

# High Performance Computing

ADVANCED SCIENTIFIC COMPUTING

**Prof. Dr. – Ing. Morris Riedel**

Adjunct Associated Professor

School of Engineering and Natural Sciences, University of Iceland, Reykjavik, Iceland

Research Group Leader, Juelich Supercomputing Centre, Forschungszentrum Juelich, Germany

LECTURE 1

[in @Morris Riedel](#)

[@MorrisRiedel](#)

[@MorrisRiedel](#)

## High Performance Computing

September 5, 2019

Room V02-258



UNIVERSITY OF ICELAND  
SCHOOL OF ENGINEERING AND NATURAL SCIENCES  
FACULTY OF INDUSTRIAL ENGINEERING,  
MECHANICAL ENGINEERING AND COMPUTER SCIENCE



**JÜLICH**  
Forschungszentrum

JÜLICH  
SUPERCOMPUTING  
CENTRE



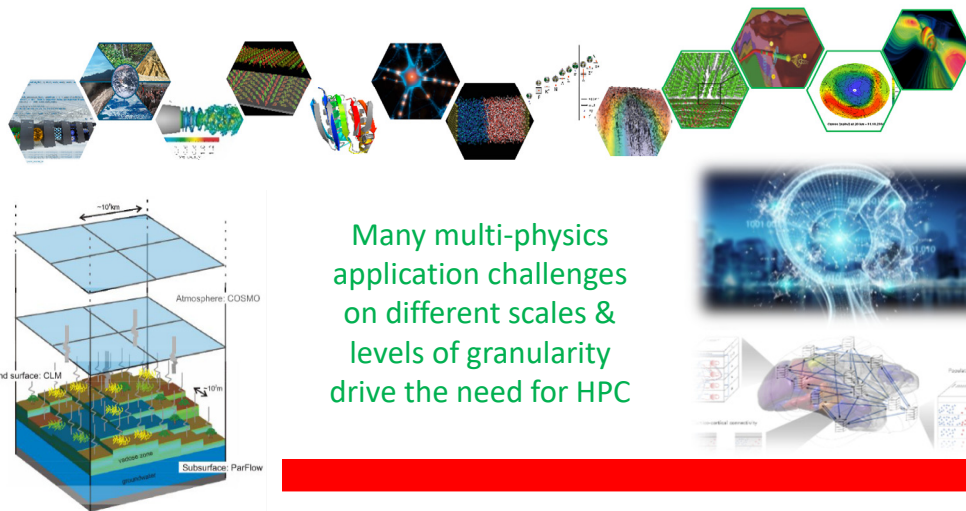
**HELMHOLTZ**  
RESEARCH FOR GRAND CHALLENGES



HELMHOLTZ  
ARTIFICIAL INTELLIGENCE  
COOPERATION UNIT

# Review of Practical Lecture 0.2 – Short Intro to C Programming & Scheduling

- Many C/C++ Programs used in HPC today



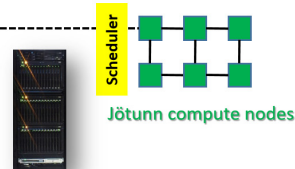
[1] Terrestrial Systems SimLab

not good!

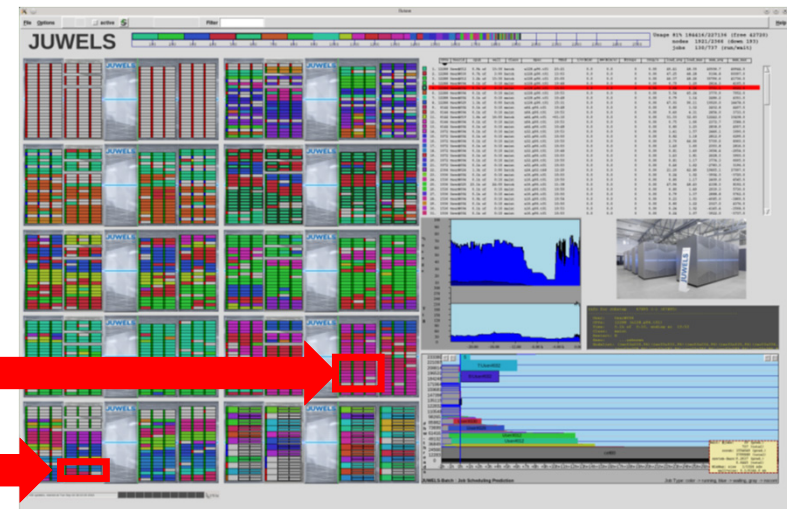
```
[morris@jotunn 2019-HPC-Course]$ ./hello
Hello World! [morris@jotunn 2019-HPC-Course]$
```

[2] NEST Web page

right way!



- Multi-user HPC usage needs Scheduling

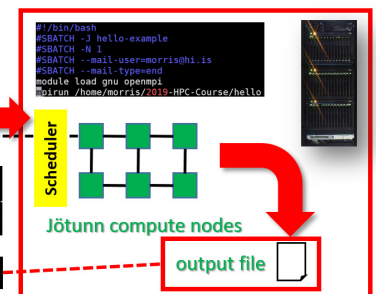


[3] LLview

```
[morris@jotunn 2019-HPC-Course]$ sbatch submit-hello.sh
Submitted batch job 198744
```

```
[morris@jotunn 2019-HPC-Course]$ qstat
Job id      Name      Username    Time Use S Queue
-----
198743      hello-example morris      00:00:00 C normal
198744      hello-example morris      00:00:00 C normal

[morris@jotunn 2019-HPC-Course]$ more slurm-198744.out
Hello World!
```



# Outline of the Course

## 1. High Performance Computing

2. Parallel Programming with MPI
3. Parallelization Fundamentals
4. Advanced MPI Techniques
5. Parallel Algorithms & Data Structures
6. Parallel Programming with OpenMP
7. Graphical Processing Units (GPUs)
8. Parallel & Scalable Machine & Deep Learning
9. Debugging & Profiling & Performance Toolsets
10. Hybrid Programming & Patterns

11. Scientific Visualization & Scalable Infrastructures
12. Terrestrial Systems & Climate
13. Systems Biology & Bioinformatics
14. Molecular Systems & Libraries
15. Computational Fluid Dynamics & Finite Elements
16. Epilogue

+ additional practical lectures & Webinars for our hands-on assignments in context

- Practical Topics
- Theoretical / Conceptual Topics

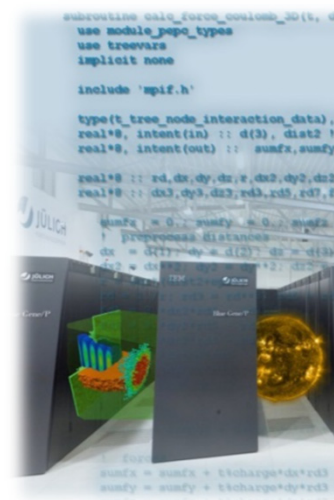
# Outline

- High Performance Computing (HPC) Basics
  - Four basic building blocks of HPC
  - TOP500 & Performance Benchmarks
  - Multi-core CPU Processors
  - Shared Memory & Distributed Memory Architectures
  - Hybrid Architectures & Programming
- HPC Ecosystem Technologies
  - HPC System Software Environment – Revisited
  - System Architectures & Network Topologies
  - Many-core GPUs & Supercomputing Co-Design
  - Relationships to Big Data & Machine/Deep Learning
  - Data Access & Large-scale Infrastructures



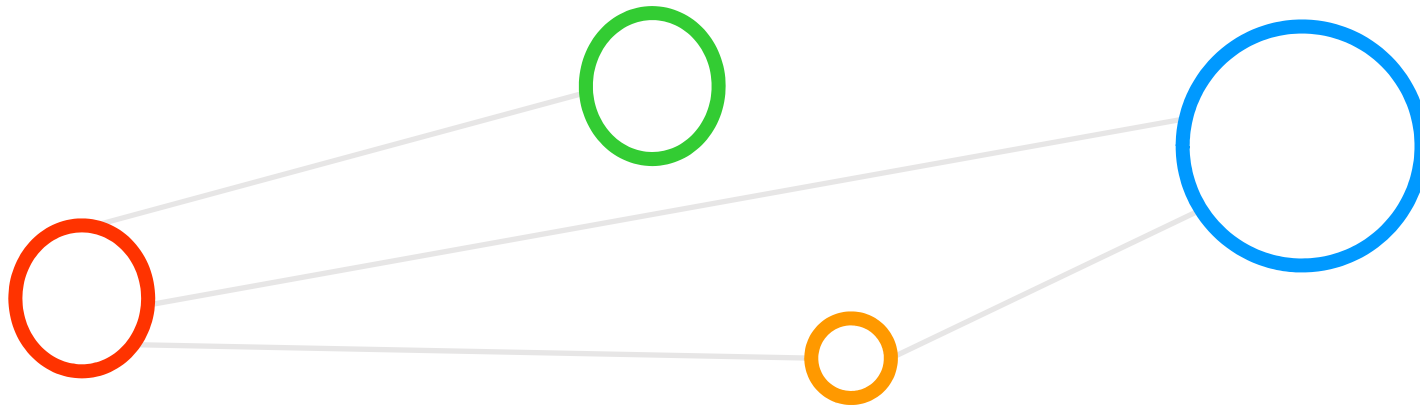


## Selected Learning Outcomes

- Students understand...
    - Latest developments in parallel processing & high performance computing (HPC)
    - How to create and use high-performance clusters
    - What are **scalable networks & data-intensive workloads**
    - The importance of **domain decomposition**
    - **Complex aspects of parallel programming**
    - **HPC environment tools** that support programming or analyze behaviour
    - Different abstractions of **parallel computing on various levels**
    - Foundations and approaches of **scientific domain-specific applications**
  - Students are able to ...
    - Programm and use HPC programming paradigms
    - **Take advantage of innovative scientific computing simulations & technology**
    - Work with technologies and tools to handle parallelism complexity
- 



# High Performance Computing (HPC) Basics



# What is High Performance Computing?

- Wikipedia: ‘redirects from HPC to [Supercomputer](#)’
  - Interesting – gives us already a hint what it is generally about

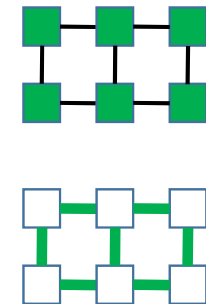
■ A supercomputer is a computer at the frontline of contemporary processing capacity – particularly speed of calculation

[4] Wikipedia ‘Supercomputer’ Online



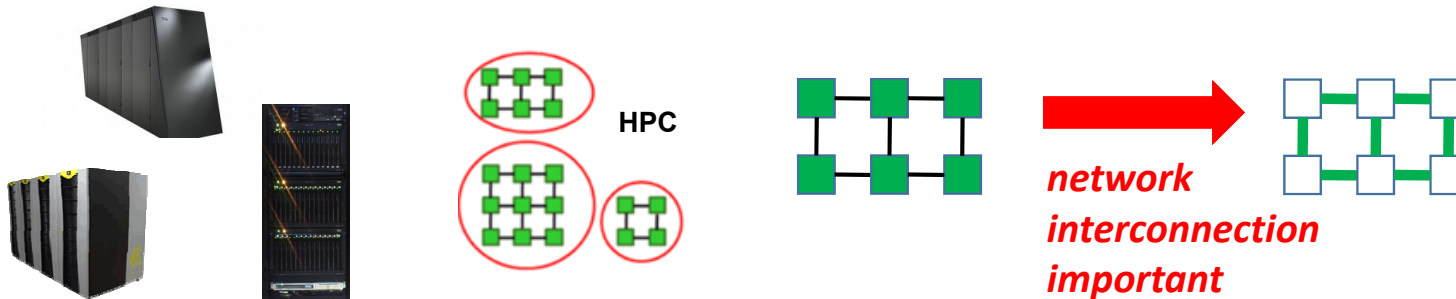
- HPC includes work on ‘[four basic building blocks](#)’ in this course
  - [Theory](#) (numerical laws, physical models, speed-up performance, etc.)
  - [Technology](#) (multi-core, supercomputers, networks, storages, etc.)
  - [Architecture](#) (shared-memory, distributed-memory, interconnects, etc.)
  - [Software](#) (libraries, schedulers, monitoring, applications, etc.)

[5] Introduction to High Performance Computing for Scientists and Engineers



# Understanding High Performance Computing (HPC) – Revisited

- High Performance Computing (HPC) is based on computing resources that enable the efficient use of parallel computing techniques through specific support with dedicated hardware such as high performance cpu/core interconnections.



- High Throughput Computing (HTC) is based on commonly available computing resources such as commodity PCs and small clusters that enable the execution of 'farming jobs' without providing a high performance interconnection between the cpu/cores.



➤ The complementary Cloud Computing & Big Data – Parallel Machine & Deep Learning Course focusses on High Throughput Computing

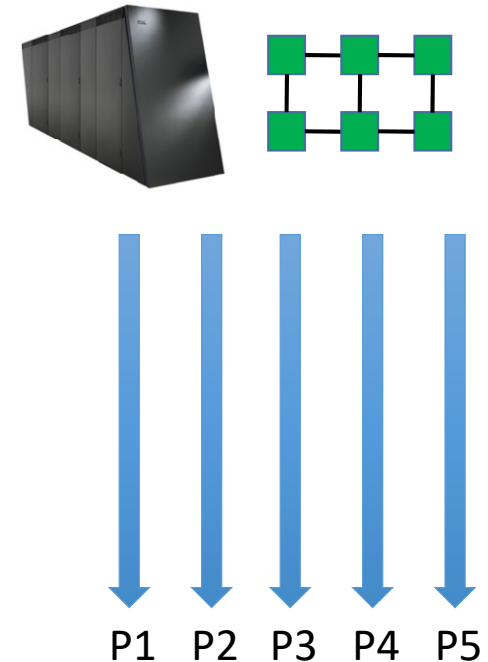
# Parallel Computing

- All modern supercomputers depend heavily on parallelism
  - Parallelism can be achieved with many different approaches

■ We speak of parallel computing whenever a number of 'compute elements' (e.g. cores) solve a problem in a cooperative way

*[5] Introduction to High Performance Computing for Scientists and Engineers*

- Often known as 'parallel processing' of some problem space
  - Tackle problems in parallel to enable the 'best performance' possible
  - Includes not only parallel computing, but also parallel input/output (I/O)
- 'The measure of speed' in High Performance Computing matters
  - Common measure for parallel computers established by TOP500 list
  - Based on benchmark for ranking the best 500 computers worldwide



*[6] TOP500 Supercomputing Sites*

➤ Lecture 3 will give in-depth details on parallelization fundamentals & performance term relationships & theoretical considerations

# TOP 500 List (June 2019)



[6] TOP500 Supercomputing Sites

- June 2019
- November 2018
- June 2018
- November 2017
- June 2017
- November 2016
- June 2016
- November 2015
- June 2015
- November 2014
- June 2014 ▶
- November 2013 ▶
- June 2013 ▶
- November 2012 ▶
- June 2012 ▶
- November 2011 ▶
- June 2011 ▶
- November 2010 ▶
- June 2010 ▶
- November 2009 ▶
- June 2009 ▶
- November 2008 ▶
- June 2008 ▶
- November 2007 ▶
- June 2007 ▶
- November 2006 ▶
- June 2006 ▶
- November 2005 ▶
- June 2005 ▶

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
2	Sierra - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
3	Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
4	Tianhe-2A - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
5	Frontera - Dell C6420, Xeon Platinum 8280 28C 2.7GHz, Mellanox InfiniBand HDR , Dell EMC Texas Advanced Computing Center/Univ. of Texas United States	448,448	23,516.4	38,745.9	
6	Piz Daint - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland	387,872	21,230.0	27,154.3	2,384
7	Trinity - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect , Cray Inc. DOE/NNSA/LANL/SNL United States	979,072	20,158.7	41,461.2	7,578
8	AI Bridging Cloud Infrastructure (ABCI) - PRIMERGY CX2570 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR , Fujitsu National Institute of Advanced Industrial Science and Technology (AIST) Japan	391,680	19,880.0	32,576.6	1,649
9	SuperMUC-NG - ThinkSystem SD650, Xeon Platinum 8174 24C 3.1GHz, Intel Omni-Path , Lenovo Leibniz Rechenzentrum Germany	305,856	19,476.6	26,873.9	

power  
challenge

EU #1

# LINPACK Benchmarks and Alternatives

- TOP500 ranking is based on the LINPACK benchmark

■ LINPACK solves a dense system of linear equations of unspecified size

*[7] LINPACK Benchmark implementation*

- LINPACK covers only a single architectural aspect ('critics exist')
  - Measures 'peak performance': All involved 'supercomputer elements' operate on maximum performance
  - Available through a wide variety of 'open source implementations'
  - Success via 'simplicity & ease of use' thus used for over two decades
- Realistic applications benchmark suites might be alternatives
  - HPC Challenge benchmarks (includes 7 tests)
  - JUBE benchmark suite (based on real applications)

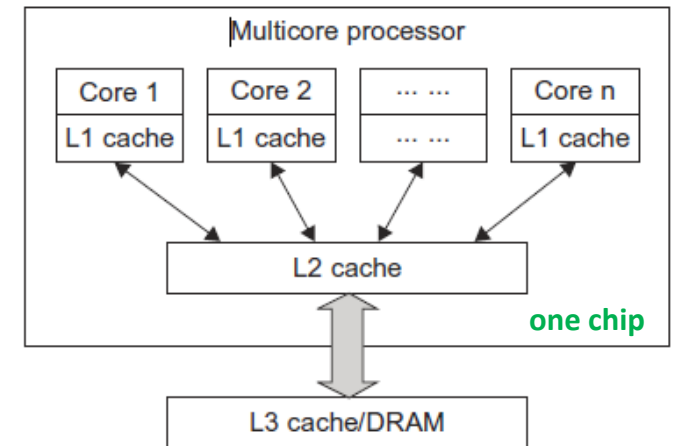
*[8] HPC Challenge Benchmark Suite*

*[9] JUBE Benchmark Suite*



# Multi-core CPU Processors

- Significant advances in CPU (or microprocessor chips)
  - Multi-core architecture with dual, quad, six, or n processing cores
  - Processing cores are all on one chip
- Multi-core CPU chip architecture
  - Hierarchy of caches (on/off chip)
  - L1 cache is private to each core; on-chip
  - L2 cache is shared; on-chip
  - L3 cache or Dynamic random access memory (DRAM); off-chip



[10] *Distributed & Cloud Computing Book*

- Clock-rate for single processors increased from 10 MHz (Intel 286) to 4 GHz (Pentium 4) in 30 years
- Clock rate increase with higher 5 GHz unfortunately reached a limit due to power limitations / heat
- Multi-core CPU chips have quad, six, or n processing cores on one chip and use cache hierarchies

# Dominant Architectures of HPC Systems

- Traditionally two dominant types of architectures

- Shared-Memory Computers
- Distributed Memory Computers

- Shared-memory parallelization with OpenMP
- Distributed-memory parallel programming with the Message Passing Interface (MPI) standard

- Often hierarchical (**hybrid**) systems of both in practice

- Dominance in the last couple of years in the community on X86-based commodity clusters running the Linux OS on Intel/AMD processors
- More recently, also **accelerators play a significant role** (e.g., many-core chips)

- More recently

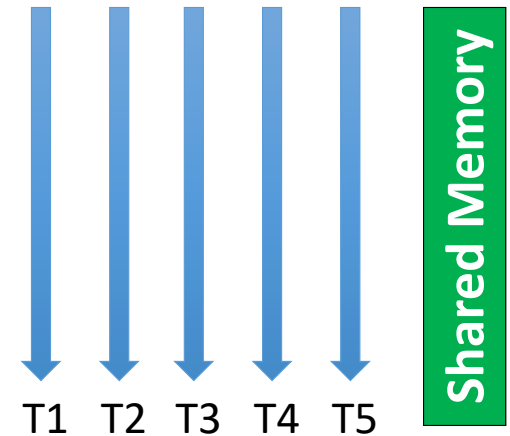
- Both above considered as '**programming models**'
- Emerging computing models getting relevant for HPC: e.g., **quantum devices, neuromorphic devices**

# Shared-Memory Computers

- A shared-memory parallel computer is a system in which a number of CPUs work on a common, shared physical address space

*[5] Introduction to High Performance Computing for Scientists and Engineers*

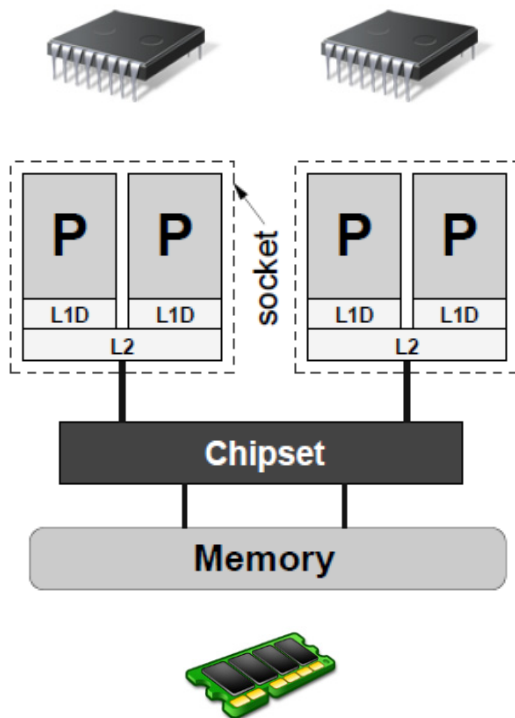
- Two varieties of shared-memory systems:
  1. Unified Memory Access (UMA)
  2. Cache-coherent Nonuniform Memory Access (ccNUMA)
- The Problem of 'Cache Coherence' (in UMA/ccNUMA)
  - Different CPUs use Cache to 'modify same cache values'
  - Consistency between cached data & data in memory must be guaranteed
  - 'Cache coherence protocols' ensure a consistent view of memory



# Shared-Memory with UMA

- UMA systems use 'flat memory model': Latencies and bandwidth are the same for all processors and all memory locations.
- Also called Symmetric Multiprocessing (SMP)

*[5] Introduction to High Performance Computing for Scientists and Engineers*



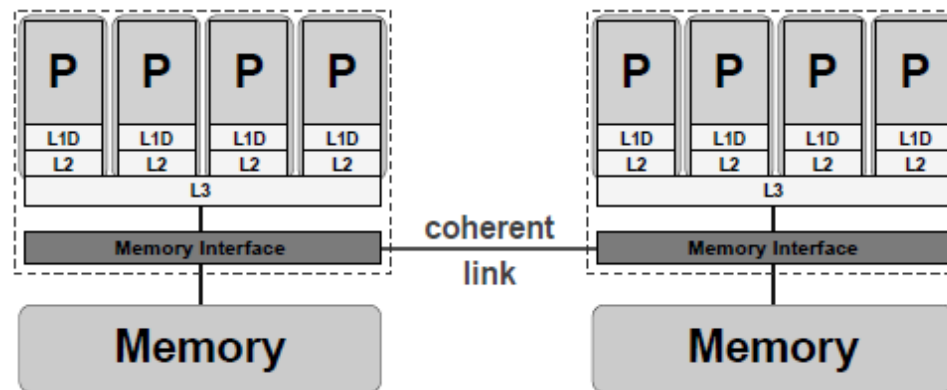
## Selected Features

- **Socket** is a physical package (with multiple cores), typically a replacable component
- Two dual core chips (2 core/socket)
- P = Processor core
- L1D = Level 1 Cache – Data (fastest)
- L2 = Level 2 Cache (fast)
- Memory = main memory (slow)
- **Chipset = enforces cache coherence and mediates connections to memory**

# Shared-Memory with ccNUMA

- ccNUMA systems share logically memory that is physically distributed (similar like distributed-memory systems)
- Network logic makes the aggregated memory appear as one single address space

*[5] Introduction to High Performance Computing for Scientists and Engineers*



## Selected Features

- Eight cores (4 cores/socket); L3 = Level 3 Cache
- Memory interface = establishes a coherent link to enable one 'logical' single address space of 'physically distributed memory'

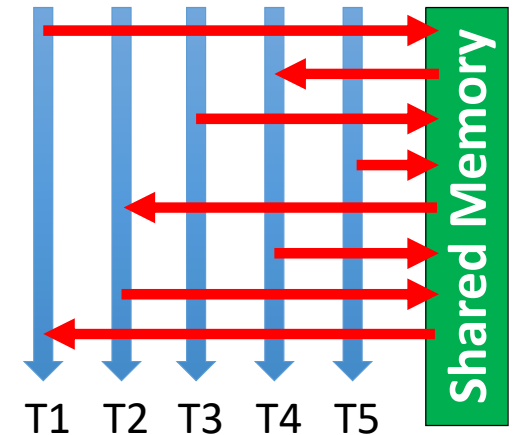
# Programming with Shared Memory using OpenMP

- Shared-memory programming enables immediate access to all data from all processors without explicit communication
- OpenMP is dominant shared-memory programming standard today (v3)
- OpenMP is a set of compiler directives to 'mark parallel regions'

*[11] OpenMP API Specification*

## ■ Features

- Bindings are defined for C, C++, and Fortran languages
- Threads TX are 'lightweight processes' that mutually access data

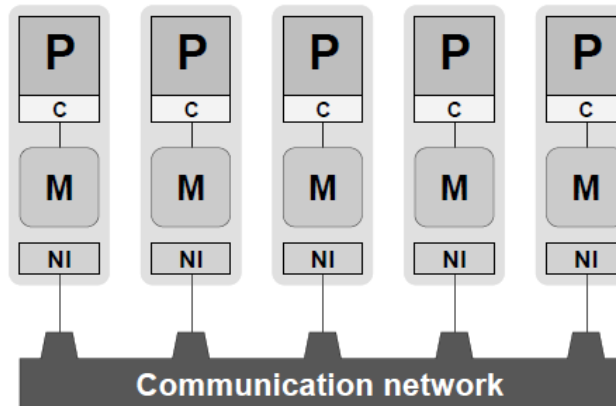


➤ Lecture 6 will give in-depth details on the shared-memory programming model with OpenMP and using its compiler directives

# Distributed-Memory Computers

- A distributed-memory parallel computer establishes a 'system view' where no process can access another process' memory directly

*[5] Introduction to High Performance Computing for Scientists and Engineers*



## ■ Features

- Processors communicate via [Network Interfaces \(NI\)](#)
- NI mediates the connection to a [Communication network](#)
- [This setup is rarely used → a programming model view today](#)



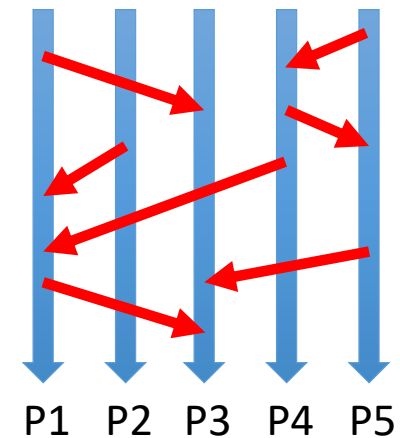
# Programming with Distributed Memory using MPI

- Distributed-memory programming enables explicit message passing as communication between processors
- Message Passing Interface (MPI) is dominant distributed-memory programming standard today (available in many different version)
- MPI is a standard defined and developed by the MPI Forum

[12] MPI Standard

## ■ Features

- No **remote memory access** on distributed-memory systems
- Require to **'send messages'** back and forth between processes PX
- Many free **Message Passing Interface (MPI)** libraries available
- Programming is tedious & complicated, but **most flexible method**

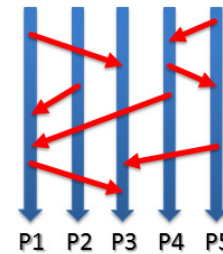


➤ Lecture 2 & 4 will give in-depth details on the distributed-memory programming model with the Message Passing Interface (MPI)

# MPI Standard – GNU OpenMPI Implementation Example – Revisited

## ■ Message Passing Interface (MPI)

- A standardized and portable message-passing standard
- Designed to support different HPC architectures
- A wide variety of MPI implementations exist
- Standard defines the syntax and semantics of a core of library routines used in C, C++ & Fortran



[12] MPI Standard

## ■ OpenMPI Implementation

- Open source license based on the BSD license
- Full MPI (version 3) standards conformance
- Developed & maintained by a consortium of academic, research, & industry partners
- Typically available as modules on HPC systems and used with mpicc compiler
- Often built with the GNU compiler set and/or Intel compilers



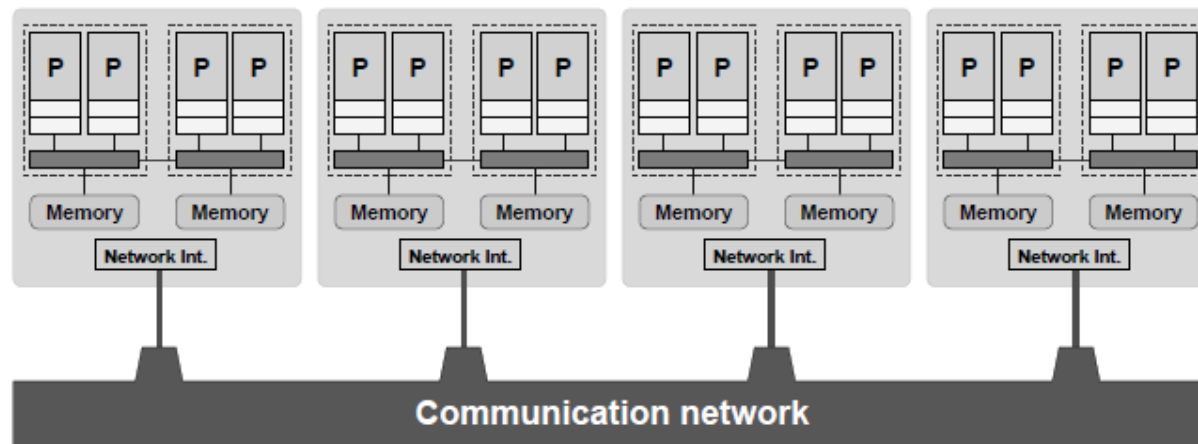
[13] OpenMPI Web page

➤ Lecture 2 will provide a full introduction and many more examples of the Message Passing Interface (MPI) for parallel programming

# Hierarchical Hybrid Computers

- A hierarchical hybrid parallel computer is neither a purely shared-memory nor a purely distributed-memory type system but a mixture of both
- Large-scale 'hybrid' parallel computers have shared-memory building blocks interconnected with a fast network today

*[5] Introduction to High Performance Computing for Scientists and Engineers*



## ■ Features

- Shared-memory nodes (here ccNUMA) with local NIs
- NI mediates connections to other remote 'SMP nodes'

# Programming Hybrid Systems & Patterns

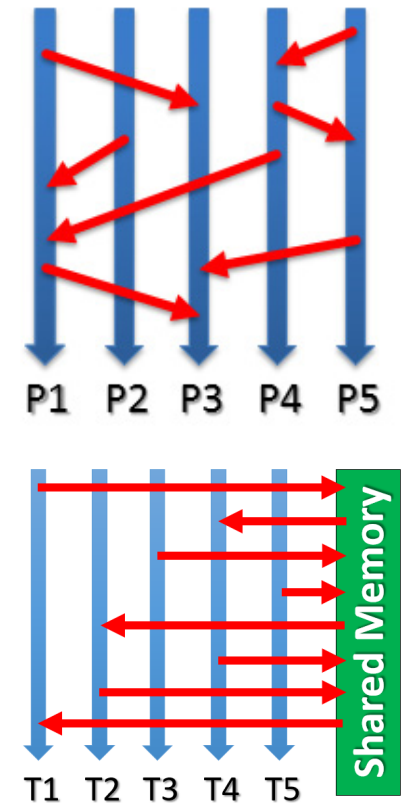
- Hybrid systems programming uses MPI as explicit internode communication and OpenMP for parallelization within the node
- Parallel Programming is often supported by using 'patterns' such as stencil methods in order to apply functions to the domain decomposition

## ■ Experience from HPC Practice

- Most parallel applications still take no notice of the hardware structure
- Use of **pure MPI for parallelization remains the dominant programming**
- Historical reason: old supercomputers all distributed-memory type
- **Use of accelerators is significantly increasing in practice today**

## ■ Challenges with the 'mapping problem'

- Performance of hybrid (as well as pure MPI codes) depends crucially on factors not directly connected to the programming model
- It largely depends on the **association of threads and processes to cores**
- **Patterns (e.g., stencil methods) support the parallel programming**

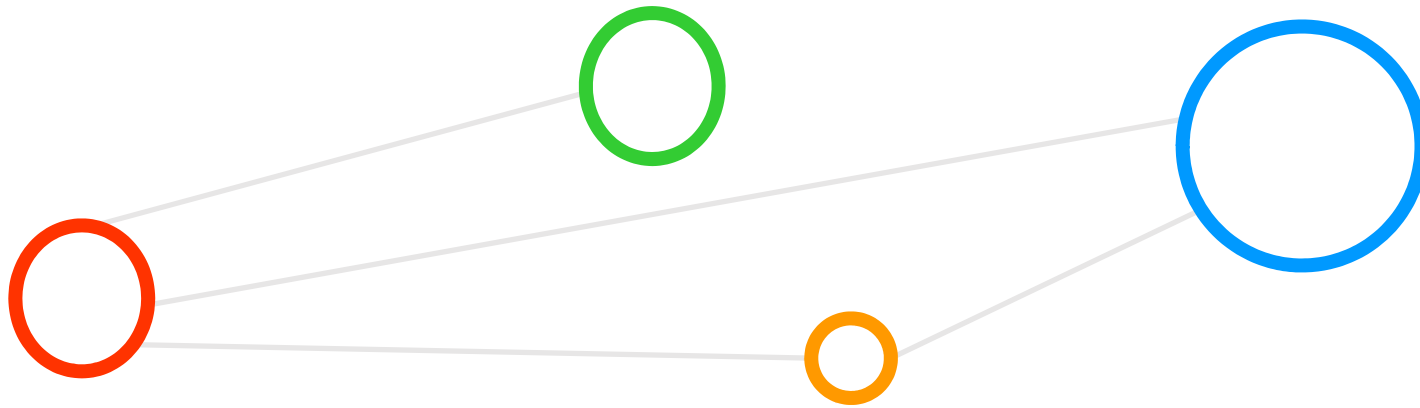


➤ Lecture 10 will provide insights into hybrid programming models and introduces selected patterns used in parallel programming

## [Video] Juelich – Supercomputer Upgrade



# HPC Ecosystem Technologies



# HPC System Software Environment – Revisited (cf. Practical Lecture 0.2)

## ■ Operating System

- Former times often ‘proprietary OS’, nowadays often (reduced) ‘Linux’

## ■ Scheduling Systems

- Manage concurrent access of users on Supercomputers
- Different scheduling algorithms can be used with different ‘batch queues’
- Example: [SLURM @ JÖTUNN Cluster](#), LoadLeveler @ JUQUEEN, etc.

## ■ Monitoring Systems

- Monitor and test status of the system (‘[system health checks/heartbeat](#)’)
- Enables view of usage of system per node/rack (‘[system load](#)’)
- Examples: [LLView](#), INCA, [Ganglia @ JOTUNN Cluster](#), etc.

## ■ Performance Analysis Systems

- Measure performance of an application and recommend improvements (.e.g Scalasca, Vampir, etc.)

▪ HPC systems and supercomputers typically provide a software environment that support the processing of parallel and scalable applications

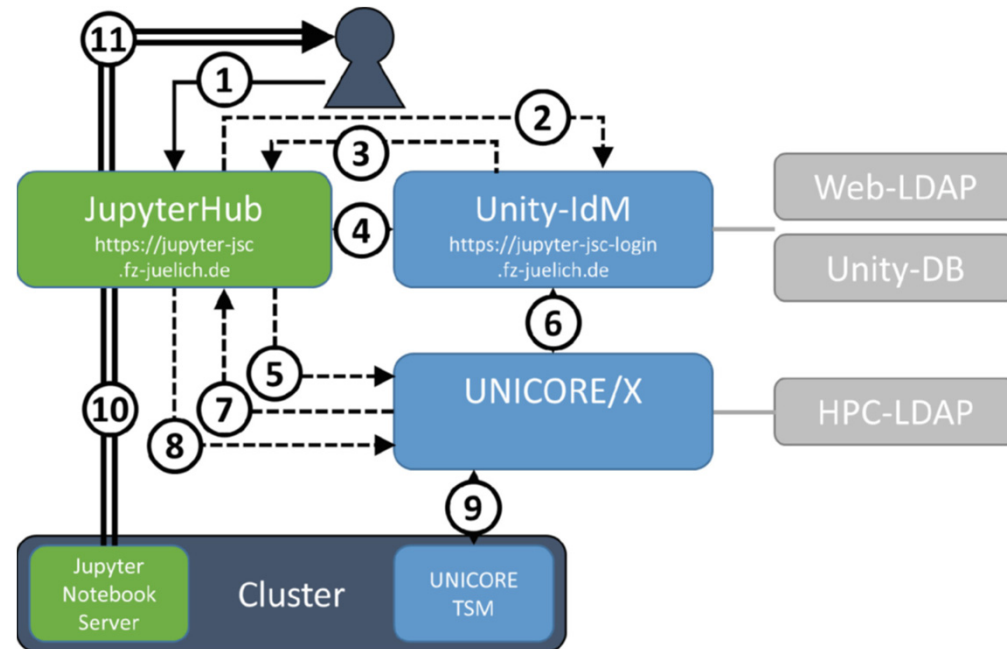
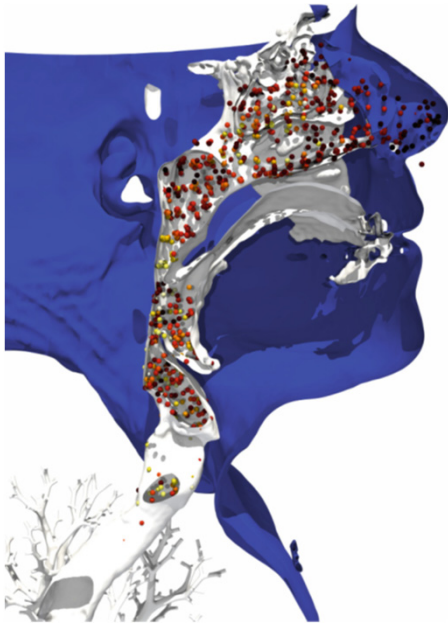
▪ Monitoring systems offer a comprehensive view of the current status of a HPC system or supercomputer

▪ Scheduling systems enable a method by which user processes are given access to processors

➤ Lecture 9 will offer more insights into performance analysis systems with debugging, profiling, and HPC performance toolsets



# Scheduling vs. Emerging Interactive Supercomputing Approaches



▪ JupyterHub is a multi-user version of the notebook designed for companies, classrooms and research labs



[20] A. Lintermann & M. Riedel et al., 'Enabling Interactive Supercomputing at JSC – Lessons Learned'

[21] A. Streit & M. Riedel et al., 'UNICORE 6 – Recent and Future Advancements'

[22] Project Jupyter Web page

# Modular Supercomputer JUWELS – Revisited



# HPC System Architectures

- HPC systems are very complex 'machines' with many elements

- CPUs & multi-cores
- 'multi-threading' capabilities
- Data access levels
- Different levels of Caches
- Network topologies
- Various interconnects

- Architecture Impacts

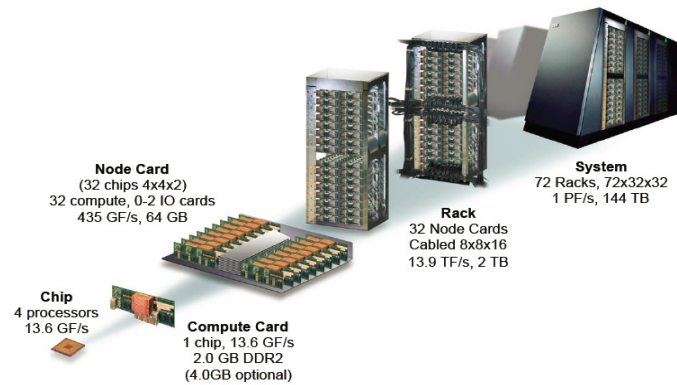
- Vendor designs, e.g., IBM Bluegene/Q
- Infrastructure, e.g., cooling & power lines in computing hall



- HPC faced a significant change in practice with respect to performance increase after years
- Getting more speed for free by waiting for new CPU generations does not work any more
- Multi-core processors emerge that require to use those multiple resources efficiently in parallel
- Many-core processors emerge that are used to accelerate certain computing application parts

# Example: IBM BlueGene Architecture Evolution

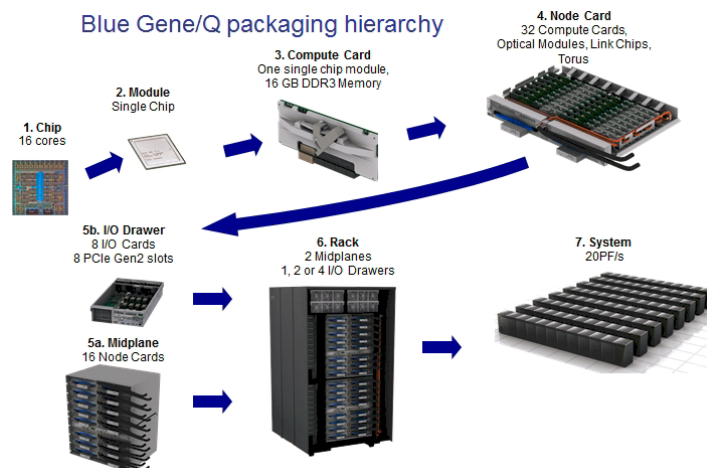
## ■ BlueGene/P



Source: IBM

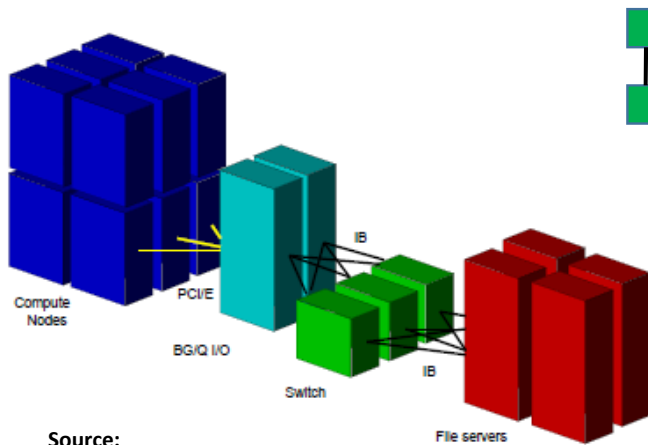
## ■ BlueGene/Q

### Blue Gene/Q packaging hierarchy

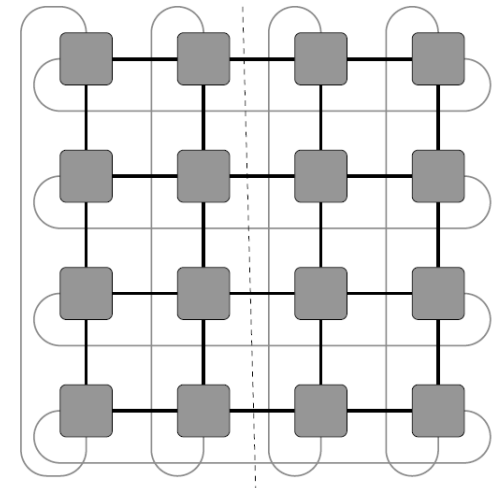


# Network Topologies

- Large-scale HPC Systems have **special network setups**
  - Dedicated I/O nodes, fast interconnects, e.g. Infiniband (IB)
  - Different network topologies, e.g. tree, 5D Torus network, mesh, etc. (raise challenges in task mappings and communication patterns)



Source:  
IBM



[5] *Introduction to High Performance Computing for Scientists and Engineers*



# HPC System Architecture & Continuous Developments

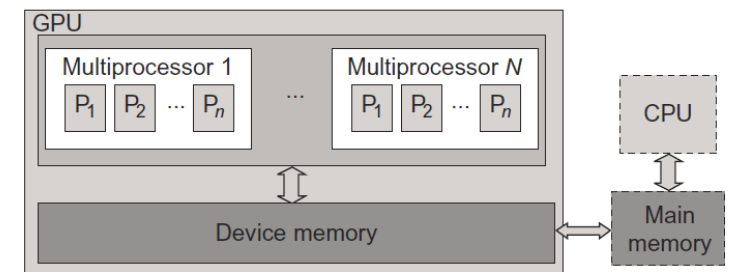
- Increasing number of other 'new' emerging system architectures
  - HPC at cutting-edge of computing and integrates new hardware developments as continuous activity
- General Purpose Computation on Graphics Processing Unit (GPGPUs/GPUs)
  - Use of GPUs instead for computer graphics for computing
  - Programming models are OpenCL and Nvidia CUDA
  - Getting more and more adopted in many application fields
- Field Programmable Gate Array (FPGAs)
  - Integrated circuit designed to be configured by a user after shipping
  - Enables updates of functionality and reconfigurable 'wired' interconnects
- Cell processors
  - Enables combination of general-purpose cores with co-processing elements that accelerate dedicated forms of computations



- Artificial Intelligence with methods from machine learning and deep learning influence HPC system architectures today
- Complement initial focus on compute-intensive with data-intensive application co-design activities

# Many-core GPGPUs

- Use of very many simple cores
  - High throughput computing-oriented architecture
  - Use massive parallelism by executing a lot of concurrent threads slowly
  - Handle an ever increasing amount of multiple instruction threads
  - CPUs instead typically execute a single long thread as fast as possible
- Many-core GPUs are used in large clusters and within massively parallel supercomputers today
  - Named General-Purpose Computing on GPUs (GPGPU)
  - Different programming models emerge



[10] Distributed & Cloud Computing Book

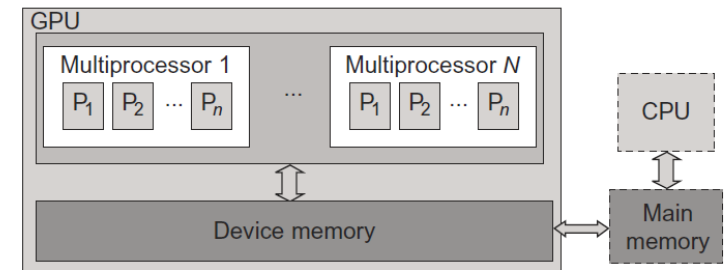
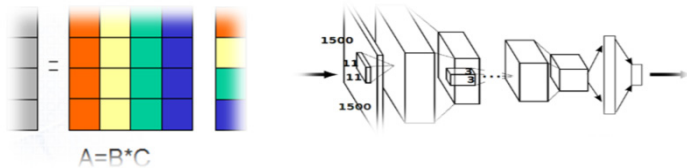
- Graphics Processing Unit (GPU) is great for data parallelism and task parallelism
- Compared to multi-core CPUs, GPUs consist of a many-core architecture with hundreds to even thousands of very simple cores executing threads rather slowly



# GPU Acceleration

- GPU accelerator architecture example (e.g. NVIDIA card)

- GPUs can have **128 cores** on one single GPU chip
- Each core can work with **eight threads** of instructions
- GPU is able to concurrently execute  **$128 * 8 = 1024$  threads**
- Interaction and thus major (bandwidth) bottleneck between CPU and GPU is via **memory interactions**
- E.g. applications that use **matrix – vector/matrix multiplication** (e.g. deep learning algorithms)

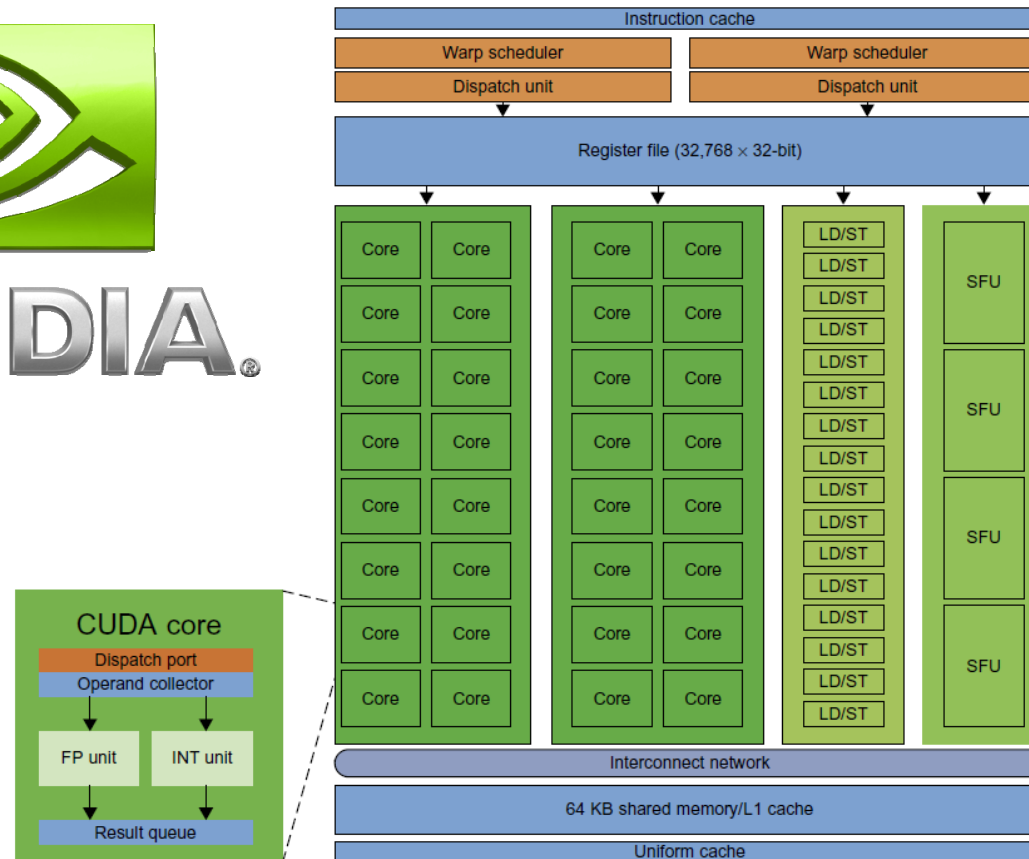


[10] Distributed & Cloud Computing Book

- CPU acceleration means that GPUs accelerate computing due to a massive parallelism with thousands of threads compared to only a few threads used by conventional CPUs
- GPUs are designed to compute large numbers of floating point operations in parallel

➤ Lecture 10 will introduce the programming of accelerators with different approaches and their key benefits for applications

# NVIDIA Fermi GPU Example



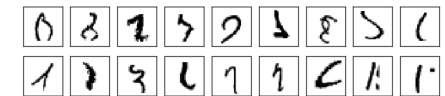
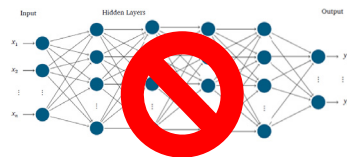
[10] Distributed & Cloud Computing Book

# DEEP Learning takes advantage of Many-Core Technologies

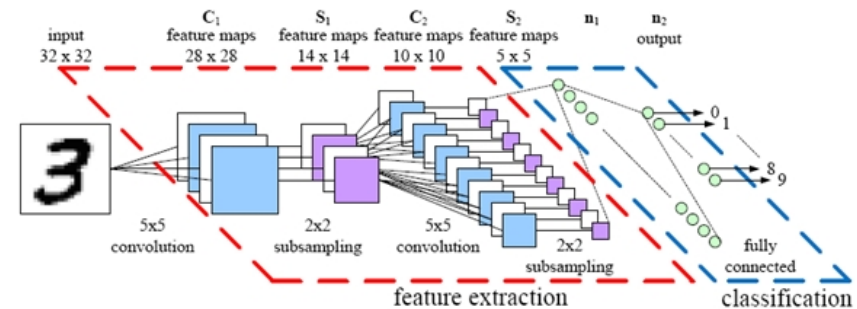


www.cybercontrols.org

[25] Neural Network 3D Simulation



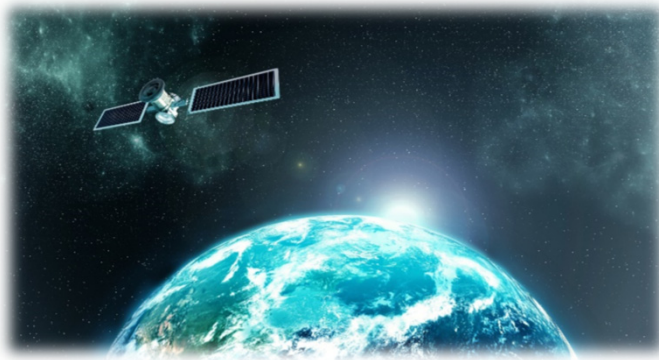
## ■ Innovation via specific layers and architecture types



[26] A. Rosebrock

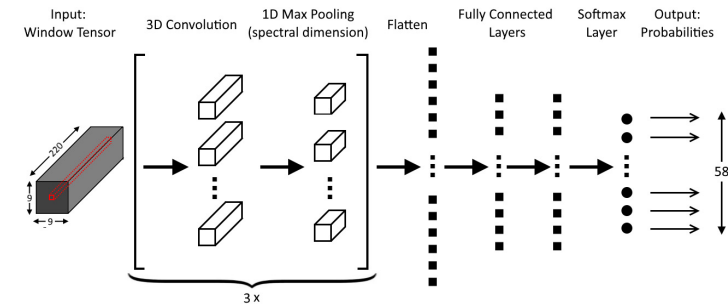
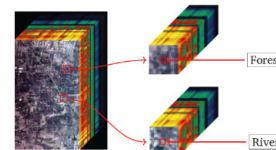
➤ Lecture 8 will provide more details about parallel & scalable machine & deep learning algorithms and how many-core HPC is used

# Deep Learning Application Example – Using High Performance Computing



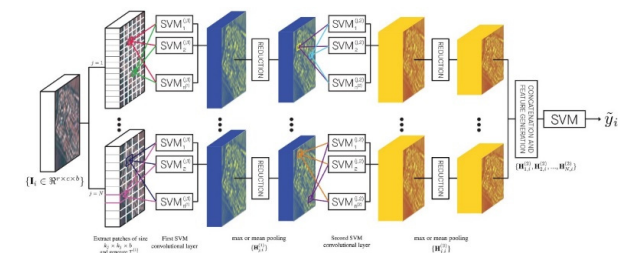
- Using Convolutional Neural Networks (CNNs) with hyperspectral remote sensing image data

[27] J. Lange and M. Riedel et al., IGARSS Conference, 2018



Feature	Representation / Value
Conv. Layer Filters	48, 32, 32
Conv. Layer Filter size	(3, 3, 5), (3, 3, 5), (3, 3, 5)
Dense Layer Neurons	128, 128
Optimizer	SGD
Loss Function	mean squared error
Activation Functions	ReLU
Training Epochs	600
Batch Size	50
Learning Rate	1
Learning Rate Decay	$5 \times 10^{-6}$

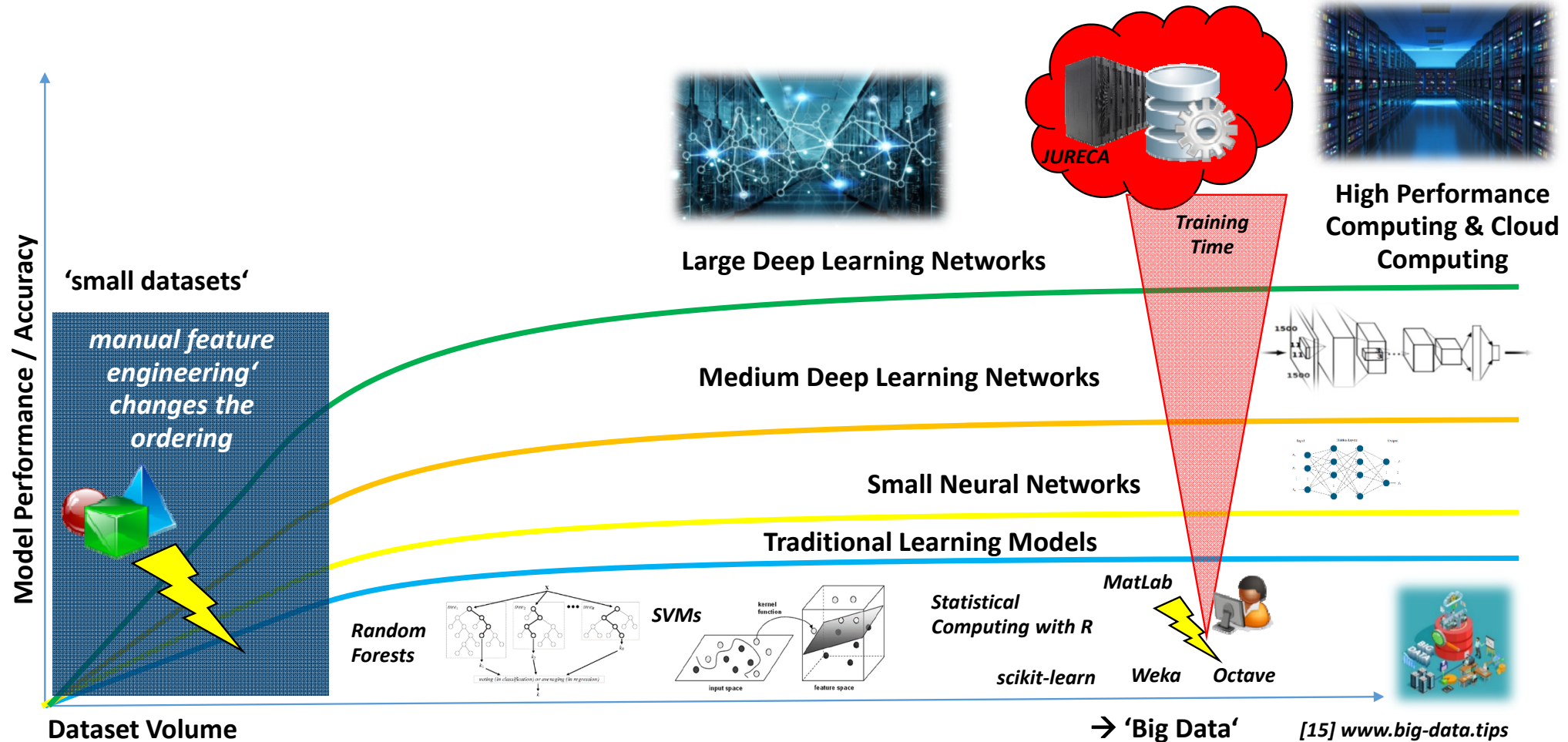
- Find Hyperparameters & joint 'new-old' modeling & transfer learning given rare labeled/annotated data in science (e.g. 36,000 vs. 14,197,122 images ImageNet)



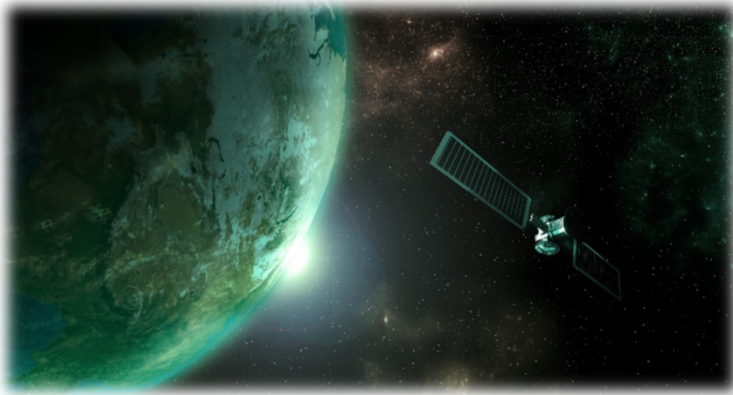
[28] G. Cavallaro, M. Riedel et al., IGARSS 2019

➤ Lecture 8 will provide more details about parallel & scalable machine & deep learning algorithms and remote sensing applications

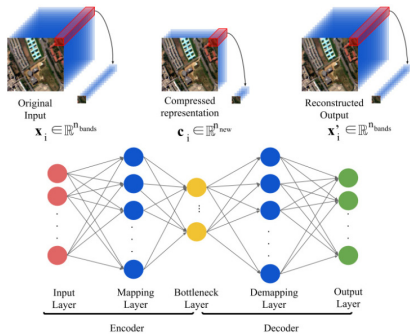
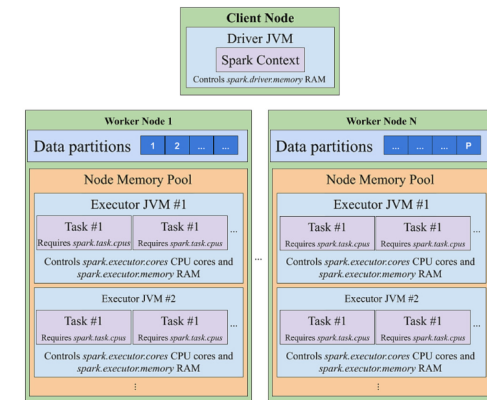
# HPC Relationship to 'Big Data' in Machine & Deep Learning



# Deep Learning Application Example – Using Cloud Computing



- Performing parallel computing with Apache Spark across different worker nodes

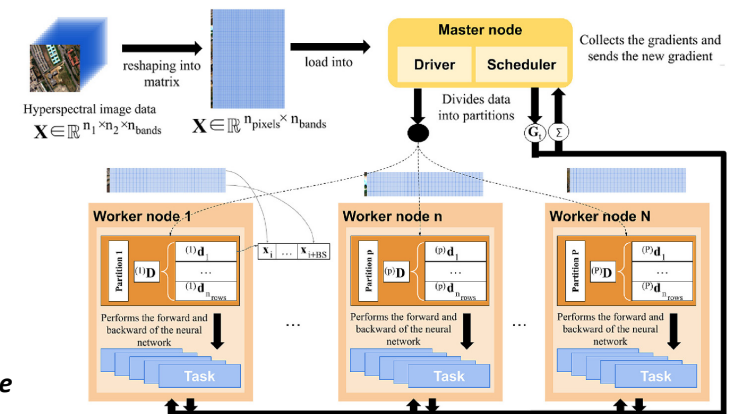


[23] J. Haut, G. Cavallaro and M. Riedel et al.,  
IEEE Transactions on Geoscience and Remote Sensing, 2019

- Using Autoencoder deep neural networks with Cloud computing



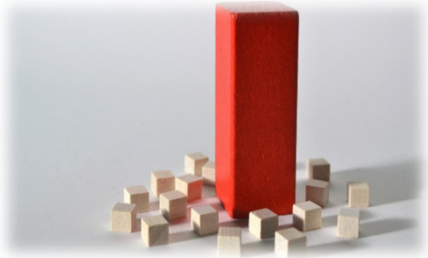
[24] Apache Spark Web page



➤ The complementary Cloud Computing & Big Data – Parallel Machine & Deep Learning Course teaches Apache Spark Approaches

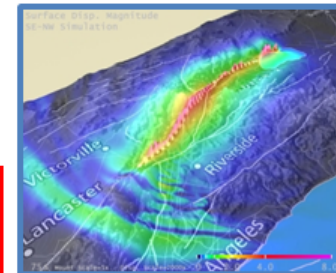


# HPC Relationship to 'Big Data' in Simulation Sciences



**Better Prediction Accuracy  
Involves “Bigger” Data**

<i>Estimated figures for simulated 240 second period, 100 hour run-time</i>	<b>TeraShake domain (600x300x80 km<sup>3</sup>)</b>	<b>PetaShake domain (800x400x100 km<sup>3</sup>)</b>
<b>Fault system interaction</b>	NO	YES
<b>Inner Scale</b>	200m	25m
<b>Resolution of terrain grid</b>	1.8 billion mesh points	2.0 trillion mesh points
<b>Magnitude of Earthquake</b>	7.7	8.1
<b>Time steps</b>	20,000 (.012 sec/step)	160,000 (.0015 sec/step)
<b>Surface data</b>	1.1 TB	1.2 PB
<b>Volume data</b>	43 TB	4.9 PB



Information courtesy of the Southern California  
Earthquake Center

[14] F. Berman: Maximising the Potential of Research Data

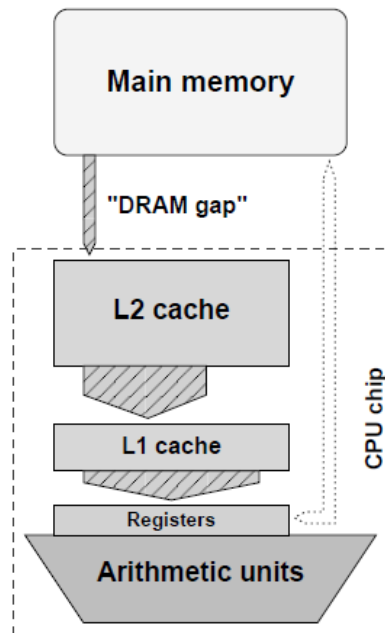
# Data Access & Challenges

*too slow*

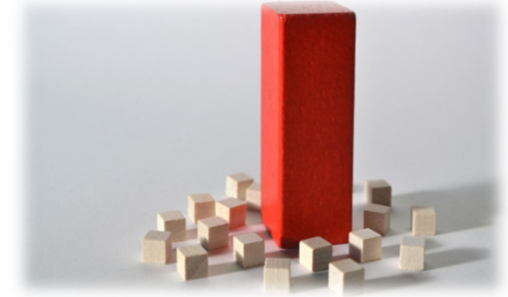


*cheaper*

*faster*



- P = Processor core elements
  - Compute: floating points or integers
  - Arithmetic units (compute operations)
  - Registers (feed those units with operands)
- 'Data access' for application/levels
  - Registers: 'accessed w/o any delay'
  - L1D = Level 1 Cache – Data (fastest, normal)
  - L2 = Level 2 Cache (fast, often)
  - L3 = Level 3 Cache (still fast, less often)
  - Main memory (slow, but larger in size)
  - Storage media like harddisk, tapes, etc. (too slow to be used in direct computing)



- The DRAM gap is the large discrepancy between main memory and cache bandwidths

[5] *Introduction to High Performance Computing for Scientists and Engineers*



# Big Data Drives Data-Intensive HPC Architecture Designs

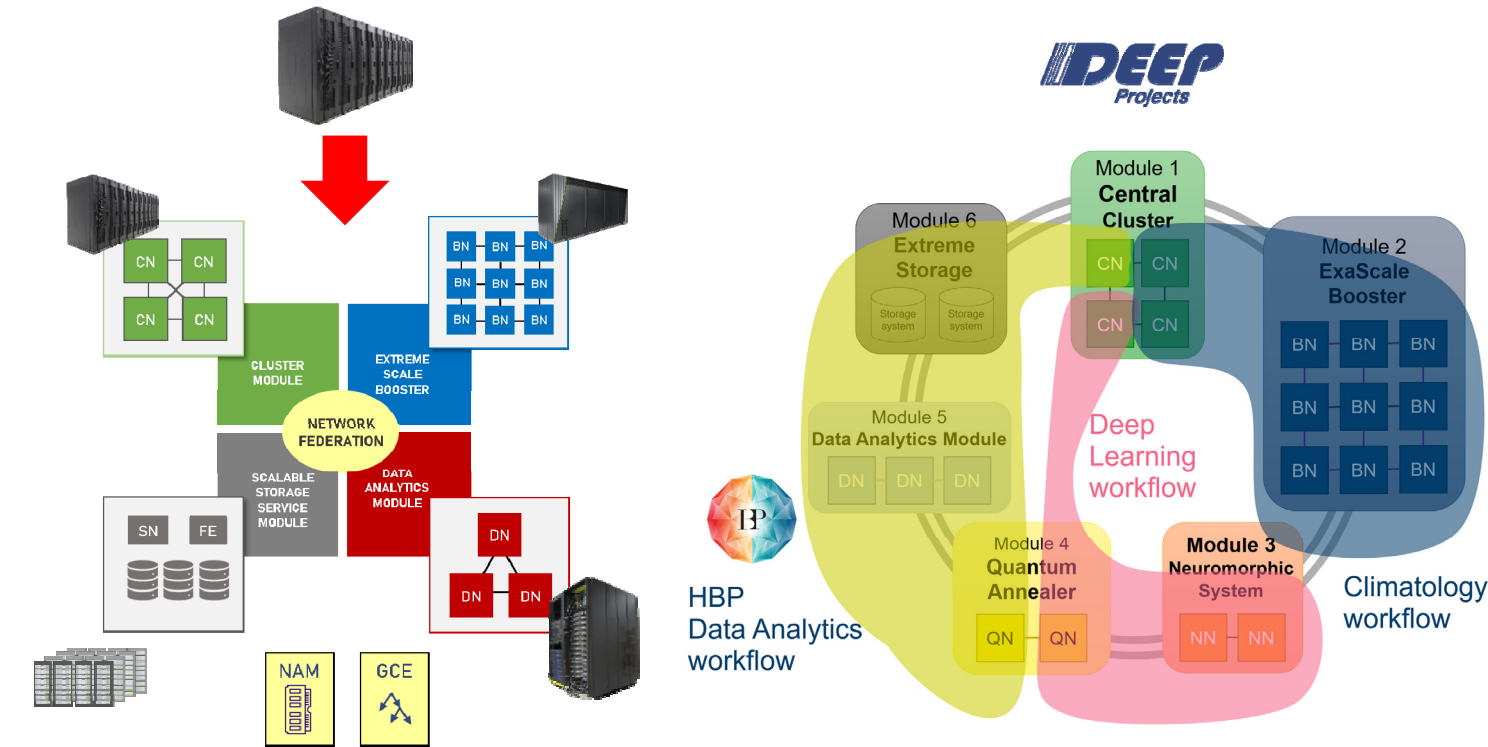
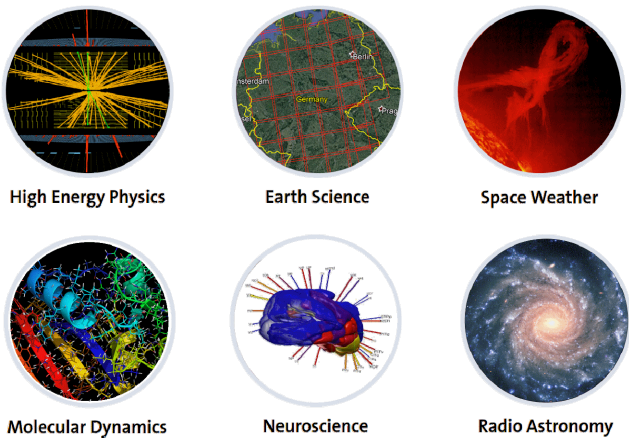
- More recently system architectures are influenced by ‘big data’
  - CPU speed has surpassed IO capabilities of existing HPC resources
  - Scalable I/O gets more and more important in application scalability
- Requirements for Hierarchical Storage Management (‘Tiers’)
  - Mass storage devices (tertiary storage) too slow to enable active processing of ‘big data’
  - Increase in simulation time/granularity means TBs equally important as FLOP/s
  - Tapes cheap, but slowly accessible, direct access to compute nodes needed
  - Drive new ‘tier-based’ designs



	server	core	mem[GB]	disk[TB]	Count
Tier 1	2950	8	16	22.50	40
Tier 2	R900	16	64	33.75	4
Tier 3	R900	16	128	11.25	2
total		416	1152	1057.50	46

[16] A. Szalay et al., ‘GrayWulf: Scalable Clustered Architecture for Data Intensive Computing’

# Application Co-Design of HPC Architectures – Modular Supercomputing Example



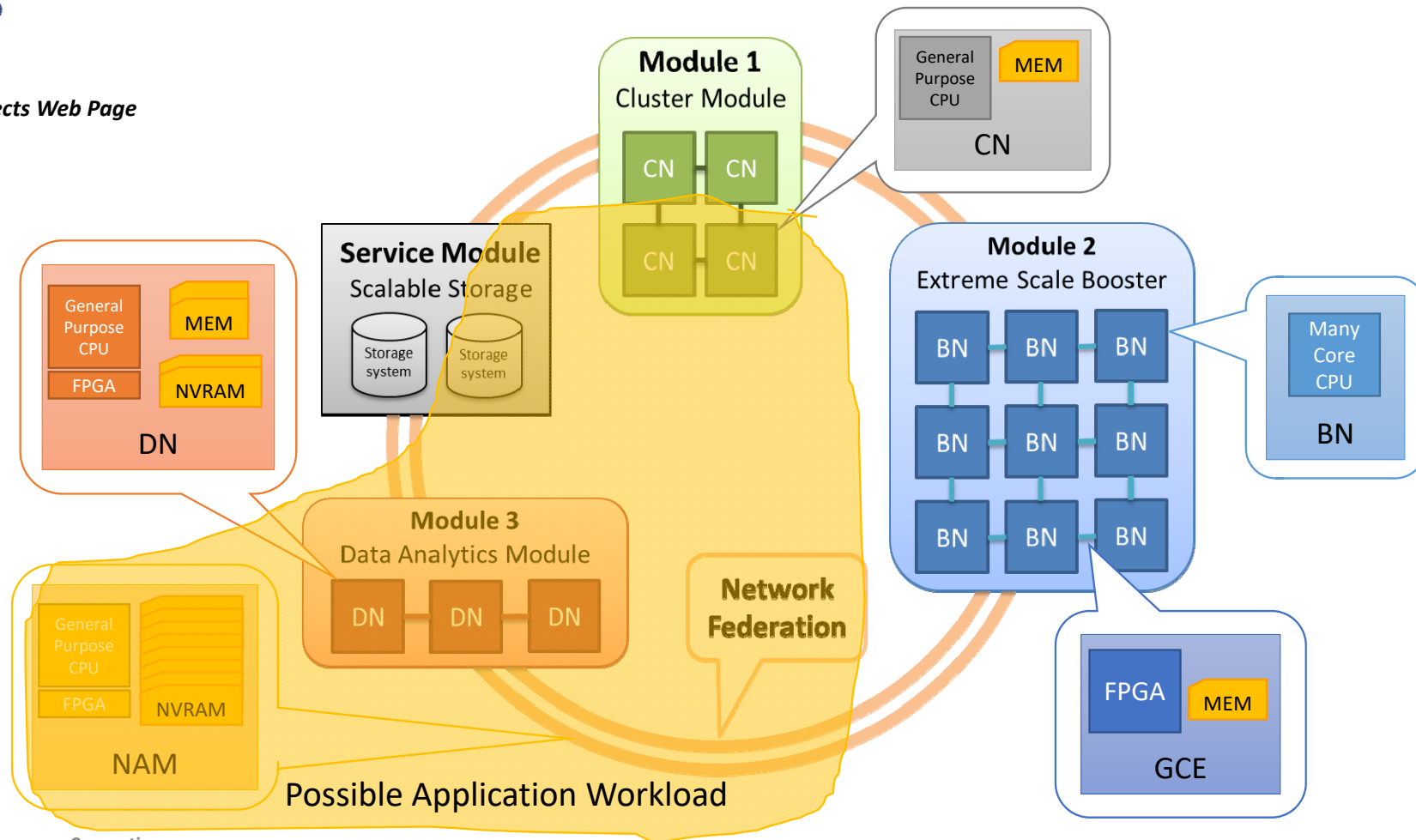
- The modular supercomputing architecture (MSA) enables a flexible HPC system design co-designed by the need of different application workloads

[17] DEEP Projects Web Page

# New HPC Architectures – Modular Supercomputing Architecture Example



[17] DEEP Projects Web Page



# Large-scale Computing Infrastructures

- Large computing systems are often embedded in infrastructures
  - Grid computing for **distributed data storage and processing** via middleware
  - The success of Grid computing was renowned when being mentioned by Prof. Rolf-Dieter Heuer, CERN Director General, in the context of the Higgs Boson Discovery:
- Other large-scale distributed infrastructures exist
  - Partnership for Advanced Computing in Europe (PRACE) → EU HPC
  - Extreme Engineering and Discovery Environment (XSEDE) → US HPC

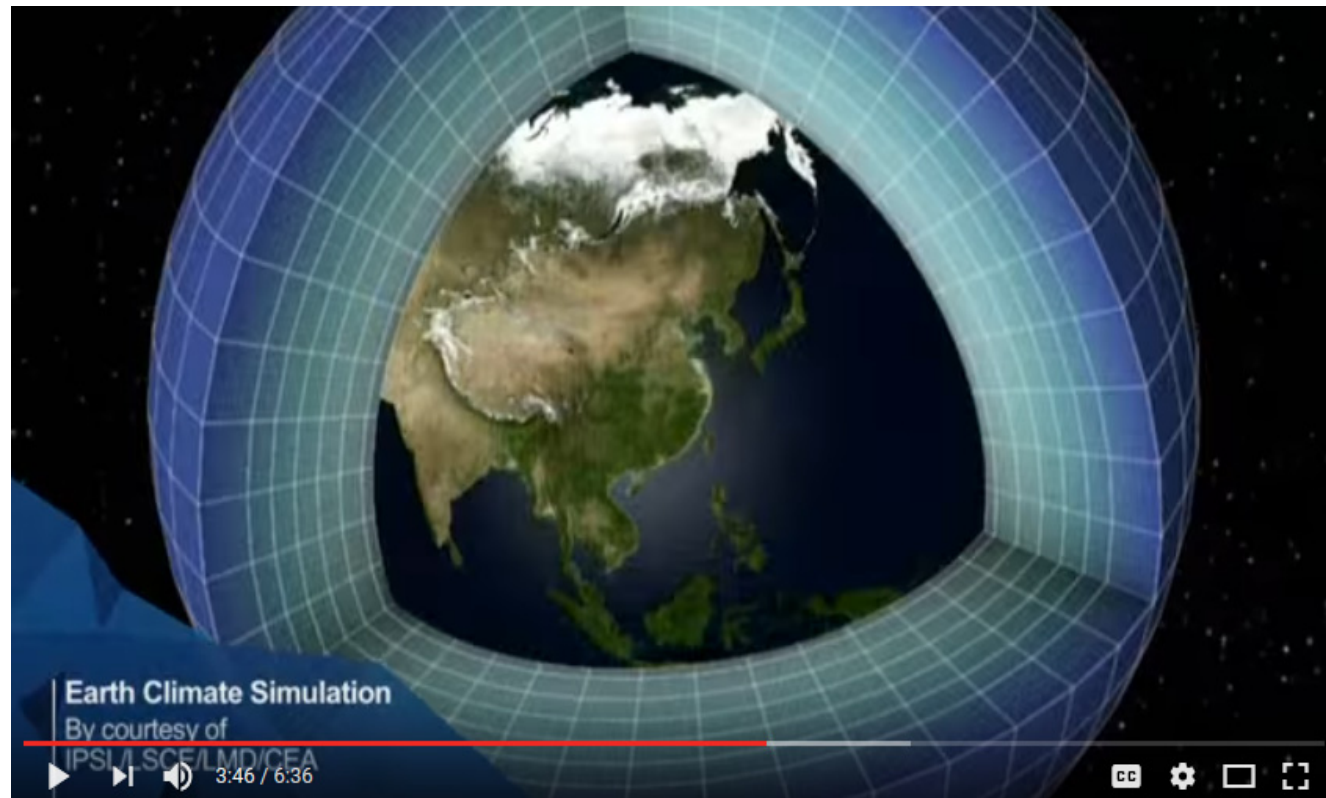
■ 'Results today only possible due to extraordinary performance of Accelerators – Experiments – Grid computing'



*[18] Grid Computing Video*

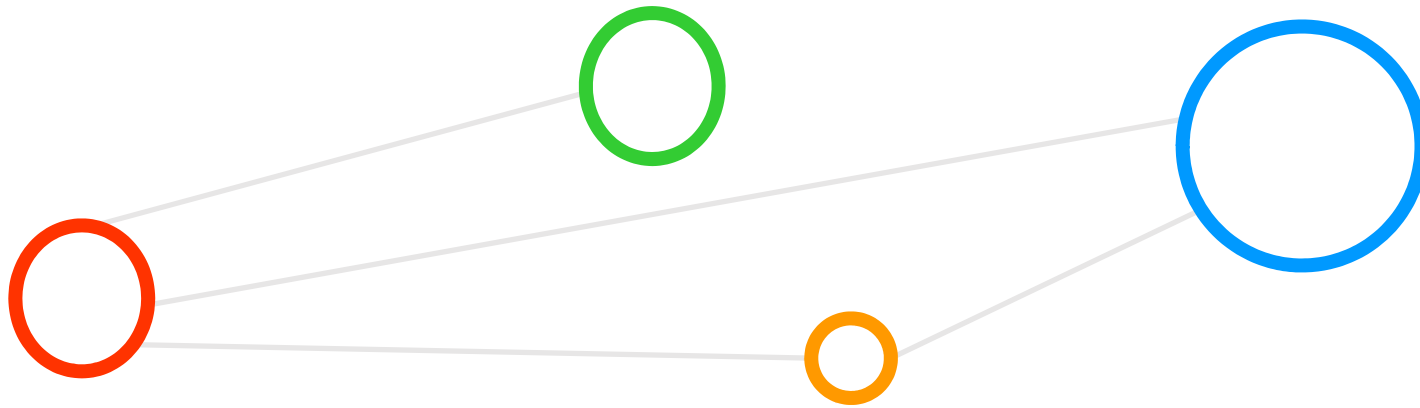
➤ Lecture 11 will give in-depth details on scalable approaches in large-scale HPC infrastructures and how to use them with middleware

## [Video] PRACE – Introduction to Supercomputing



*[19] PRACE – Introduction to Supercomputing*

# Lecture Bibliography





# Lecture Bibliography (1)

- [1] Terrestrial Systems Simulation Lab, Online:  
[http://www.hpsc-terrsys.de/hpsc-terrsys/EN/Home/home\\_node.html](http://www.hpsc-terrsys.de/hpsc-terrsys/EN/Home/home_node.html)
- [2] Nest:: The Neural Simulation Technology Initiative, Online:  
<https://www.nest-simulator.org/>
- [3] T. Bauer, 'System Monitoring and Job Reports with LLView', Online:  
[https://www.fz-juelich.de/SharedDocs/Downloads/IAS/JSC/EN/slides/supercomputer-ressources-2018-11/12b-sc-llview.pdf?\\_\\_blob=publicationFile](https://www.fz-juelich.de/SharedDocs/Downloads/IAS/JSC/EN/slides/supercomputer-ressources-2018-11/12b-sc-llview.pdf?__blob=publicationFile)
- [4] Wikipedia 'Supercomputer', Online:  
<http://en.wikipedia.org/wiki/Supercomputer>
- [5] Introduction to High Performance Computing for Scientists and Engineers, Georg Hager & Gerhard Wellein, Chapman & Hall/CRC Computational Science, ISBN 143981192X, English, ~330 pages, 2010, Online:  
<http://www.amazon.de/Introduction-Performance-Computing-Scientists-Computational/dp/143981192X>
- [6] TOP500 Supercomputing Sites, Online:  
<http://www.top500.org/>
- [7] LINPACK Benchmark, Online:  
<http://www.netlib.org/benchmark/hpl/>
- [8] HPC Challenge Benchmark Suite, Online:  
<http://icl.cs.utk.edu/hpcc/>
- [9] JUBE Benchmark Suite, Online:  
[http://www.fz-juelich.de/ias/jsc/EN/Expertise/Support/Software/JUBE/\\_node.html](http://www.fz-juelich.de/ias/jsc/EN/Expertise/Support/Software/JUBE/_node.html)
- [10] K. Hwang, G. C. Fox, J. J. Dongarra, 'Distributed and Cloud Computing', Book, Online:  
[http://store.elsevier.com/product.jsp?locale=en\\_EU&isbn=9780128002049](http://store.elsevier.com/product.jsp?locale=en_EU&isbn=9780128002049)
- [11] The OpenMP API specification for parallel programming, Online:  
<http://openmp.org/wp/openmp-specifications/>

# Lecture Bibliography (2)

- [12] The MPI Standard, Online:  
<http://www.mpi-forum.org/docs/>
- [13] OpenMPI Web page, Online:  
<https://www.open-mpi.org/>
- [14] Fran Berman, 'Maximising the Potential of Research Data'
- [15] Big Data Tips – Big Data Mining & Machine Learning, Online:  
<http://www.big-data.tips/>
- [16] A. Szalay et al., 'GrayWulf: Scalable Clustered Architecture for Data Intensive Computing', Proceedings of the 42nd Hawaii International Conference on System Sciences – 2009
- [17] DEEP Projects Web page, Online:  
<http://www.deep-projects.eu/>
- [18] How EMI Contributed to the Higgs Boson Discovery, YouTube Video, Online:  
<http://www.youtube.com/watch?v=FgcoLUys3RY&list=UUz8n-tukF1S7fqI19KOAAhw>
- [19] PRACE – Introduction to Supercomputing, Online:  
<https://www.youtube.com/watch?v=D94FJx9vxFA>
- [20] A. Lintermann & M. Riedel et al., 'Enabling Interactive Supercomputing at JSC – Lessons Learned', ISC 2019, Frankfurt, Germany, Online:  
[https://www.researchgate.net/publication/330621591\\_Enabling\\_Interactive\\_Supercomputing\\_at\\_JSC\\_Lessons\\_Learned\\_ISC\\_High\\_Performance\\_2018\\_International\\_Workshops\\_FrankfurtMain\\_Germany\\_June\\_28\\_2018\\_Revised\\_Selected\\_Papers](https://www.researchgate.net/publication/330621591_Enabling_Interactive_Supercomputing_at_JSC_Lessons_Learned_ISC_High_Performance_2018_International_Workshops_FrankfurtMain_Germany_June_28_2018_Revised_Selected_Papers)
- [21] A. Streit & M. Riedel et al., 'UNICORE 6 – Recent and Future Advancements', Online:  
[https://www.researchgate.net/publication/225005053\\_UNICORE\\_6\\_-\\_recent\\_and\\_future\\_advancements](https://www.researchgate.net/publication/225005053_UNICORE_6_-_recent_and_future_advancements)
- [22] Project Jupyter Web page, Online:  
<https://jupyter.org/hub>



## Lecture Bibliography (3)

- [23] J. Haut, G. Cavallaro and M. Riedel et al., IEEE Transactions on Geoscience and Remote Sensing, 2019, Online: [https://www.researchgate.net/publication/335181248\\_Cloud\\_Deep\\_Networks\\_for\\_Hyperspectral\\_Image\\_Analysis](https://www.researchgate.net/publication/335181248_Cloud_Deep_Networks_for_Hyperspectral_Image_Analysis)
- [24] Apache Spark, Online: <https://spark.apache.org/>
- [25] YouTube Video, 'Neural Network 3D Simulation', Online: <https://www.youtube.com/watch?v=3JQ3hYko51Y>
- [26] A. Rosebrock, 'Get off the deep learning bandwagon and get some perspective', Online: <http://www.pyimagesearch.com/2014/06/09/get-deep-learning-bandwagon-get-perspective/>
- [27] J. Lange, G. Cavallaro, M. Goetz, E. Erlingsson, M. Riedel, 'The Influence of Sampling Methods on Pixel-Wise Hyperspectral Image Classification with 3D Convolutional Neural Networks', Proceedings of the IGARSS 2018 Conference, Online: [https://www.researchgate.net/publication/328991957\\_The\\_Influence\\_of\\_Sampling\\_Methods\\_on\\_Pixel-Wise\\_Hyperspectral\\_Image\\_Classification\\_with\\_3D\\_Convolutional\\_Neural\\_Networks](https://www.researchgate.net/publication/328991957_The_Influence_of_Sampling_Methods_on_Pixel-Wise_Hyperspectral_Image_Classification_with_3D_Convolutional_Neural_Networks)
- [28] G. Cavallaro, Y. Bazi, F. Melgani, M. Riedel, 'Multi-Scale Convolutional SVM Networks for Multi-Class Classification Problems of Remote Sensing Images', Proceedings of the IGARSS 2019 Conference, to appear

