Executing Engineering Pipelines on Exascale Systems

Lubomír Říha, Tomáš Brzobohatý and Jan Martinovič

IT4Innovations
national01$#&0
supercomputing
center@#01%101

Scientific Pipelines for Exascale

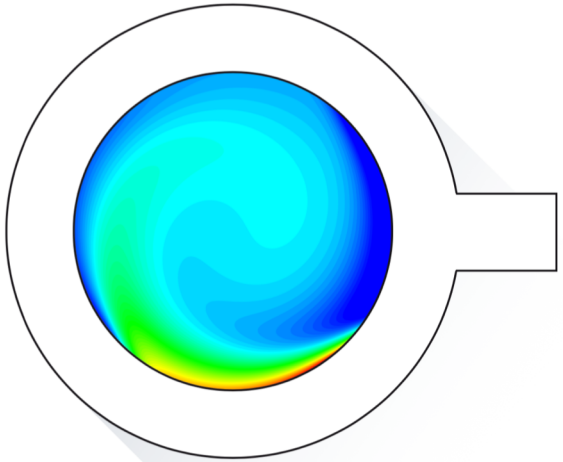
Vision

Combination of …

Scientific Pipelines for Exascale

Vision

Combination of **petascale application** and …

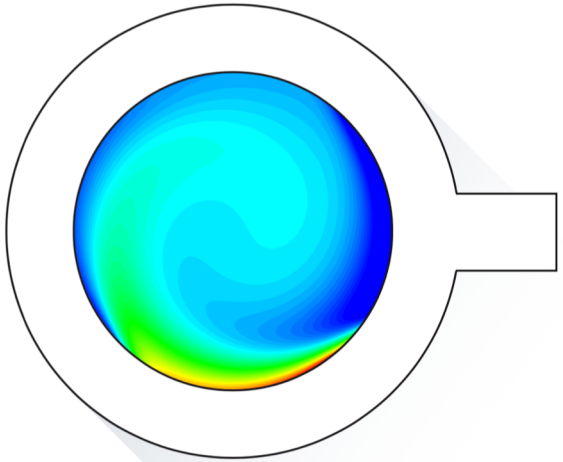


ESPRESO

Scientific Pipelines for Exascale

Vision

Combination of **petascale application** and **optimal scheduler** for optimal utilization of Exascale machine.



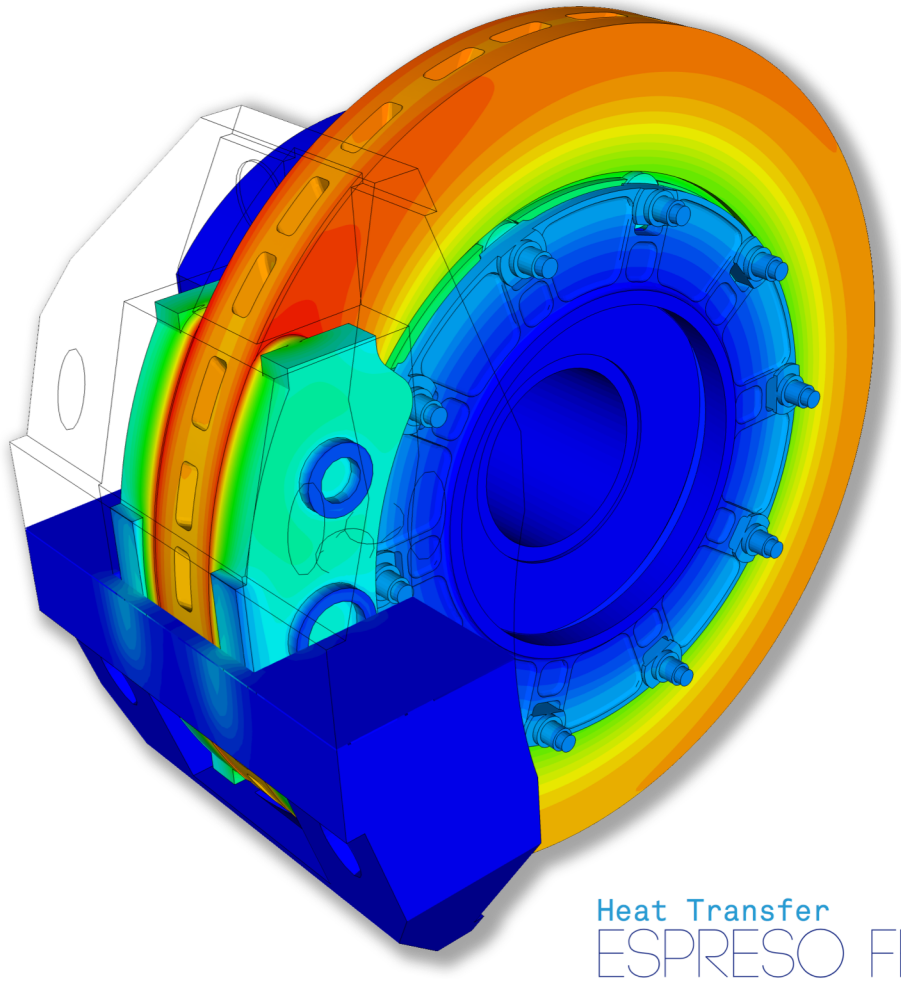
ESPRESO

HyperLoom

Part 1 – Petascale Application

ESPRESO

- ESPRESO vision is to create complex, free of charge, open source finite element tool designed from scratch for HPC environment
- More generally applicable than standard open source FEM codes for real industrial problems
- Connection to popular commercial codes (ANSYS) and other open source tools such as OpenFOAM and ELMER
- Contain highly scalable solvers capable to run on today's most powerful supercomputers
- "Solver as a Service" – connection to the IT4I HPC–as-a-Service infrastructure



ESPRESO Massively Parallel Framework for Engineering Applications

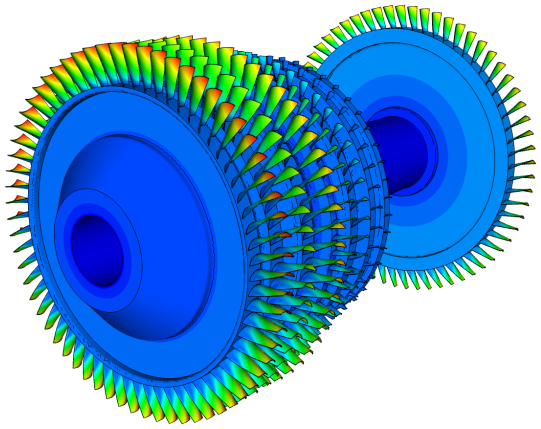
Commercial tools

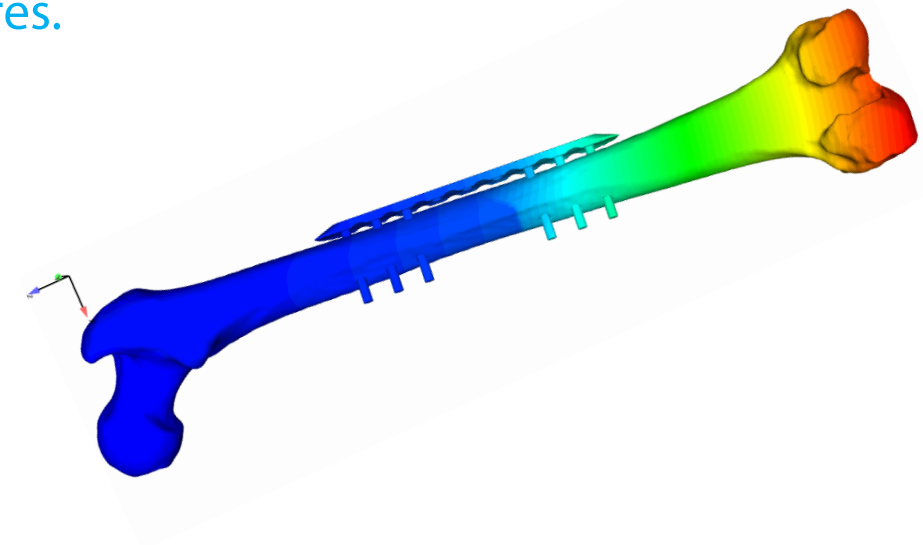
- No source code
- **Robust and complex pre-processing with GUI interface**
- Limited number of users/cores
- Commercial licenses are very expensive for HPC infrastructures

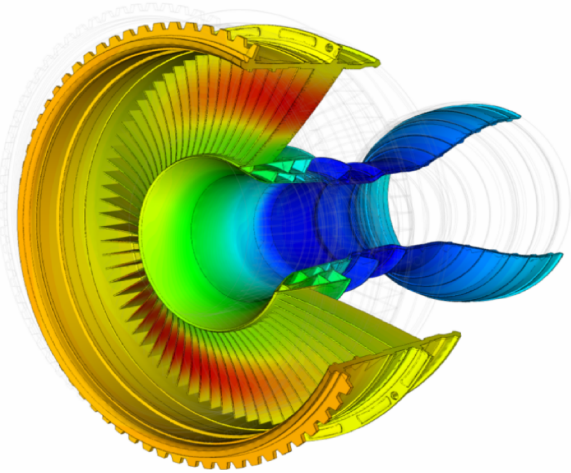
Open source tools

- Source code available for easy customization
- without complex GUI interface but simplest expandability and solver control
- **Unlimited number of users/cores**
- **Optimized solver for HPC infrastructures**

Combine it together! You obtain fast preprocessing and possibilities to solve large problems on hundreds of thousand of cores.

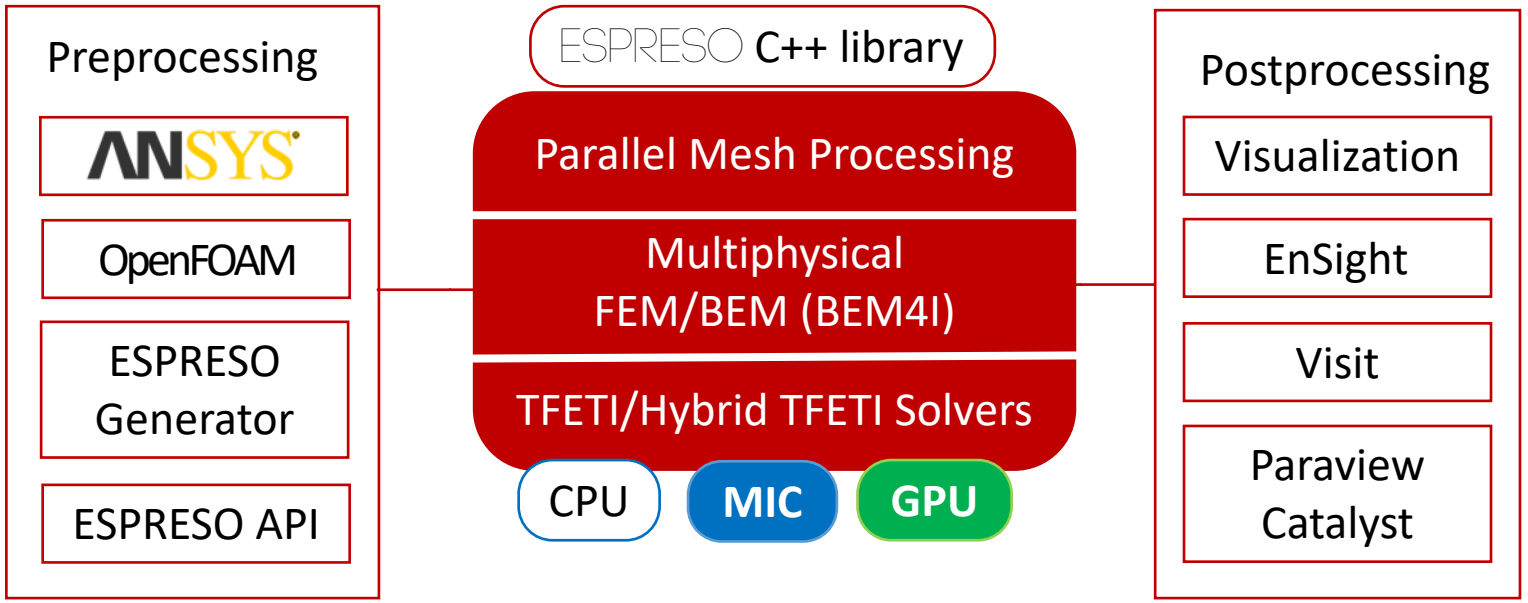






ESPRESO Massively Parallel Framework for Engineering Applications

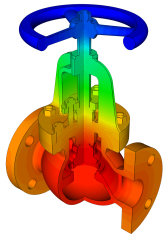
FEM/BEM framework for massively parallel machines with focus on parallel mesh pre-processing, physical solvers, non/linear solvers and post-processing.



ESPRESO Massively Parallel Framework for Engineering Applications

Heat Transfer Module Capability List:

Load steps definition for combination of multiple steady-state and time dependent analyses

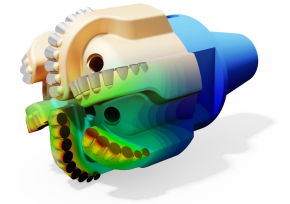


Transient solvers
- Generalized trapezoidal rule
- Automatic time stepping based on response frequency approach

Nonlinear solvers
- Newton Raphson – full and symmetric
- Newton Raphson with constant tangent matrices
- Line search damping
- Sub-steps definition
- Adaptive precision control for iterative solvers

Linear and quadratic finite element discretization

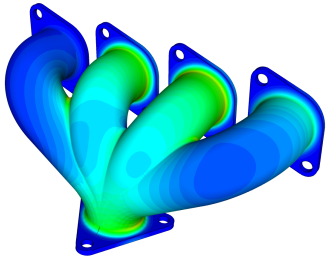
Gluing nonmatching grids by mortar discretization techniques



Complex material models
- nonlinear materials
- isotropic, orthotropic and anisotropic material models
- materials for phase change

Element coordinate system definition – cartesian, polar and spherical

Temperature and time dependent boundary conditions
- linear convection
- nonlinear convection
- heat flow
- heat flux
- diffuse radiation
- heat source
- translation motion



Consistent SUPG and CAU stabilization for Translation Motion (advection), Inconsistent stabilization

Phase Change based on apparent heat capacity method

Boundary element discretization for selected physical applications

Highly parallel multilevel FETI domain decomposition based solver for billions of unknowns for symmetric and non-symmetric systems with accelerators support and combination of MPI and OpenMP techniques
Asynchronous parallel I/O

Mesh processing - Domain Decomposition based on combination of the in house space filling curves algorithm and Metis/ParMetis - material separation- region separation,
RBF - Radial Basis Function mesh morphing for shape optimization

Input mesh format from popular open source and commercial packages like OpenFOAM, ELMER or ANSYS

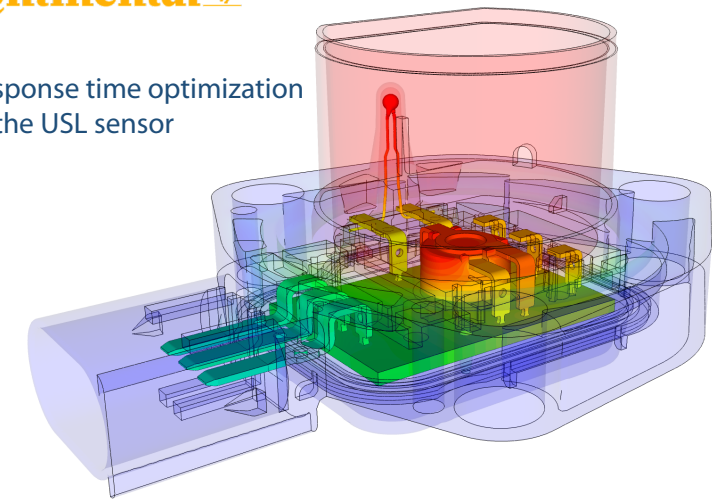
Output to commonly used post-processing formats, VTK and EnSight

Monitoring results on selected regions for statistic and optimization toolchain

Simple text Espreso Configuration File (ecf) for setting all ESPRESO FEM solver parameters without GUI. Control each parameter in ecf file from command line

Continental

Response time optimization of the USL sensor



ESPRESO Massively Parallel Framework for Engineering Applications

5th in TOP500 LIST

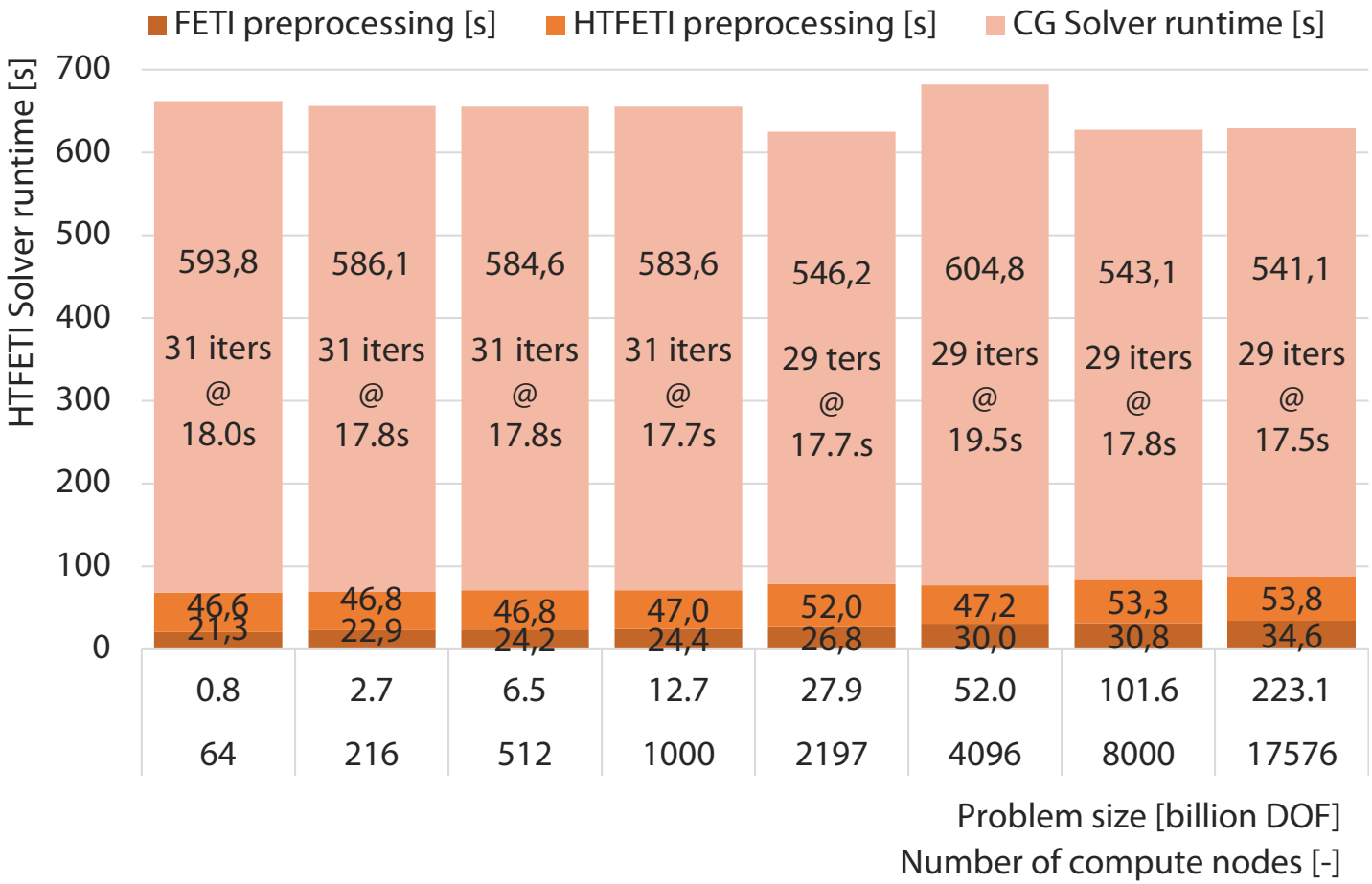
18,688 AMD Opteron 6274 16-core CPUs
18,688 NVIDIA Tesla K20X GPUs

2.7 million core hours dedicated to:
- scalability optimization of ESPRESO
- optimization of GPU accelerated version for large scale problems

OAK RIDGE National Laboratory

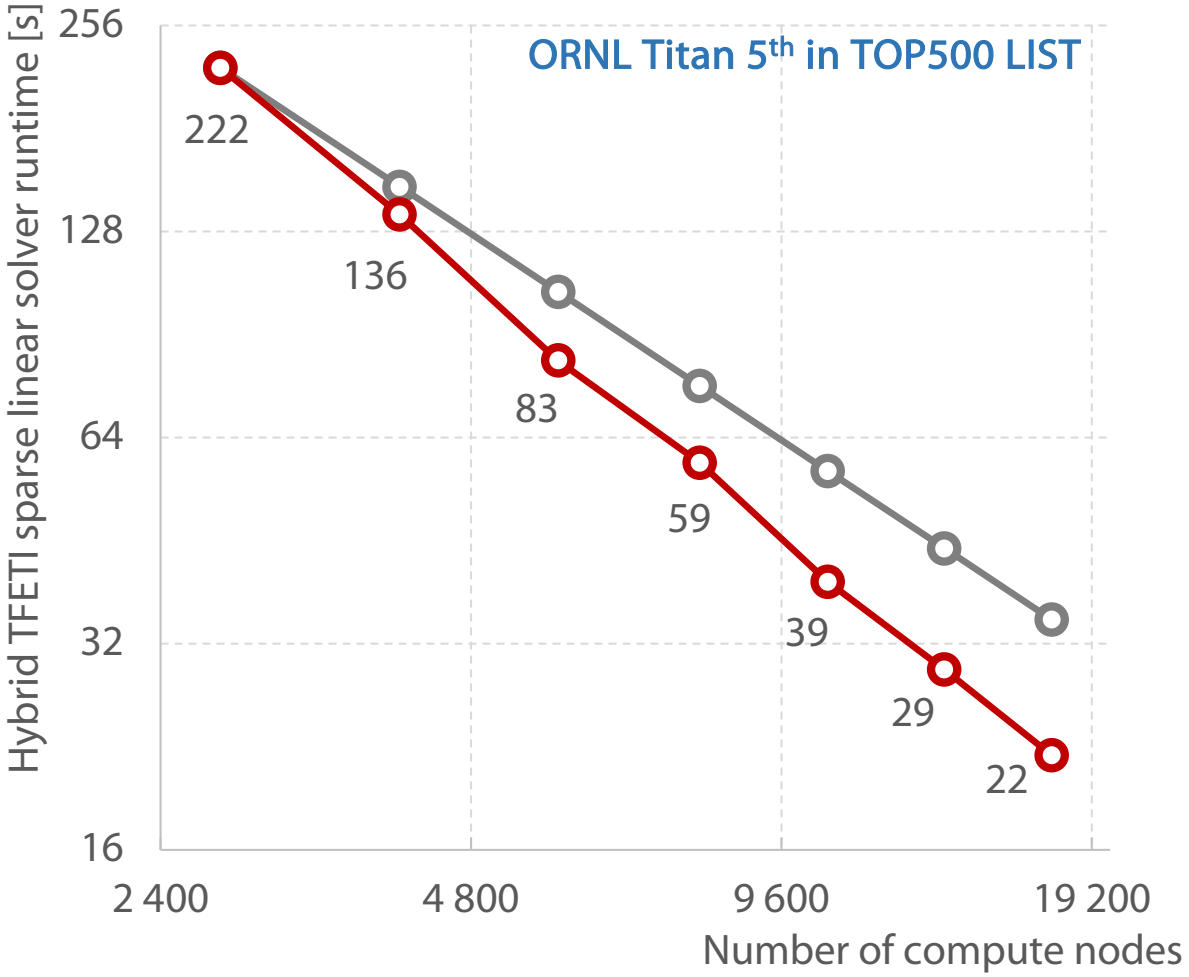
Weak Scalability Test

Up to **223 billion** DOF on 17576 Compute Nodes (281 216 cores)
Heat transfer (Laplace equation)

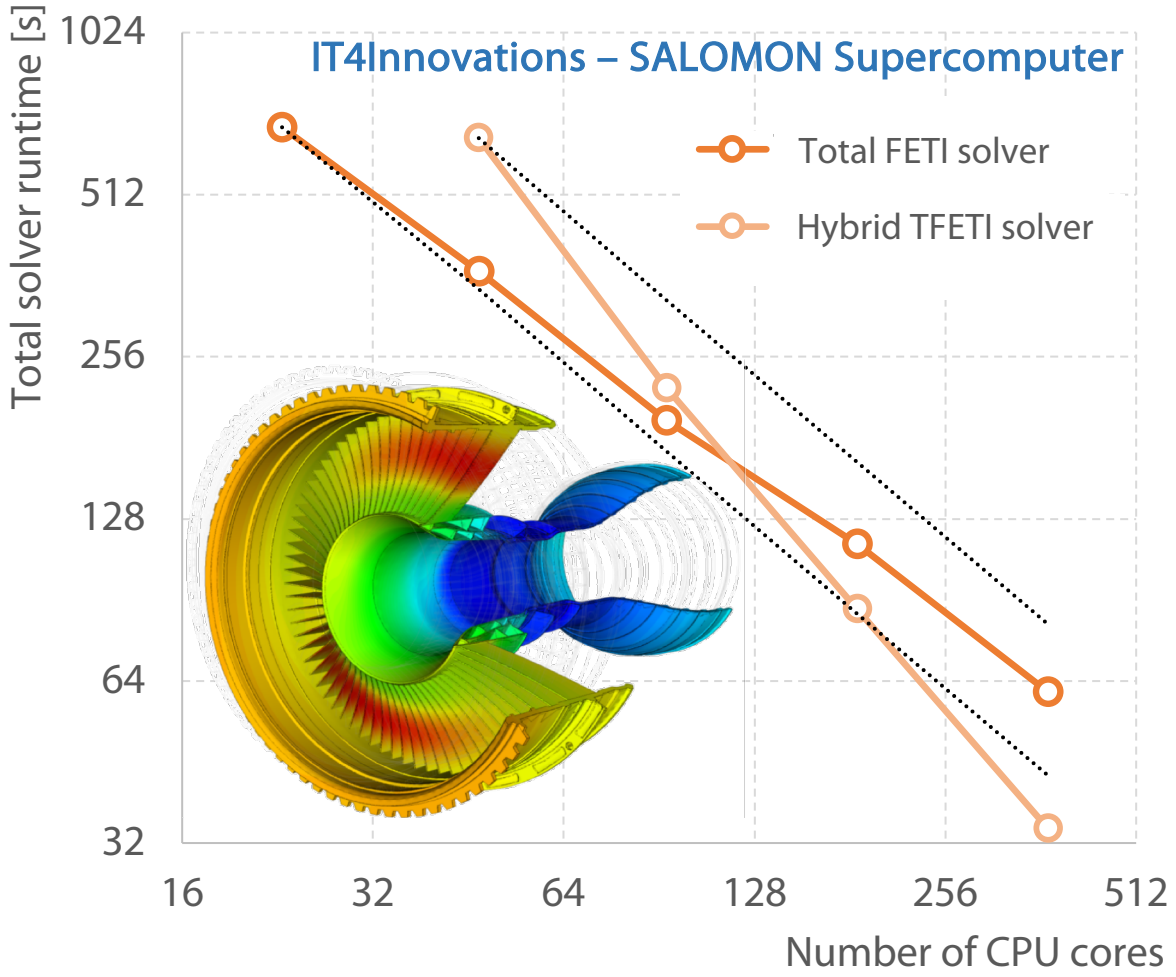


Strong scalability of Hybrid TFETI solvers

20 billion DOF on up to 17 576 Compute Nodes (**281 216 cores**)
3D heat transfer



300 million unknown - ANSYS Workbench real world problem
Linear elasticity – Hybrid FETI with Dirichlet preconditioner



Hardware Acceleration of FETI solvers

Local Schur Complement (LSC) Method

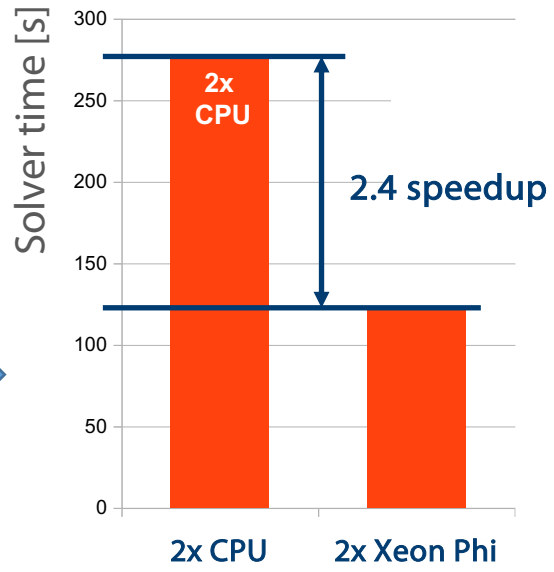
1: $r_0 := b - Ax_0$; $u_0 := M^{-1}r_0$; $p_0 := u_0$
2: **for** $i = 0, \ldots, m-1$ **do**
3: $s := Ap_i$
4: $\alpha := \langle r_i, u_i \rangle / \langle s, p_i \rangle$
5: $x_{i+1} := x_i + \alpha p_i$
6: $r_{i+1} := r_i - \alpha s$
7: $u_{i+1} := M^{-1} r_{i+1}$
8: $\beta := \langle r_{i+1}, u_{i+1} \rangle / \langle r_i, u_i \rangle$
9: $p_{i+1} := u_{i+1} + \beta p_i$
10: **end for**

Algorithm modification:

Converting sparse data structures to dense for optimal hardware utilization

Pre-processing – K factorization
1.) $x = B_1^T \cdot \lambda$ - SpMV
2.) $y = K^{-1} \cdot x$ - solve
3.) $\lambda = B_1 \cdot y$ - SpMV
4.) stencil data exchange in λ
- MPI – Send and Recv
- OpenMP – shared mem. vec

Pre-processing - $S_c = B_1 K^{-1} B_1^T \rightarrow$ MIC
1.) $\lambda \rightarrow$ MIC - **PCIe transfer from CPU**
2.) $\lambda = S_c \cdot \lambda$ - **DGEMV, DSYMV on MIC**
3.) $\lambda \leftarrow$ MIC - **PCIe transfer to CPU**
4.) stencil data exchange in λ
- MPI – Send and Recv
- OpenMP – shared mem. vec



Hardware Acceleration of FETI solvers

Processor/Accelerator	Number of cores	Peak floating point performance SP/DP [GFLOPS]	Memory bandwidth [GB/s]	Memory type
Intel Xeon E5-2680v3	12	960/480	**68**	DDR4
Intel Xeon Phi 7120p	61	2420/1210	**180 (352*)**	GDDR5
Intel Xeon Phi 7210	64	5325/2662	**102 /400**	DDR4/MCDRAM
NVIDIA Tesla K80 (1 chip)	1597 SP/533 DP	4365/1455	**240**	GDDR5
NVIDIA Tesla K20X (Titan)	2688 SP / 896 DP	3950/1310	**250**	GDDR5
NVIDIA Tesla P100 (PCIe)	3584 SP / 1792 DP	8345/4217	**720**	HBM2

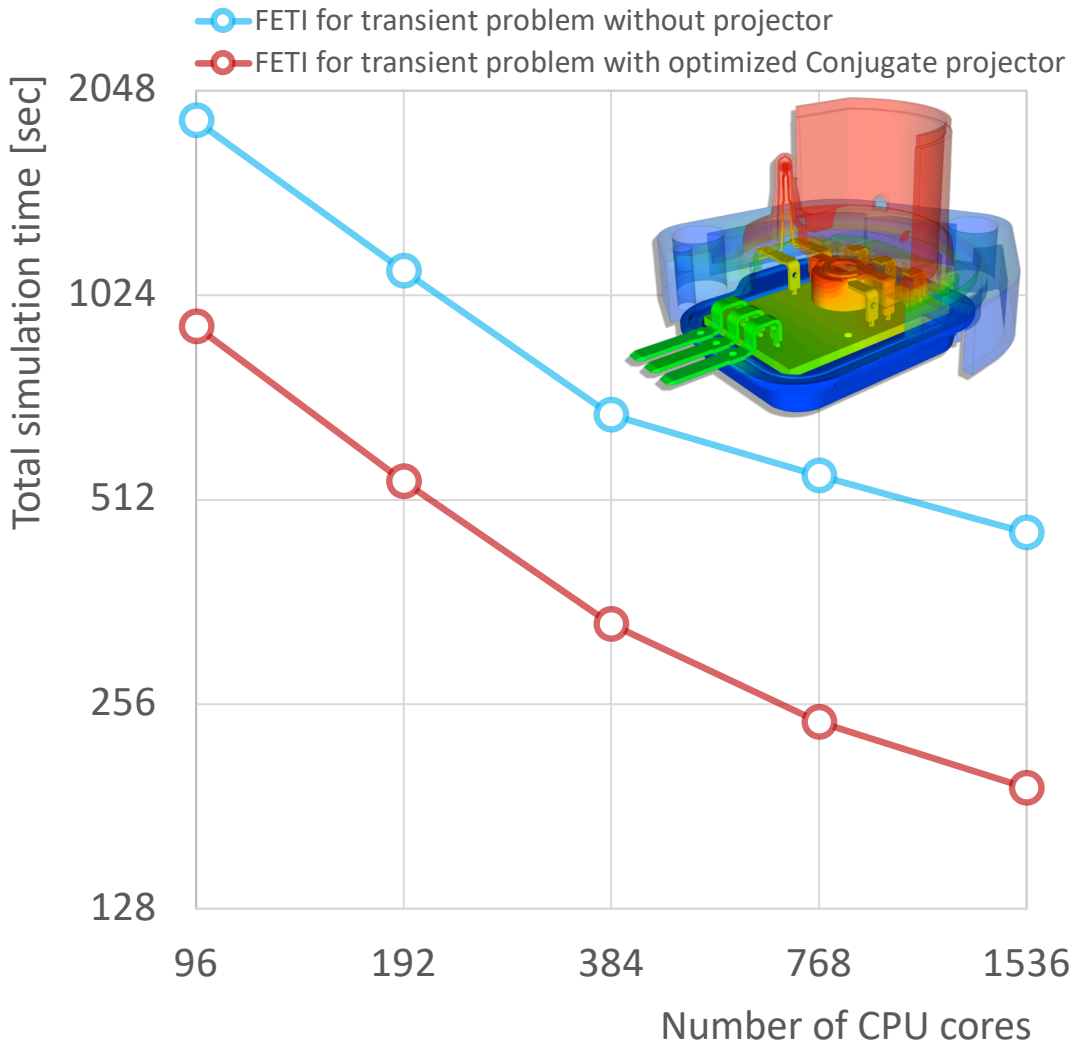
Performance of LSC Method - Heat Transfer problem

Solver	Problem size and decomposition		Solve time[s]	Speedup by LSC method				
	number of subdomains [-]	Subdomain size [DOF]	PARDISO CPU only	**CPU only**	**KNC & CPU***	**KNL only**	**K80 & CPU***	**P100 & CPU***
Total FETI	512	6859	21.4	**1.3**	**2.1**	**2.2**	**1.5**	**4.0**
Hybrid Total FETI	512	6859	26.0	**1.2**	**2.4**	**2.5**	**1.6**	**3.5**

*** Note:** CPU does not process LSCs

ESPRESO Massively Parallel Framework for Engineering Applications

Full nonlinear transient simulation with adaptive time stepping

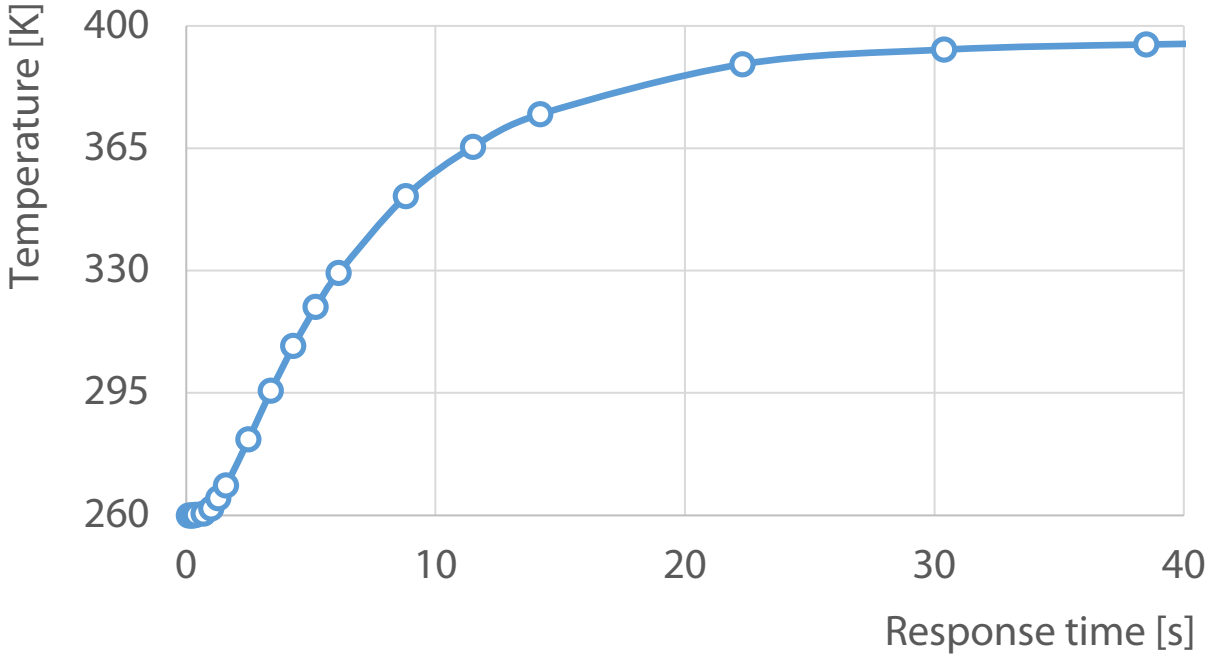


Still is possible to decrease the computational time by hardware optimization of the nonlinear steps, matrix assembler vectorization

Single and Adaptive Design of Experiment

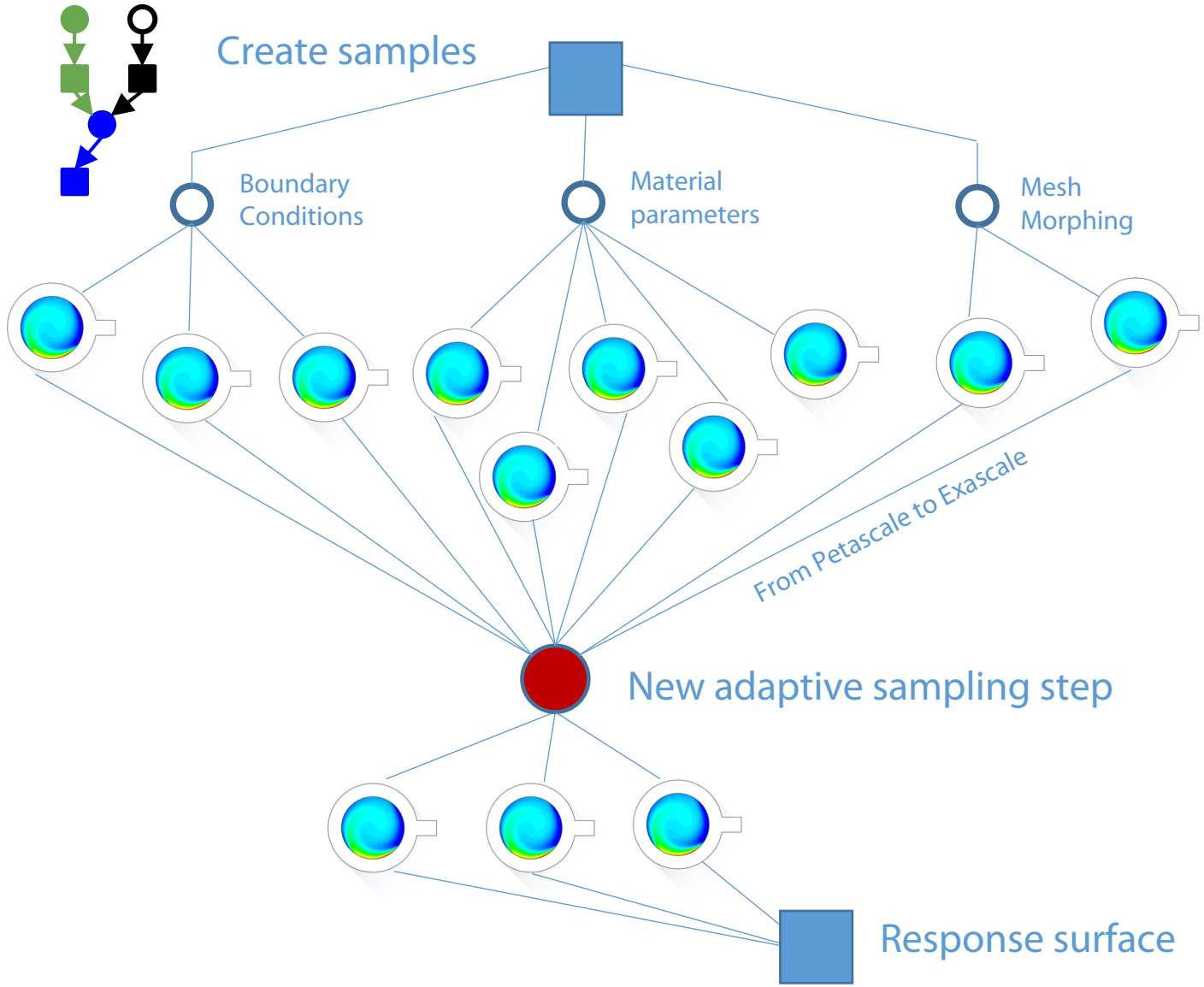
Response time optimization, optimization of material parameters, mesh morphing for shape optimization without remeshing

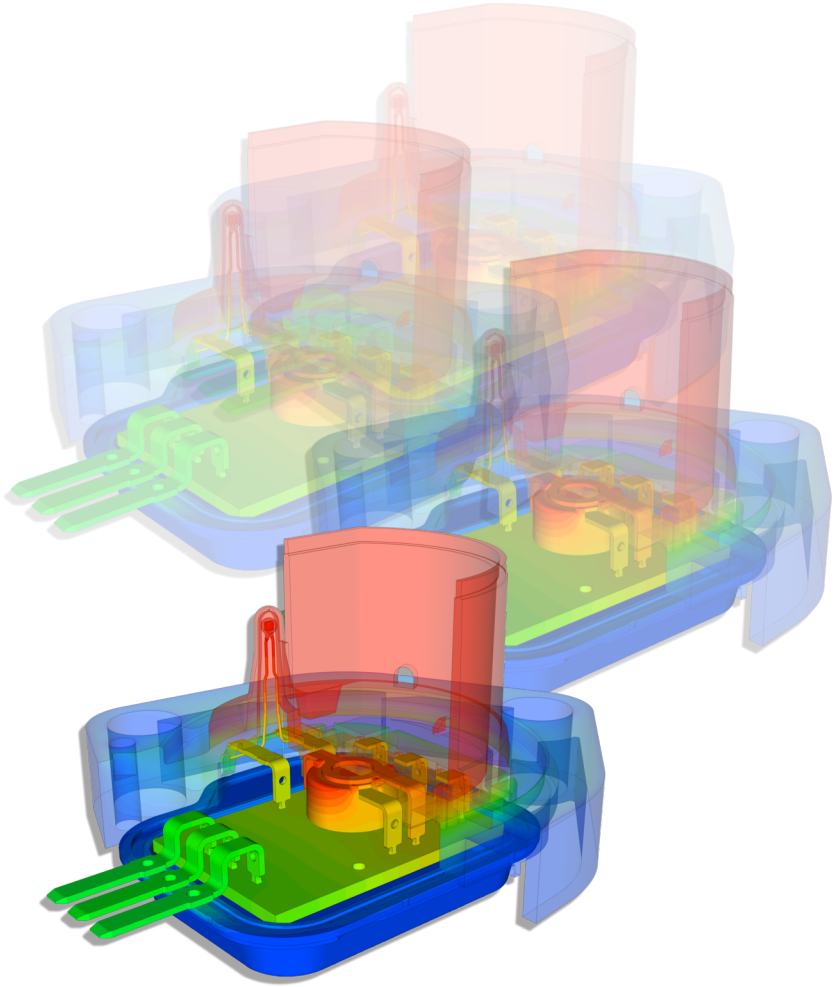
Leads to tens of simulations - Response surface in few minutes



mesh generated in ANSYS - solved by ESPRESO

Single and adaptive design of experiment - Leads to tens of simulations - response surface in few minutes



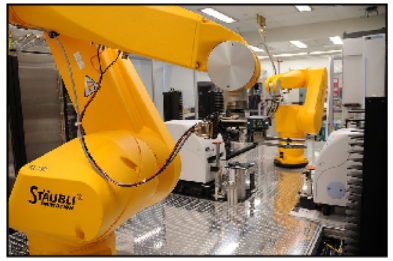


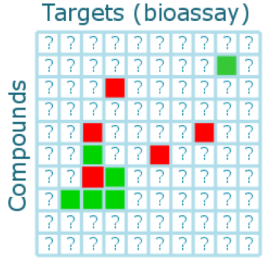
HyperLoom

A Platform for Defining and Executing Scientific Pipelines in Distributed Environments

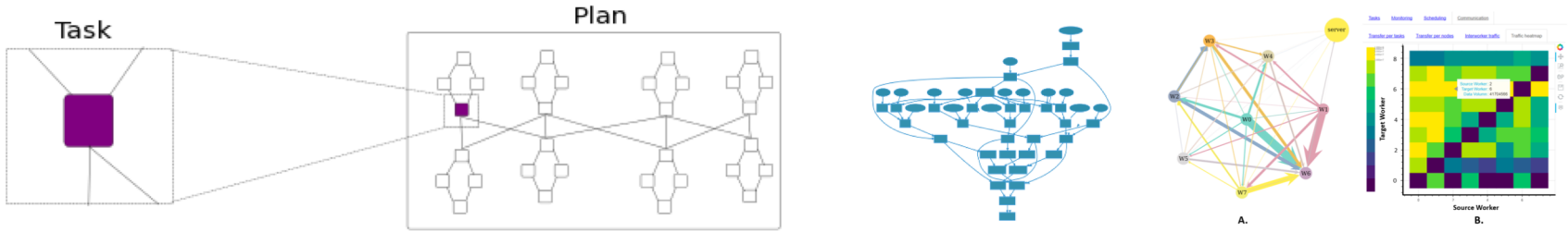
- high performance end-to-end data processing made simple
- initial motivation: **chemogenomics for novel drug discovery**

EXCAPE





HyperLoom - Task Distribution Framework



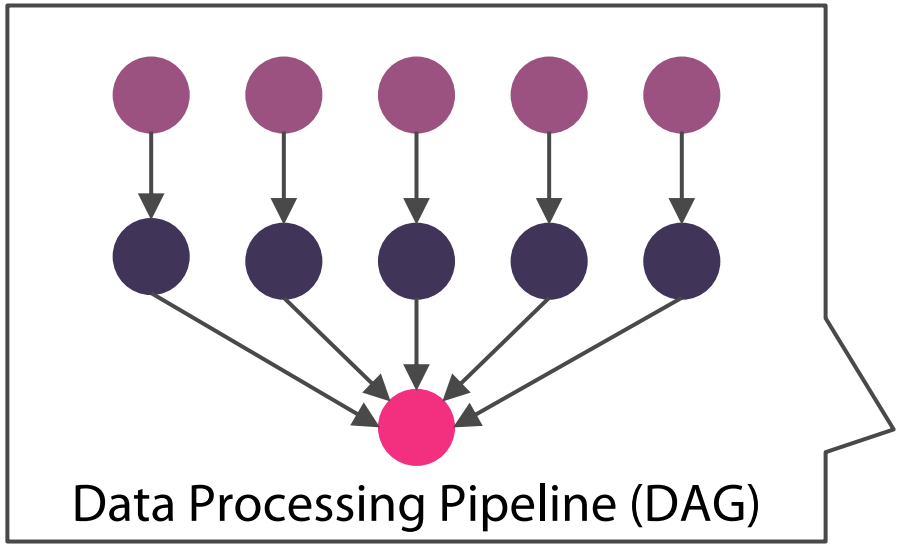
- Platform for **defining and executing workflow pipelines** in large-scale distributed environments
- Implements its own **scheduling algorithm optimized for execution of millions of interconnected tasks** on hundreds of computational nodes
- **Thin Python client module** that allows to easily define and execute the pipelines

- Research poster about HyperLoom has been accepted at supercomputing SC17 conference
  - Has been selected as one of the 9 Best Poster Candidates

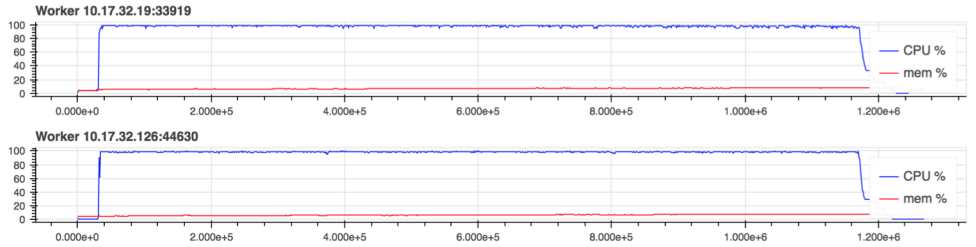
BSD license - https://HyperLoom.eu

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 671555

HyperLoom - Task Distribution Framework



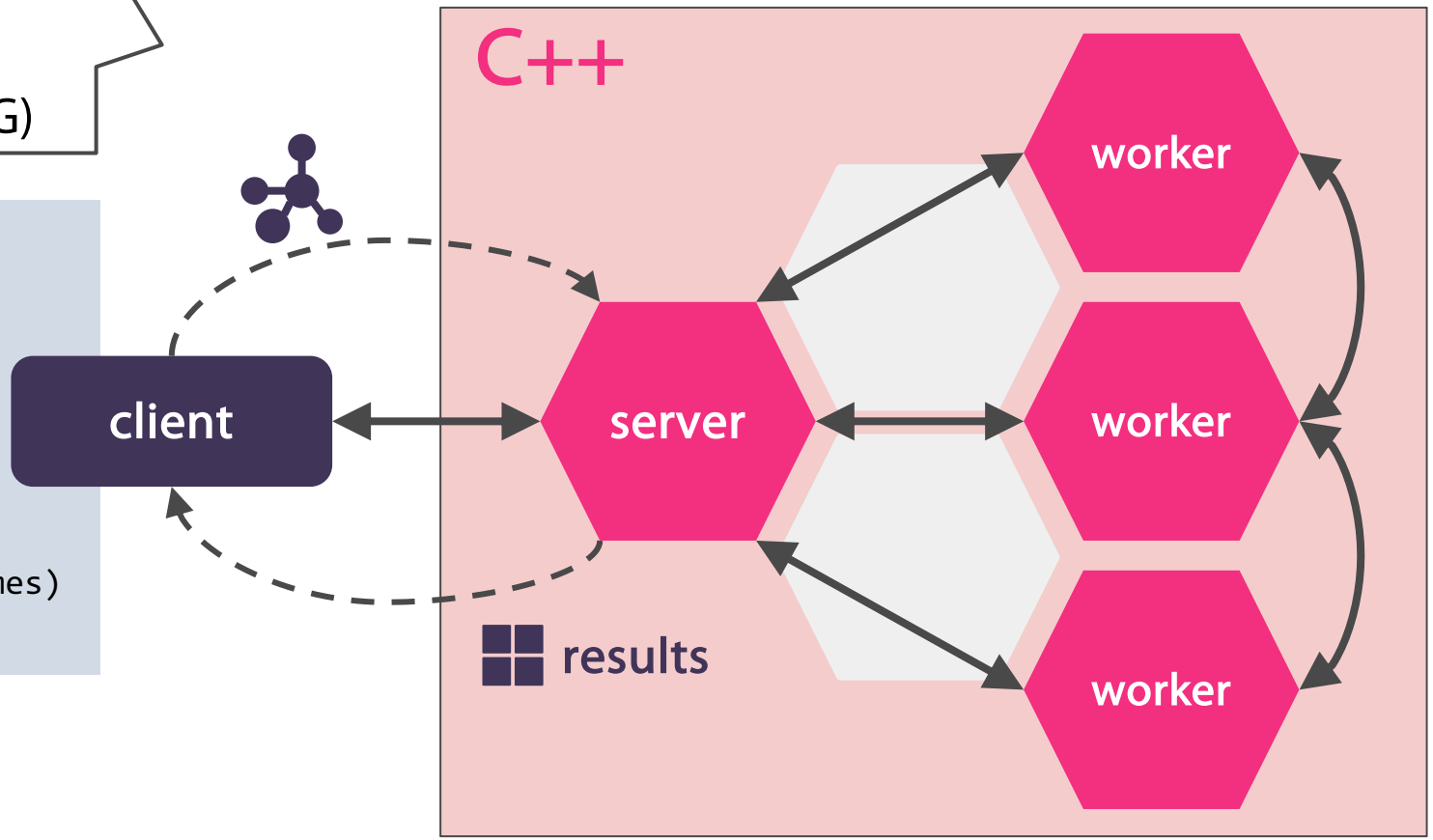
LORE profiler



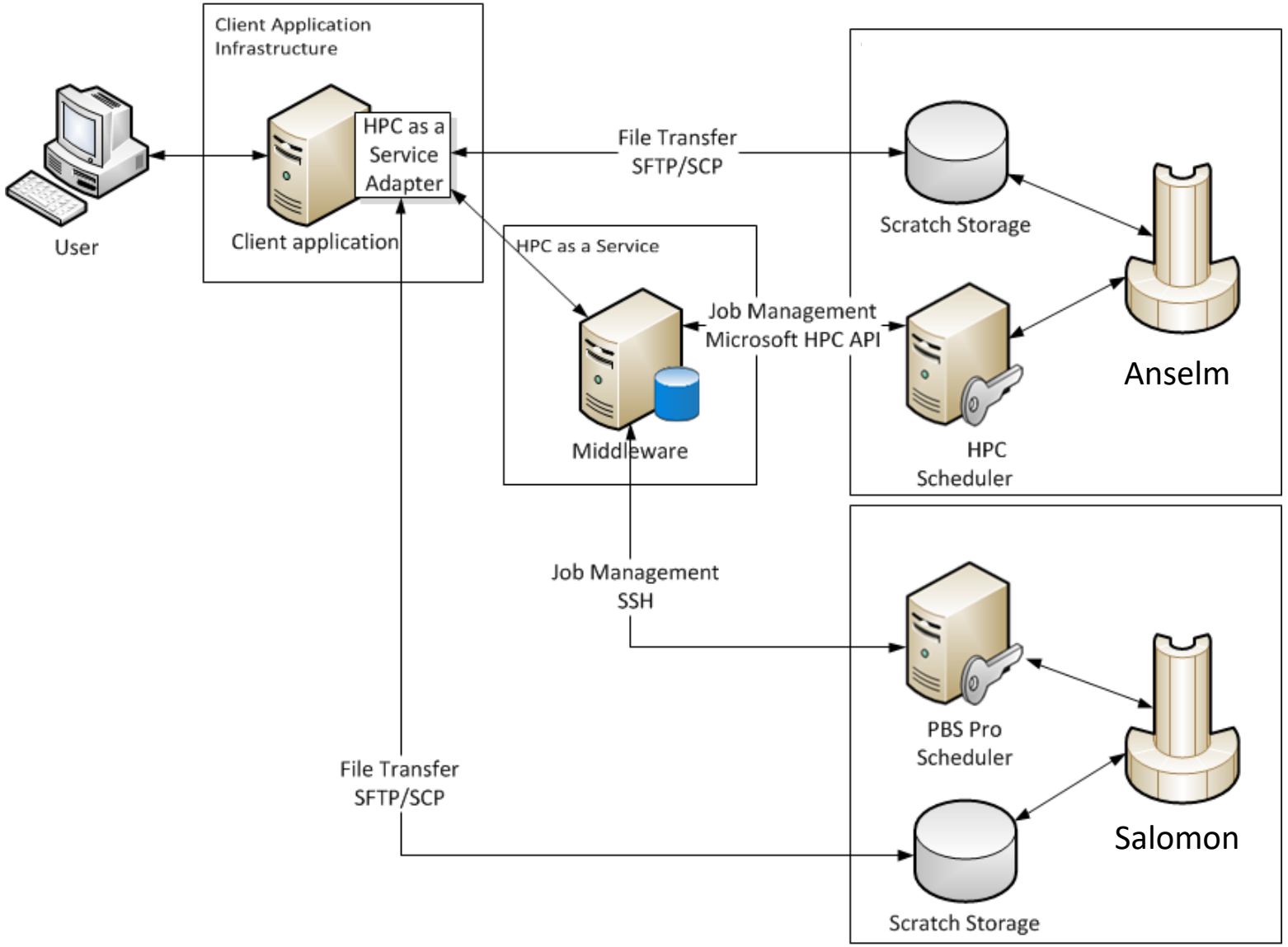

```python
import loom.client as lc
tasks = []
for i in range(9):
    t = lc.tasks.run("/bin/hostname")
    tasks.append(t)
merged_hostnames = lc.tasks.merge(tasks)

client = lc.Client("localhost", 9010)
future = client.submit_one(merged_hostnames)
result = future.gather()
```

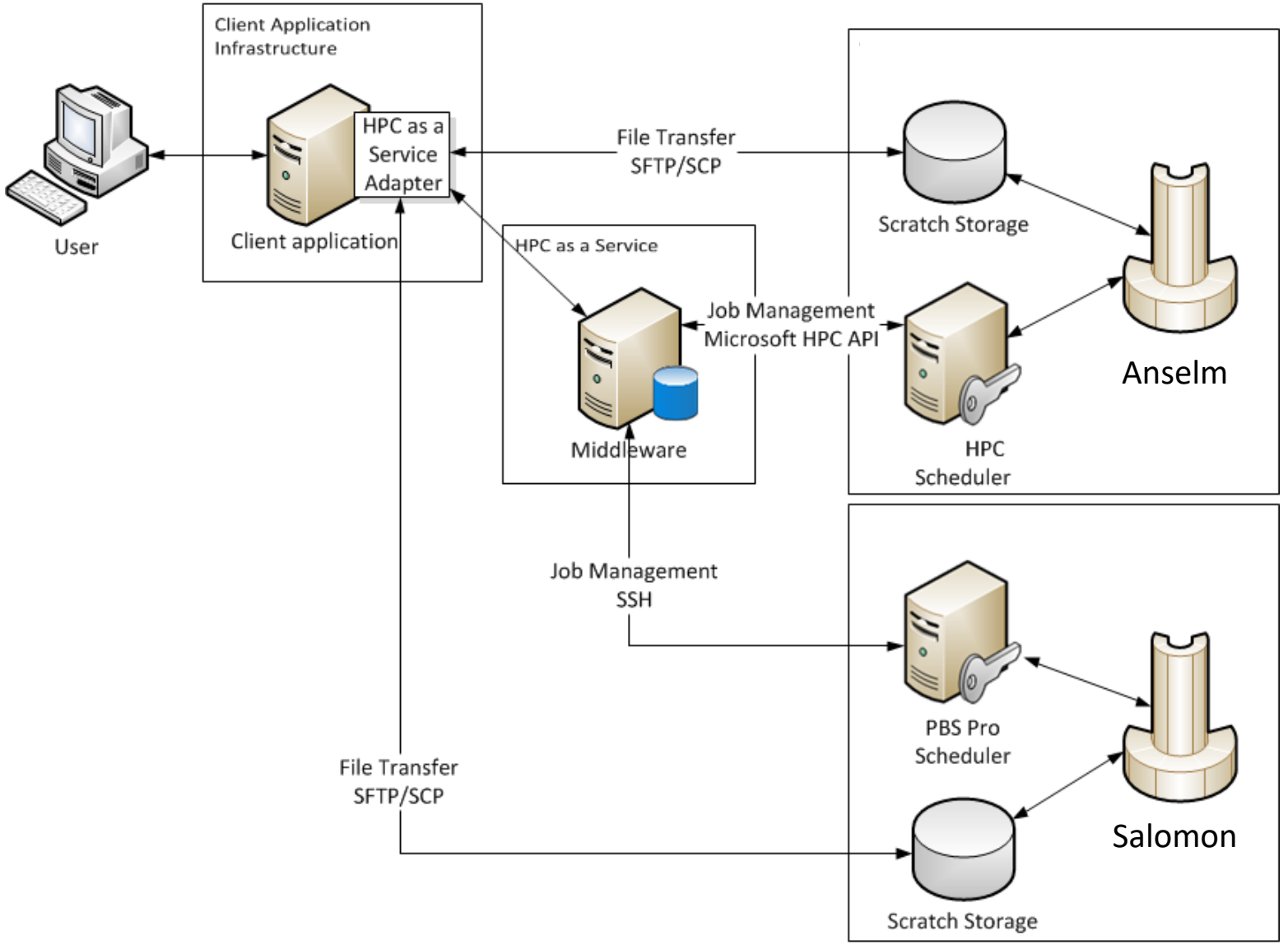
Python



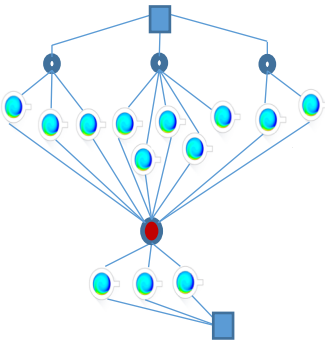
HPC as a Service



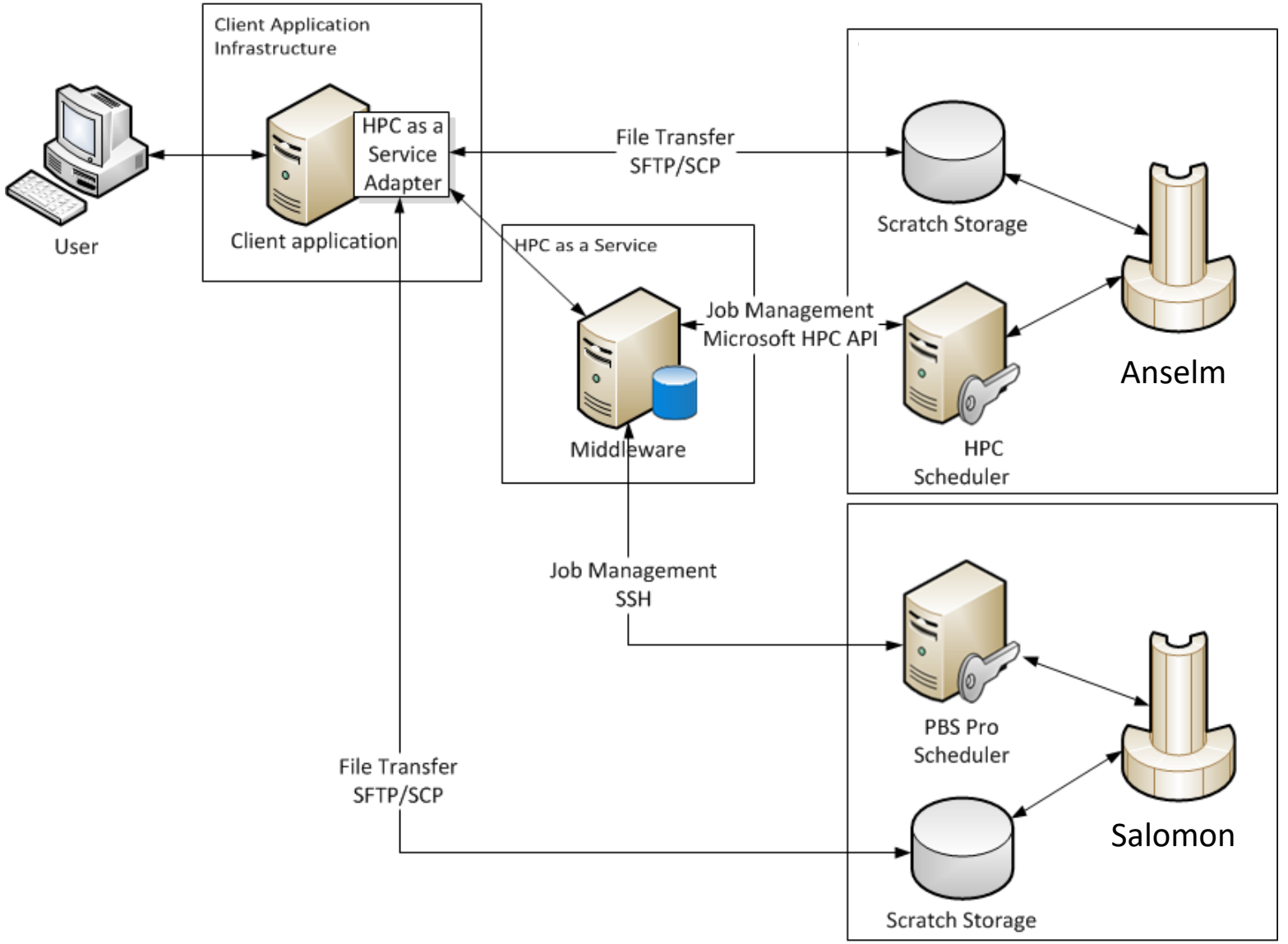
HPC as a Service & HyperLoom



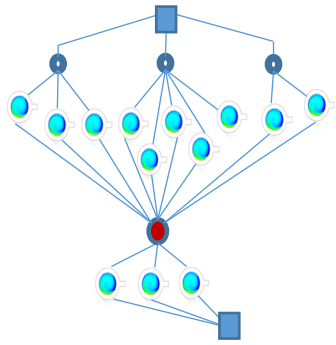
HyperLoom



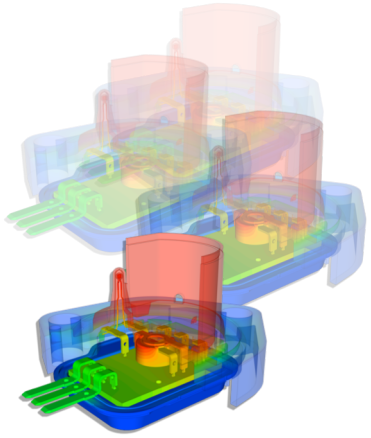
HPC as a Service & HyperLoom

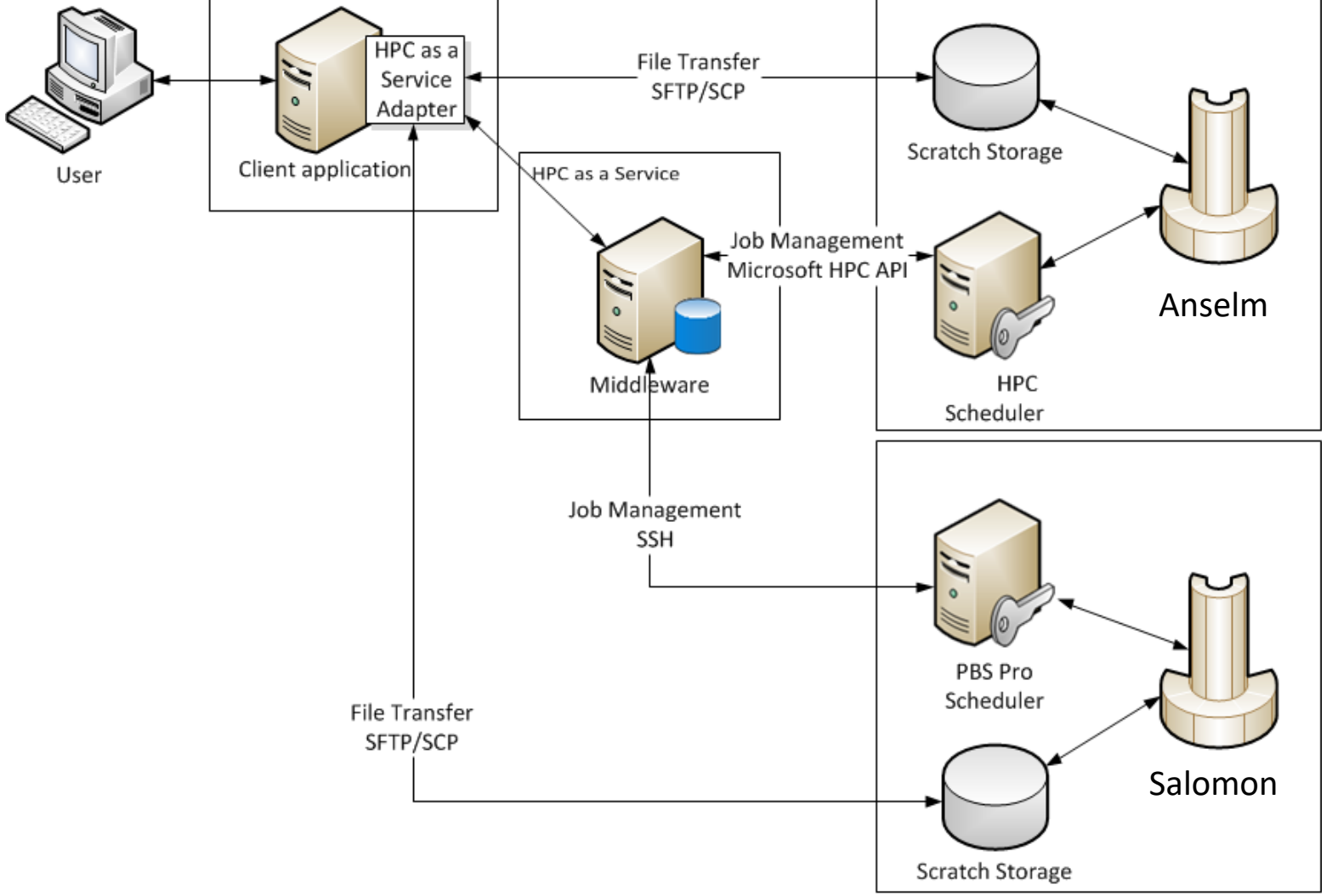


HyperLoom

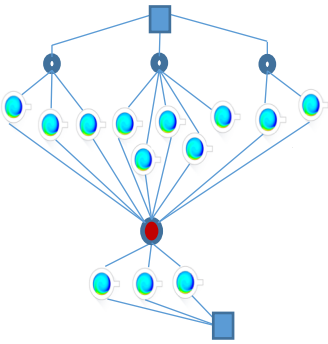


Solver as a Service & HyperLoom

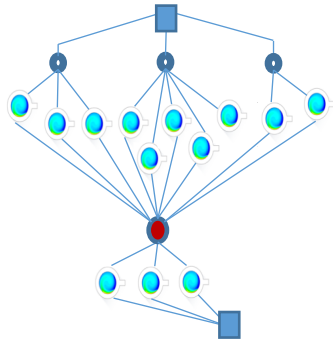




HyperLoom



HyperLoom



Thank you

IT4Innovations
national01$#&0
supercomputing
center@#01%101