAGES -----------
------------------ LEFT OUT HERE --------------------
```

# Most Prominent Problem: Quantity

**Handling Quantity**

- Encryption, IO, PCI ... on Chip
- Error Correction (ECC ...)
- Research and Development
- Scientific and academic staff and support
- Maintenance
- Operative and administrative staff
- Secondary dependencies: energy resources

# Most Prominent Problem with Quantity: Consumption

**Power and Energy Measures**

- Low Voltage memory (LV DIMM)
- Light Load Efficiency Mode (LLEM) multiple Power Supplies
- Watercooler chasis & air conditioning
- Higher temperature cooling
- Hybrid cooling systems
- Energy and Power Manager (Active Energy Manager AEM ...)
- Application/energy frequency optimisation
- Energy reduced low frequency Processors
- Power Management
- Energy Management
- ...

# Expertise, infrastructures, and other challenges

**Summary**

- Experience, expertise, quantities, and qualities are closely linked – which counts exponentially when it comes to high performance.
- Infrastructures at the high end of computing are challenges (large electrical installations, providing clean room conditions, maintaining and pampering infrastructures, air conditioning, liquid cooling, . . .)
- Purposes: Continuous service operation for users, high availability of resources, minimisation of computing downtime, minimisation of service interrupts, minimisation of time to solution, fostering the reputation of resources, . . .
- Users should not be bothered by the resources' and services' infrastructure challenges or consequences in order to focus on their disciplines, results, data, and computation.

# Disciplines and sample fields

**Fields of demand:**

- Geophysics, Geosciences, Particle Physics, Cosmology, . . .
- Environmental Sciences, Ocean Modelling, . . .
- Engineering, Computational Mechanics, Computational Fluid Dynamics, Material Sciences, . . .
- Life Sciences, Computational Chemistry, Biology . . .

**Examples:**

- Seismic Processing, Knowledge Discovery, Molecular Dynamic Structure Analysis, Quantumchemical Simulation, Laminar-Turbulent Transition, Flow Fields, Solar Convection Modelling, Chemical Reactions, Ab-Initio Simulations, 3-D Simulation, Calculation of the Decay, Calculation of Heavy Quark Masses, Climate Modelling, Sound Propagation of Machinery, Hydrodynamics, Global Climate System Effects, Quantum Chromo Dynamics, Molecular Dynamics Simulations, CFD Engineering, Heat Flow Calculation, Aerodynamics, Molecular Dynamics Simulations, Protein Decomposition, Ecosystem Modelling, Simulation of Atmospheres, Calculation of Metal Structures, Laser Material Processing, Sedimentary Modelling, . . .

# Sciences and disciplines

**Statements from knowledge-and-IT experts:**

- **"Persistent data are alpha and omega of scientific research and beyond."** *Dr. Friedrich Hülsmann, Gottfried Wilhelm Leibniz Bibliothek (GWLB) Hannover, Germany, Knowledge in Motion (KiM) long-term project, DIMF.*
- **"Intelligently structured digital long-term resources can help protect against colateral damages to knowledge such as mankind experienced from the destruction of the library of Alexandria."** *Dipl.-Biol. Birgit Gersbeck-Schierholz, Leibniz Universität Hannover, Germany, Knowledge in Motion (KiM) long-term project, DIMF.*
- **"Content is the primary long-term target and value and we need powerful and secure information technology to support this on the long run."** *EULISP post-graduate participants, European Legal Informatics Study Programme, Leibniz Universität Hannover, Germany.*

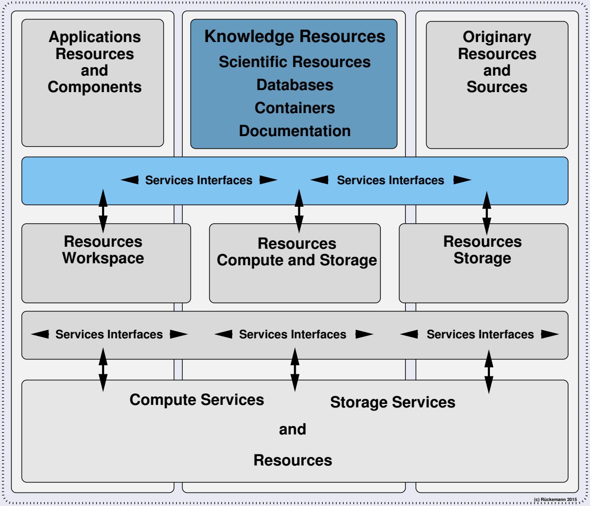
# User perspective on computing resources and tools

**Can user/groups easily overview and handle "their" issues:**

- Computing, heterogenous resources and configuration?
- Code porting and handling?
- Efficient programming (parallelisation, optimisation, scripting)?
- Data locality, porting, and optimisation?
- Input/output requirements and analysis?
- Memory requirements and analysis?
- Network requirements and analysis?
- Checkpointing on applications?
- Resources policies and exceptions?
- Functional archiving restrictions?
- Data long-term issues?
- Library issues?
- ...

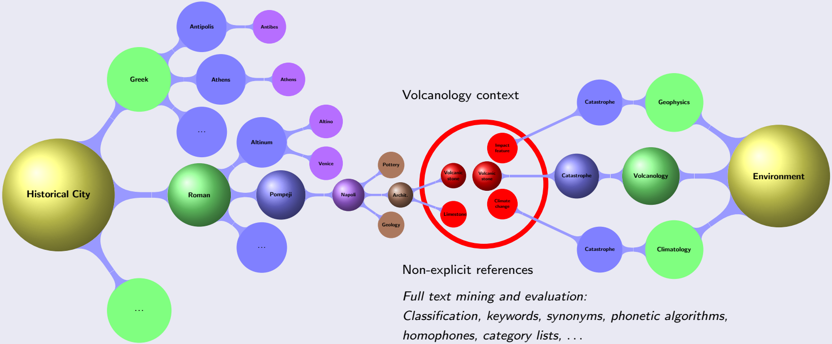
# Example Long-term Architecture, Implementation, and Resources

## Long-term architecture: Central component: Knowledge resources



# Knowledge Discovery Example: Computing object carousel connections

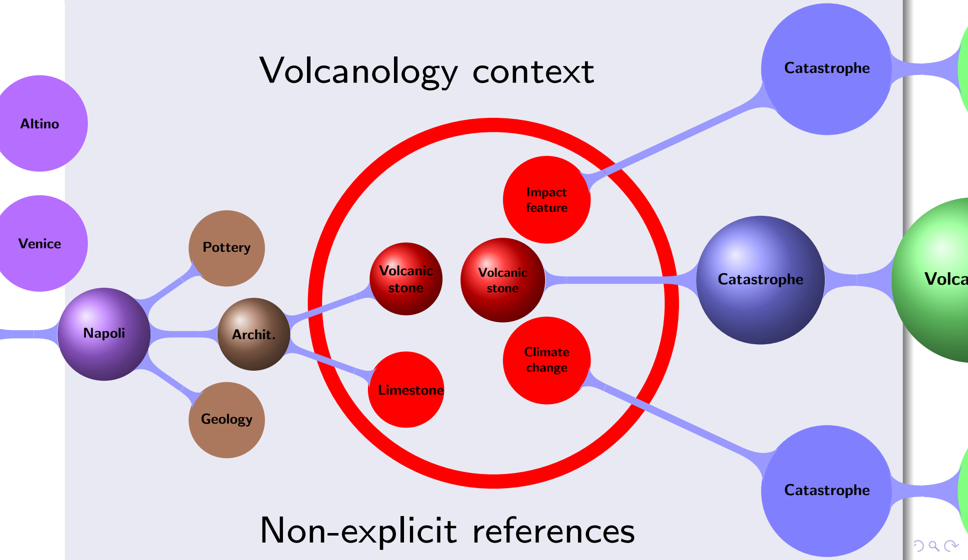
## Historical city and environment object carousels, trees with computed references




Carousel links, calculated via non-explicit references of comparable objects (red) from knowledge resources within trees. Starting topics are identified by large golden bullets. The two fitting lines within the object carousels are `Historical City : Roman : Pompeji : Napoli : Architecture : Volcanic stone` and `Environment : Volcanology : Catastrophe : Volcanic stone`. Fitting object term for historical city and environment is `Volcanic stone`. Excerpt of associated multi-disciplinary branch level objects: `Limestone`, `Impact feature`, `Climate change`.

# Knowledge Discovery Example: Computing object carousel connections

## Historical city and environment object carousels, trees with computed references



# Conclusions and Lessons Learned

## High End Resources / Sciences, Computing, Content:

- Computing resources: Computing resources and infrastructures are too non-sustainable, too volatile, too heterogeneous, porting is too time consuming, automation is too error prone.

# Conclusions and Lessons Learned

## High End Resources / Sciences, Computing, Content:

- Computing resources: Computing resources and infrastructures are too non-sustainable, too volatile, too heterogeneous, porting is too time consuming, automation is too error prone.
- Sciences' requirements: Long-term requirements of sciences, disciplines, and groups are not reflected by common computing resources.

# Conclusions and Lessons Learned

## High End Resources / Sciences, Computing, Content:

- Computing resources: Computing resources and infrastructures are too non-sustainable, too volatile, too heterogeneous, porting is too time consuming, automation is too error prone.
- Sciences' requirements: Long-term requirements of sciences, disciplines, and groups are not reflected by common computing resources.
- Knowledge: There is need for long-term knowledge resources, multi-disciplinary documentation and decision making.

# Conclusions and Lessons Learned

## High End Resources / Sciences, Computing, Content:

- Computing resources: Computing resources and infrastructures are too non-sustainable, too volatile, too heterogeneous, porting is too time consuming, automation is too error prone.
- Sciences' requirements: Long-term requirements of sciences, disciplines, and groups are not reflected by common computing resources.
- Knowledge: There is need for long-term knowledge resources, multi-disciplinary documentation and decision making.
- Best Practice: Funding for computing resources as well as for long-term knowledge creation is (still) not based on best practice and sustainable personalised funding.

# Future Challenges

**Following events:**

**How can sciences and long-term multi-disciplinary work drive resources' development?**

# Future Challenges

## Following events:

**How can sciences and long-term multi-disciplinary work drive resources' development?**

## Overall goals:

- **Foster best practice for management processes and funding, improve decision making processes.**
- **Foster the long-term creation of knowledge and improve the Quality of Data.**
- **Foster multi-disciplinary documentation and work.**
- **Where we are: Heterogeneous resources, content, data,...**
- **Mid- and long-term: Improve the positions and roles of scientific experts over administrative heads.**
- **Where we go: Advanced computing tools – sustainable, long-term, reliable, efficient, robust, automatable.**

# Follow-up topics at this years' conference

## Panel:

- **Tuesday, 2015-06-23, 13:45 – 15:30**
  **INFOCOMP International Expert Panel:**

  **Emerging Solutions in Scientific and High End Computing: Coping with Challenges and Requirements on the Long-term**

  Program: http://www.iaria.org/conferences2015/ProgramINFOCOMP15.html

- **Wednesday, 2015-06-24, 10:30 – 12:15**
  **INFOCOMP 4–Session, Discussion on:**

  **Creation of Objects and Concordances for Knowledge Processing and Advanced Computing.**

  Program: http://www.iaria.org/conferences2015/ProgramINFOCOMP15.html

- **Thursday, 2015-06-25, 13:45 – 15:30**
  **International Expert Panel:**
  **Future Technologies / Urban / Empirical**

  **From Today's to Tomorrow's Technologies: The Winners are . . .**

  Program: http://www.iaria.org/conferences2015/ProgramINFOCOMP15.html

# References

**References and acknowledgements, see:**

⇒ C.-P. Rückemann, "Creation of Objects and Concordances for Knowledge Processing and Advanced Computing," in *Proceedings of The Fifth International Conference on Advanced Communications and Computation (INFOCOMP 2015), June 21–26, 2015, Brussels, Belgium*. XPS Press, 2015, ISSN: 2308-3484, ISBN-13: 978-1-61208-416-9, pp. 91–98, URL: http://www.thinkmind.org/index.php?view=instance&instance=INFOCOMP+2015 [accessed: 2015-06-21], http://www.iaria.org/conferences2015/ProgramINFOCOMP15.html [accessed: 2015-06-21].

⇒ C.-P. Rückemann, "Fundamental Aspects of Information Science, Security, and Computing," 2007–2015, *(Univ. Lectures). ISSC, EULISP Lecture Notes, European Legal Informatics Study Programme*. Institut für Rechtsinformatik (IRI), Leibniz Universität Hannover, URL: http://www.eulisp.org [accessed: 2015-06-21].

# Networking

**Thank you for your attention!**

**Wish you an inspiring conference and a pleasant stay in Brussels!**