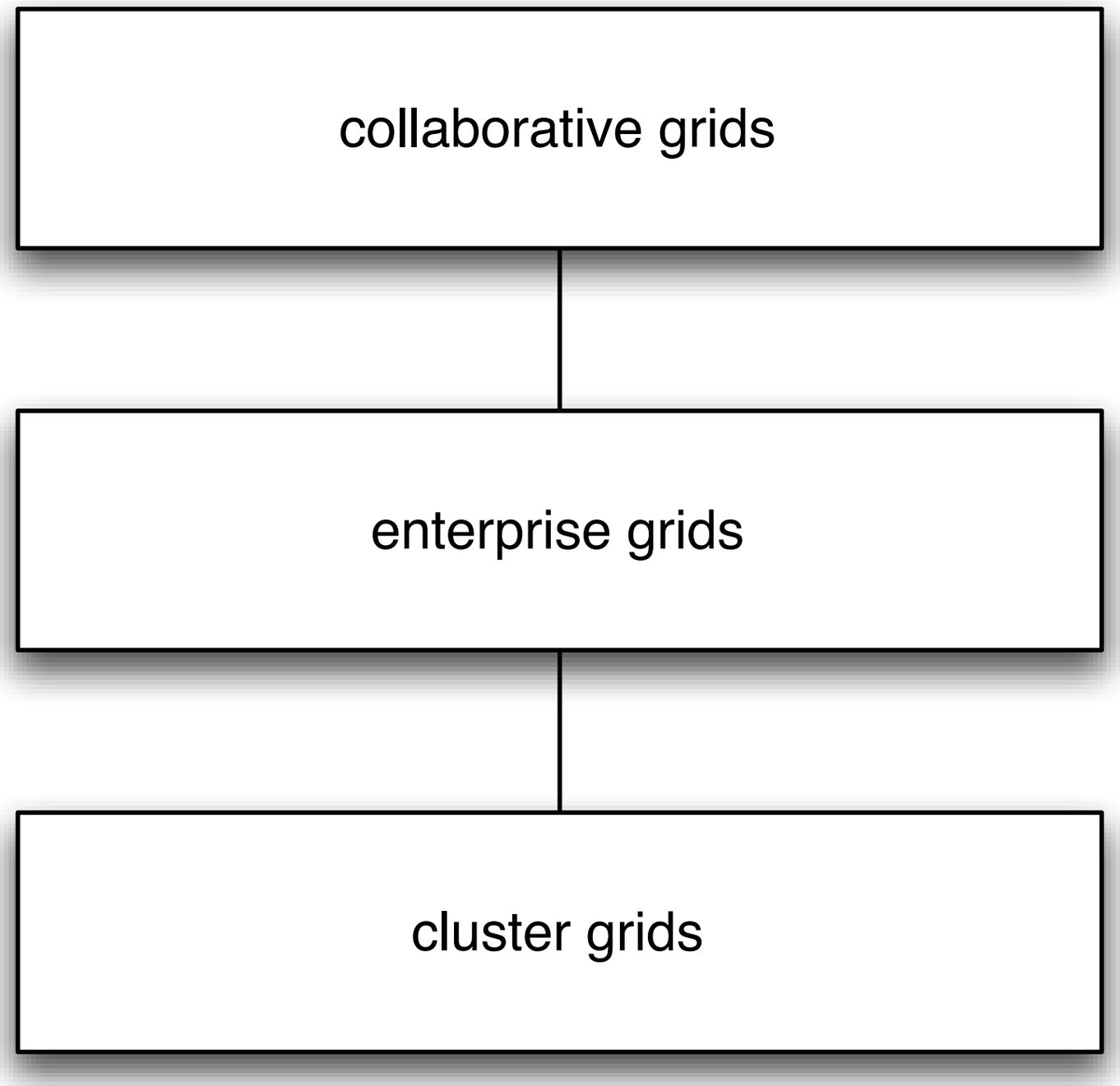
Requirements for Enterprise Grid Applications

Mathias Dalheimer
dalheimer@itwm.fhg.de

I am with the Competence Center for High-Performance Computing at the Fraunhofer ITWM, Kaiserslautern, Germany.

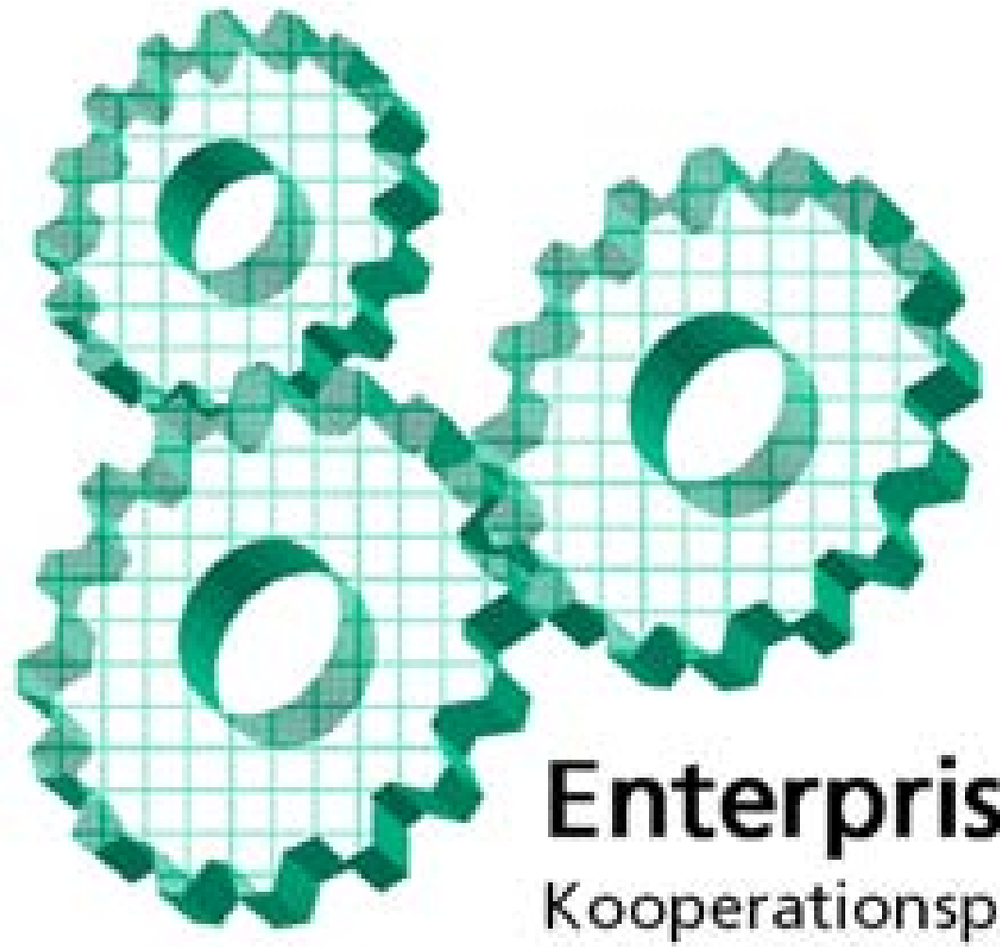
Grid definition

definition of grid in this talk:
- there are three grid definitions around



- collaborative grids: multi-institutional, general purpose grids (EGEE)
- cluster grids: provide a seamless access to different clusters in a homogeneous environment, same file system, same userids
- enterprise grids: grid technology used within a single enterprise (but maybe accross departments)

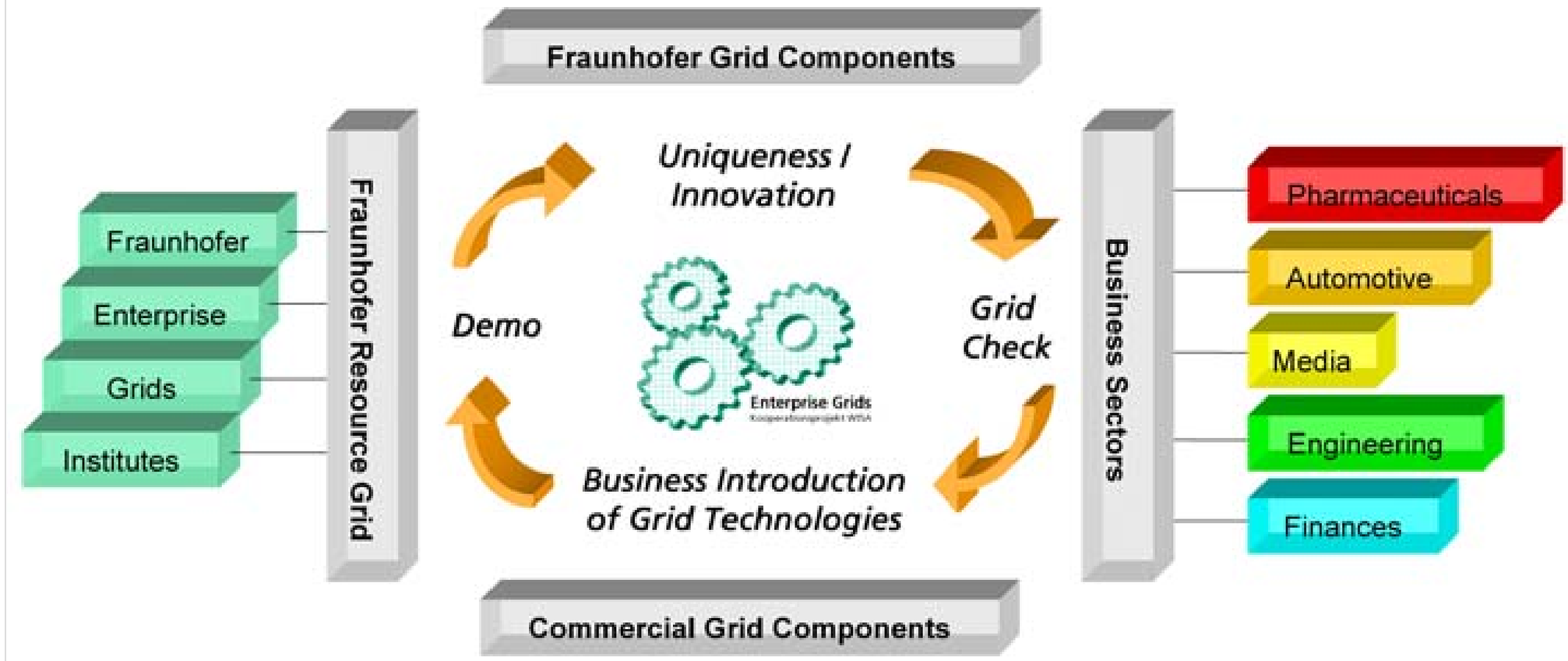
-> there is only one entity which can enforce the policy, everything else is as complicated as with science grids (data transfers)



Enterprise Grids

Kooperationsprojekt WISA

Enterprise Grid Project – an internal project with four Fraunhofer institutes to bring our grid expertise on the market.
Fraunhofer SCAI, ITWM, IAO, FIRST.



Enterprise Grid Project brings our competences together: we have both the experience with grid technologies (Fraunhofer Resource Grid) and the applications (we are developing various HPC applications inhouse and we also use a wide range of applications).

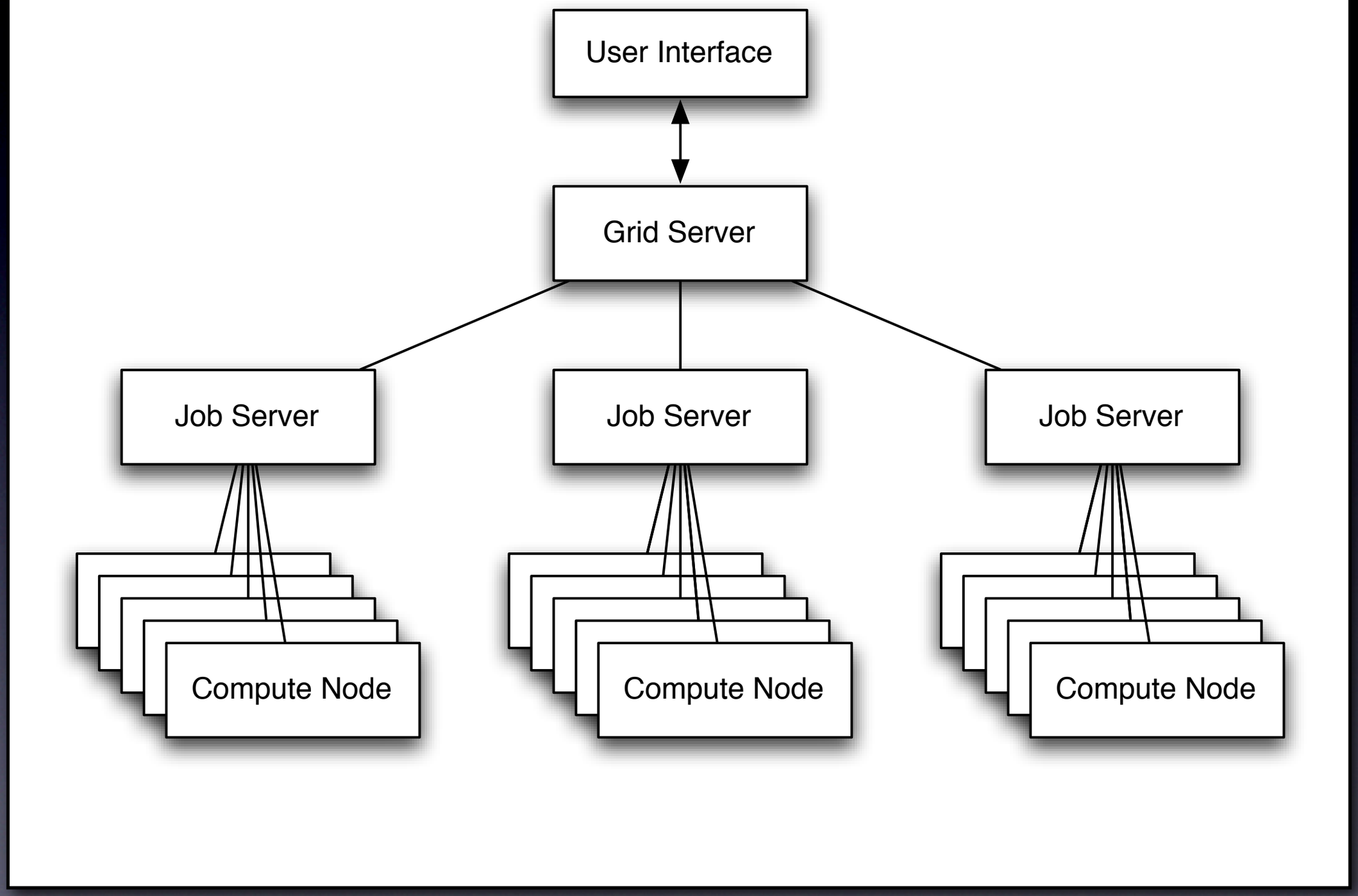
In addition, we know commercial and open source grid components and we are actively developing grid technologies.

before I start: this is a selection of our current projects. I can't give you all detail, but all projects are real.

Fraunhofer

PHASTGrid

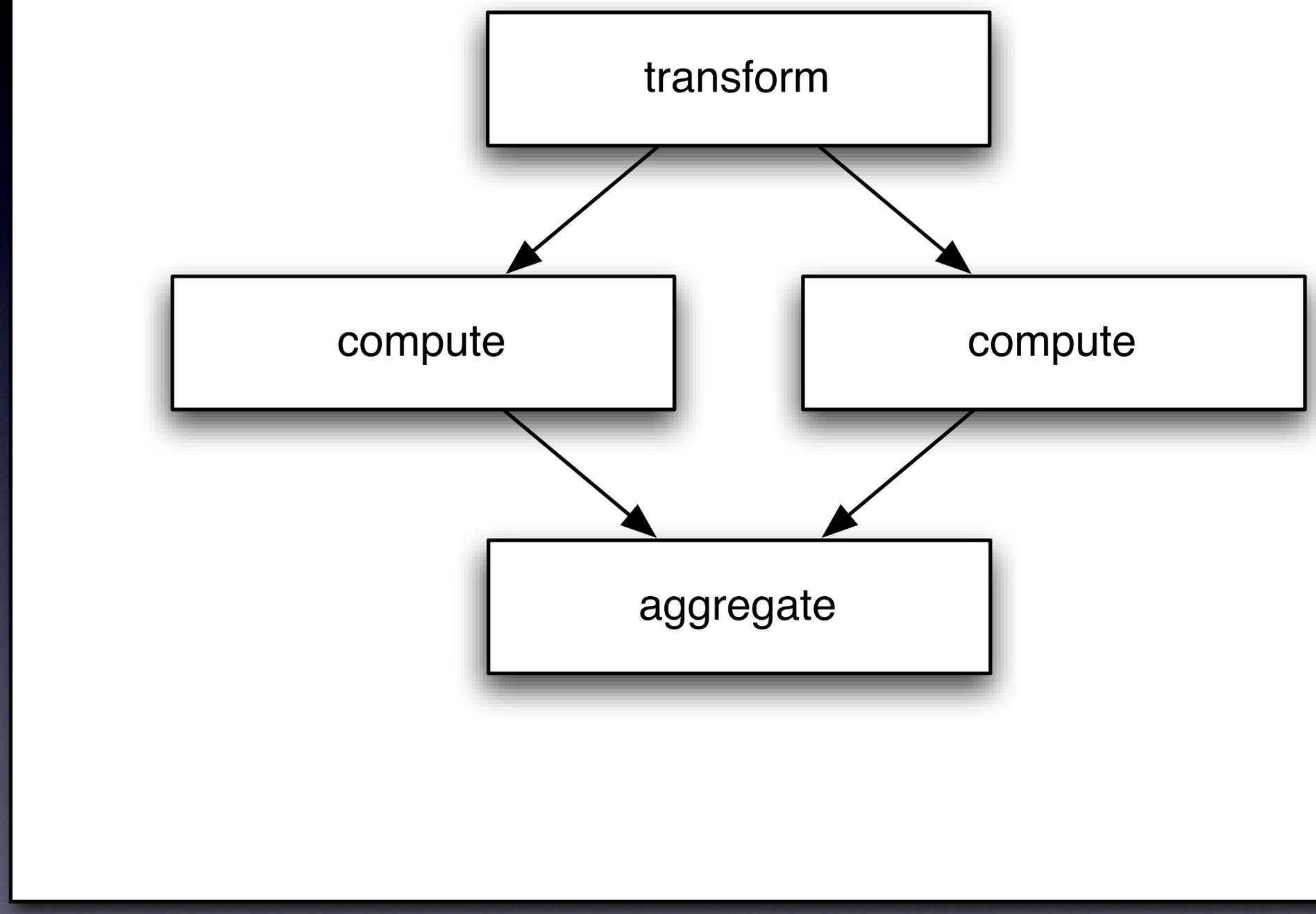
Before I start with the use cases, I'd like to introduce our PHASTGrid product
- simple grid middleware



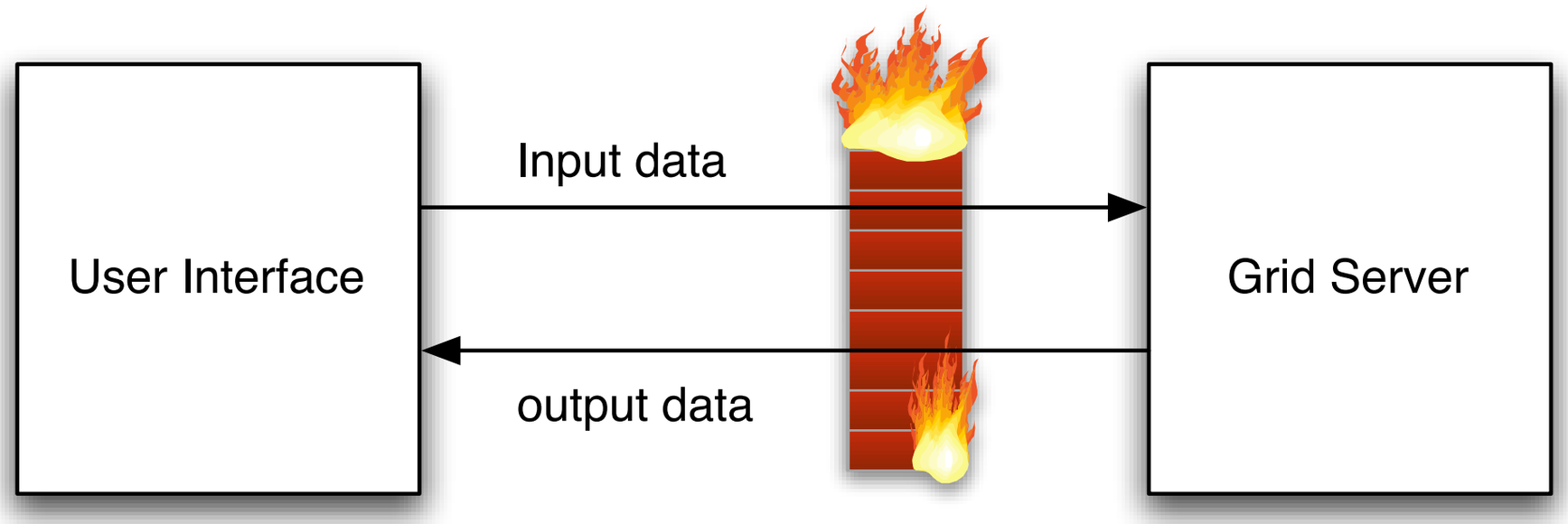
PHASTGrid exposes a web service so that users can interact with the grid.
- incoming jobs are distributed amongst the available job servers
- job servers manage a collection of compute nodes
- compute nodes might be a cluster system or a pool of workstations
- Not shown: Redundancy of gridserver, data management layer etc.
- one feature: the job server-compute node segment can be run in a HA-mode: upon the failure of the job server, a new one is elected amongst the compute nodes.

Explicit Job Integration

- It is not possible to run arbitrary binaries directly.
- Each application needs to be integrated into the middleware.
- The application then exposes an webservice to the outer world - all operations can be mapped this way.
- The application is fully tested and we are sure that it works.

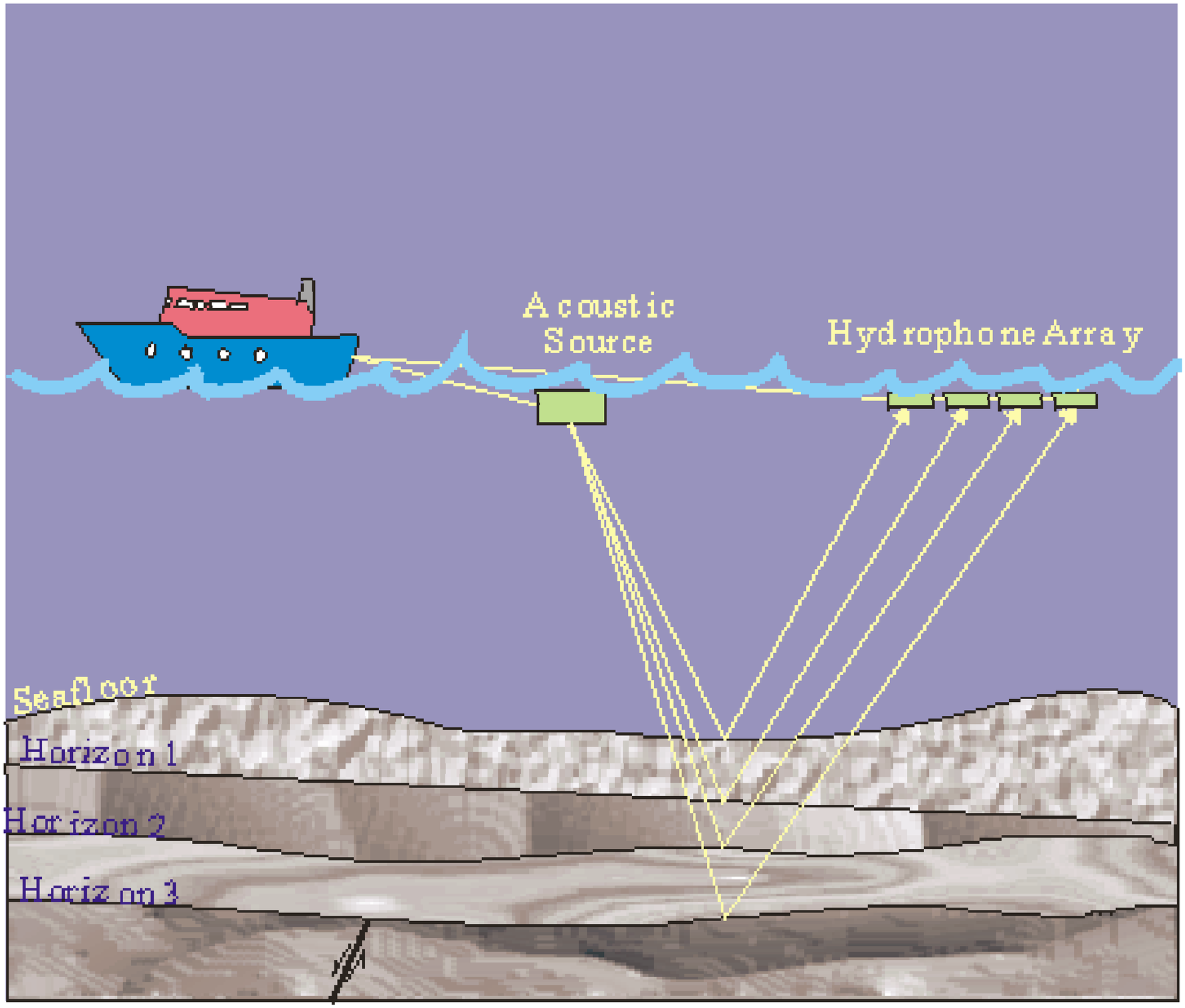


PHASTGrid also has a simple job lifecycle:
- Parallelization is considered within the framework. We support three steps:
(1) the incoming request is transformed, i.e. splitted into several subjobs.
(2) the subjobs are then computed separately on different worker nodes.
(3) in the aggregate step, the results are collected and combined.
-> This works especially well for massively parallel tasks, e.g. MC Simulations



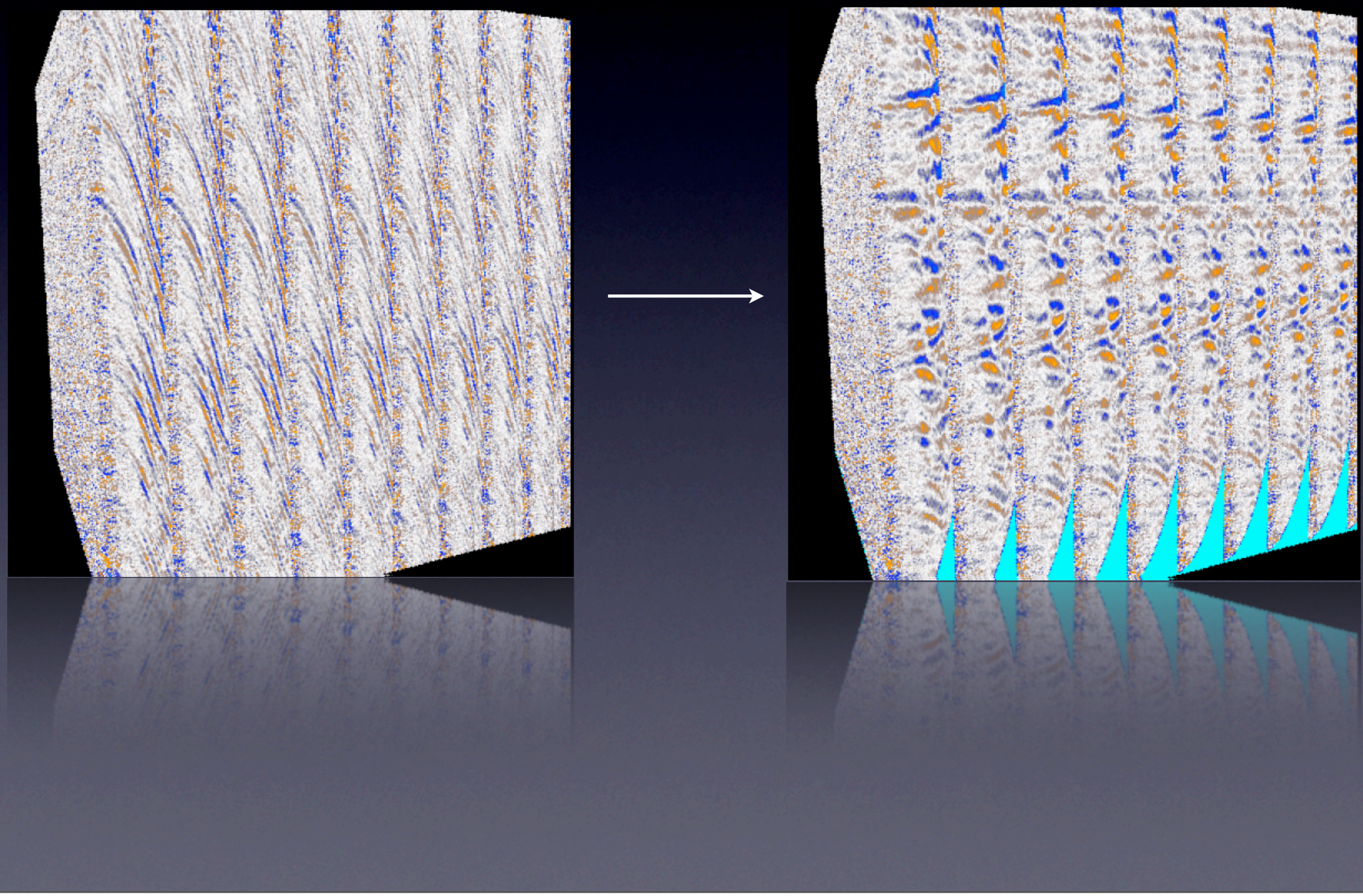
clean integration of application into the framework
- application appears as a web service at the frontend.
- we can ensure that the application works within the backend
- we can handle security at the grid server and trust the backend infrastructure

Seismic image processing

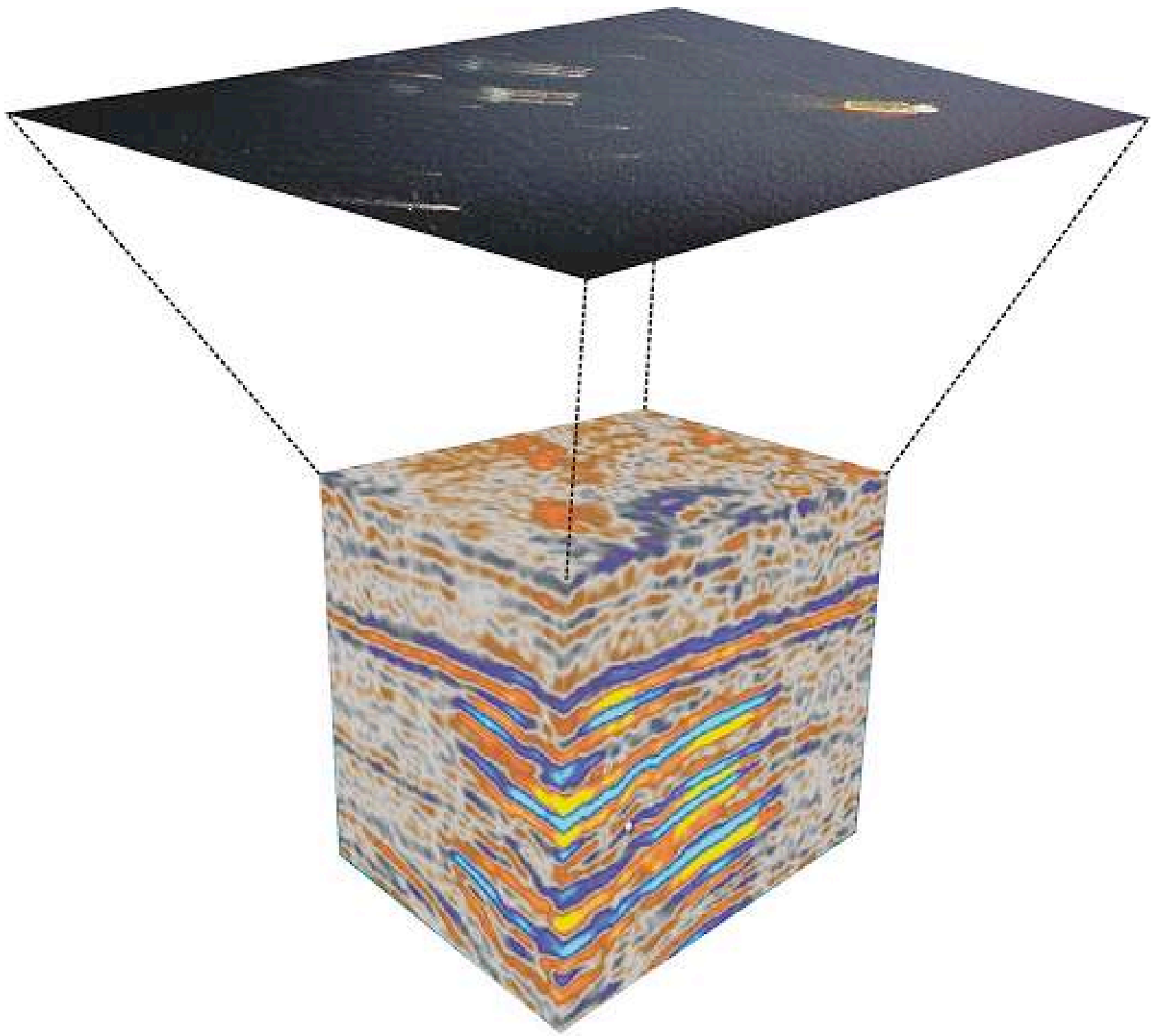


Sound is sent out,
The hydrophone array receives the reflections of different layers in the sea.
As a result, you have overlapping datasets which need to be postprocessed in order to get a clear picture of the sea ground.

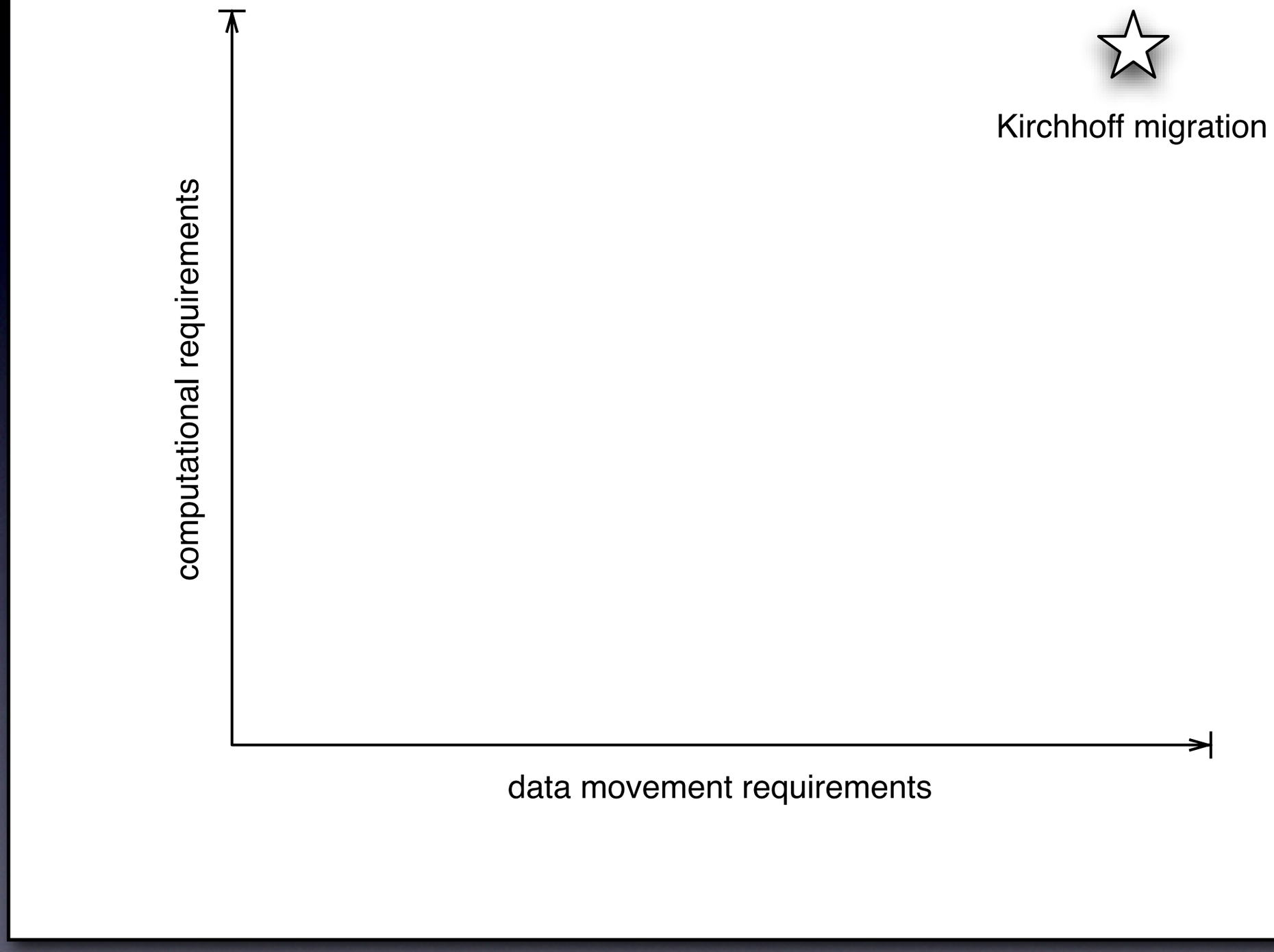
Kirchhoff-Migration



You need to correct the differences in signal runtime in order to get a clear picture. This method is called Kirchhoff-Migration
(I'll give no details on this since I am not an expert...)



Goal: The Geologists want to be able to dive into the cube, move it around, filter values etc.
7-dimensional dataset with some 100s GByte
-> This is currently done with our visualization system PV4D

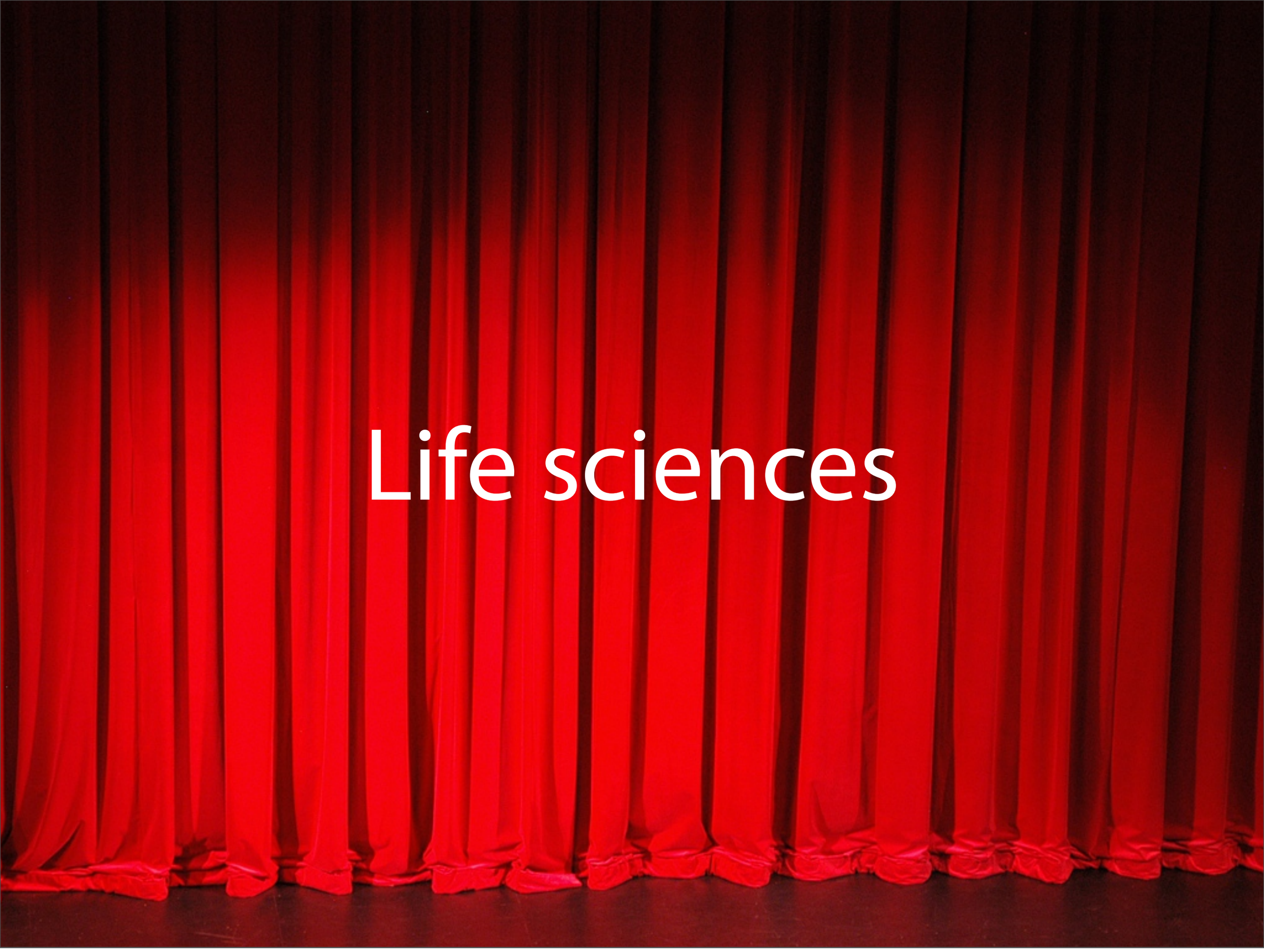


The challenge with this application lies within its huge demands:
- Runtimes can easily be 1 hour on a 10-CPU cluster for a 1 GB dataset with our implementation
- real runs are in the range of days and weeks for a 1000 CPU cluster!

Implementation

How is the Kirchhoff-Migration implemented?
We use our own code which is an MPI-Program we sell for inhouse use.

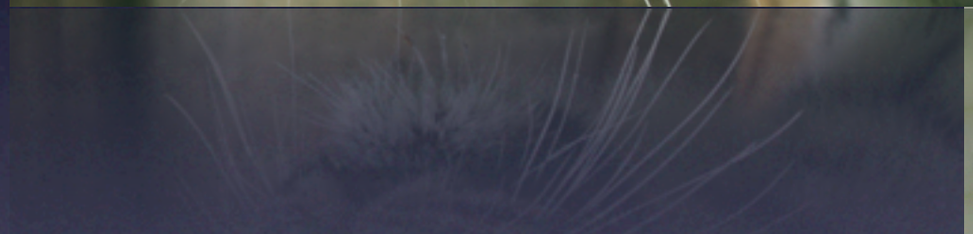
- Drawback MPI: If one process dies, the whole execution breaks
-> huge problem in big environments.
- we are working on a PHASTGrid implementation which will not use MPI to overcome this.
- our implementation is highly efficient.

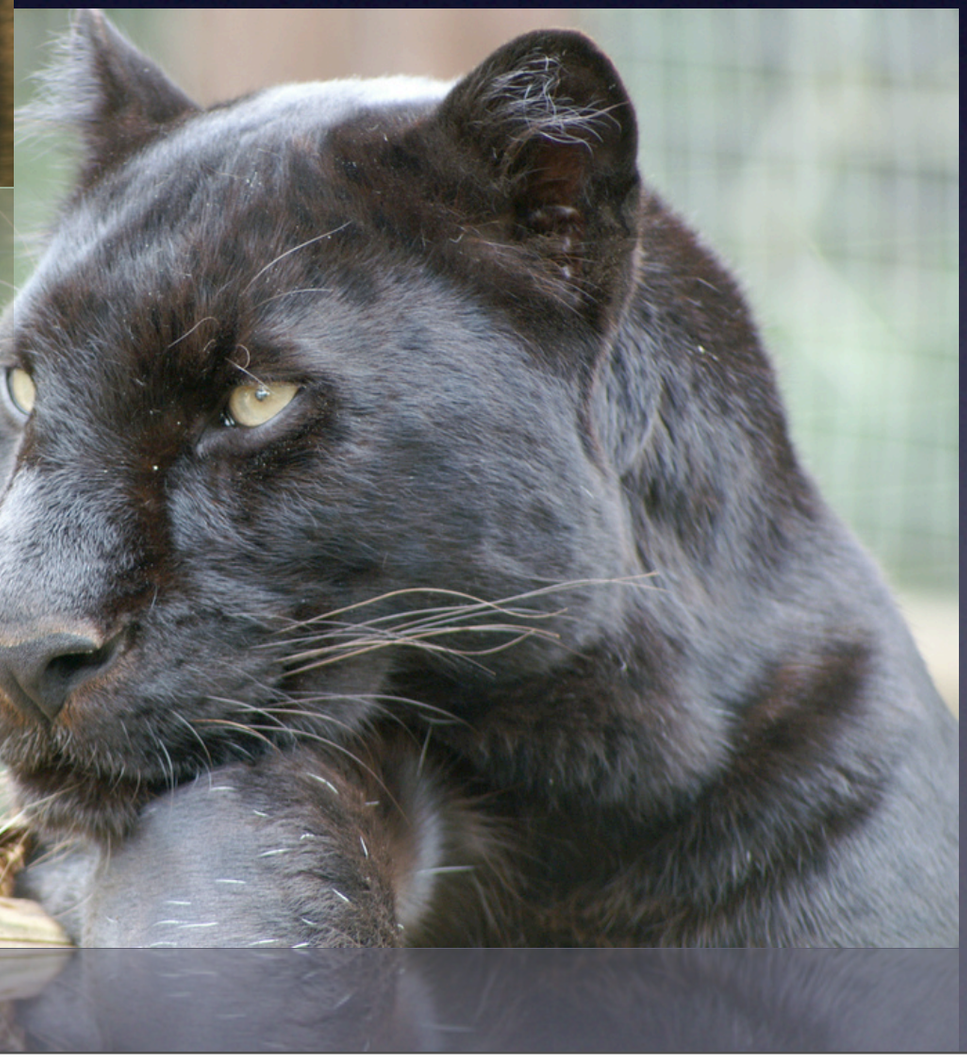


Short: PAUP

Phylogenetic Analysis Using Parsimony (PAUP)

Collaboration with the biology department of our local university.
- they want to analyze the genetic relation of organisms using PAUP
- they want to use some spare cycles which we donate ;-)

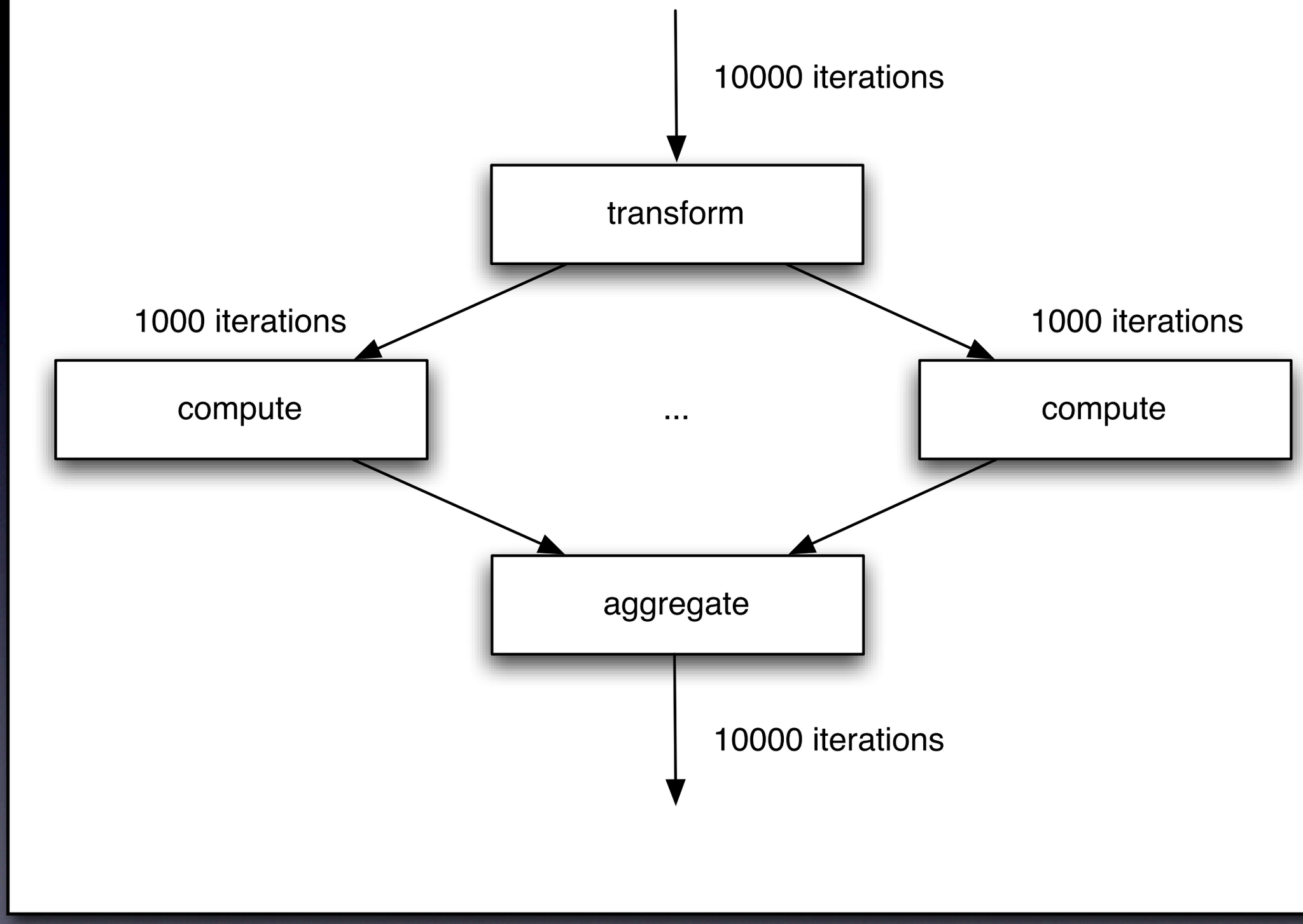




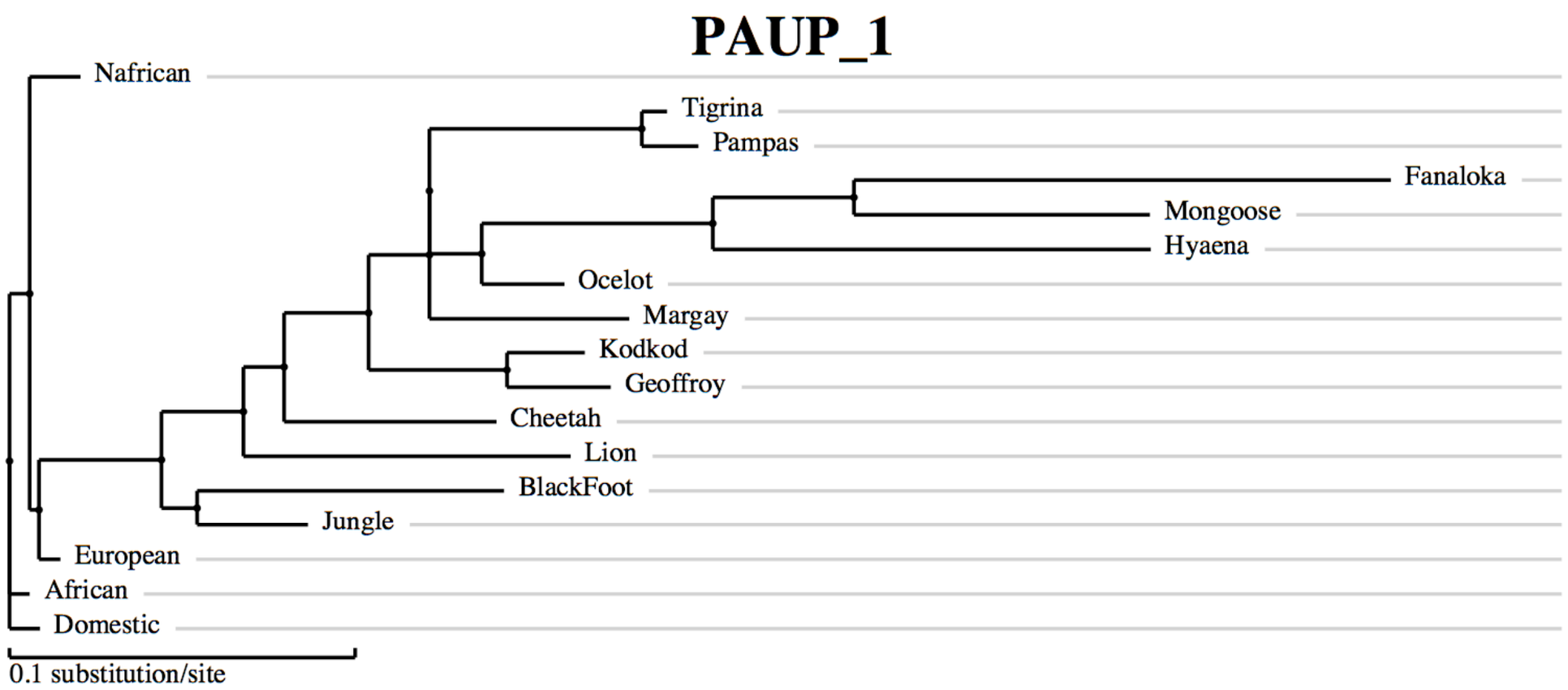
PAUP takes a set of genes and computes the distance between the organisms
– for example, for big cats.

```
BEGIN DATA;
DIMENSIONS NTAX=17 NCHAR= 363;
FORMAT INTERLEAVE MISSING=?  GAP=- MATCHCHAR=.  DATATYPE=DNA;
MATRIX
[ 111 111 111 122 222 222 223 333 333 333 444 444 444 455 123 456 789
012 345 678 901 234 567 890 123 456 789 012 345 678 901 ]
Domestic  ATG ACC AAC ATT CGA AAA TCA CAC ACC CTT ATC AAA ATT ATT AAT CAC TCA
African   ... ... ... ... ... ... ... ... C.. ... ... ... ... ... ... ... ...
European  ... ... ... ... ... ... ... ... C.. ... ... ... ... ... ..C ... ...
Nafrican  ... ... ... ... ... ... ... .C. C.T ... ... ... ... ... ..C ... ...
Jungle    ... ... ... ... ... ... ... ... C.T ..C ... ... ... ... ..C ... ...
BlackFoot ... ... ... ... ... ... ... .C. C.. ... ..T ... ..C ... ..C ... ...
Lion      ... ... ... ... ... ... ... ... C.. ... ... ... ..C ..C ... ... ...
Cheetah   ... ... ... ... ... ... ... ... C.. ... ... ... ..C G.. ... ... ...
Geoffroy  ... ... ... ... ... ... ... ... ... ... ... ..T ... ... ..C ..C ... ...
Kodkod    ... ... ... ... ... ... ... ... ... ... ... ..T ... ... ..C ... ... ...
Margay    ... ... ... ... ... ... ... ... C.. ... G.T ... ... ... ... ... ... ...
Ocelot    ... ... ... ... ... ... ... ... C.. ... ..T ... ... ... ... ... ... ...
Pampas    ... ... ... ... ... ... ... ... CT. ... ..T ... ... ... ..C ... ... ...
Tigrina   ... ... ... ... ... ... ... ... C.T ... ..T ... ... ..C ..C ... ... ...
Hyaena    ... ... ... ... ... ... ..T ... C.A ..C ..T ... ... ..C ..C A.A ... ...
Mongoose  ... ... ... ..C T.. ..G ..T ... C.G ..C ... ... ..C ..C ... A.G ..G
Fanaloka  ... ... ... ..C ... ... ... ... C.A ... ... C.. ... ..C ..C G.A ... ...
[ 111 555 555 556 666 666 666 777 777 777 788 888 888 889 999 999 999
000 234 567 890 123 456 789 012 345 678 901 234 567 890 123 456 789
012 ]
```

- Input: A rather small file, specifying some genes of the different organisms to be compared.



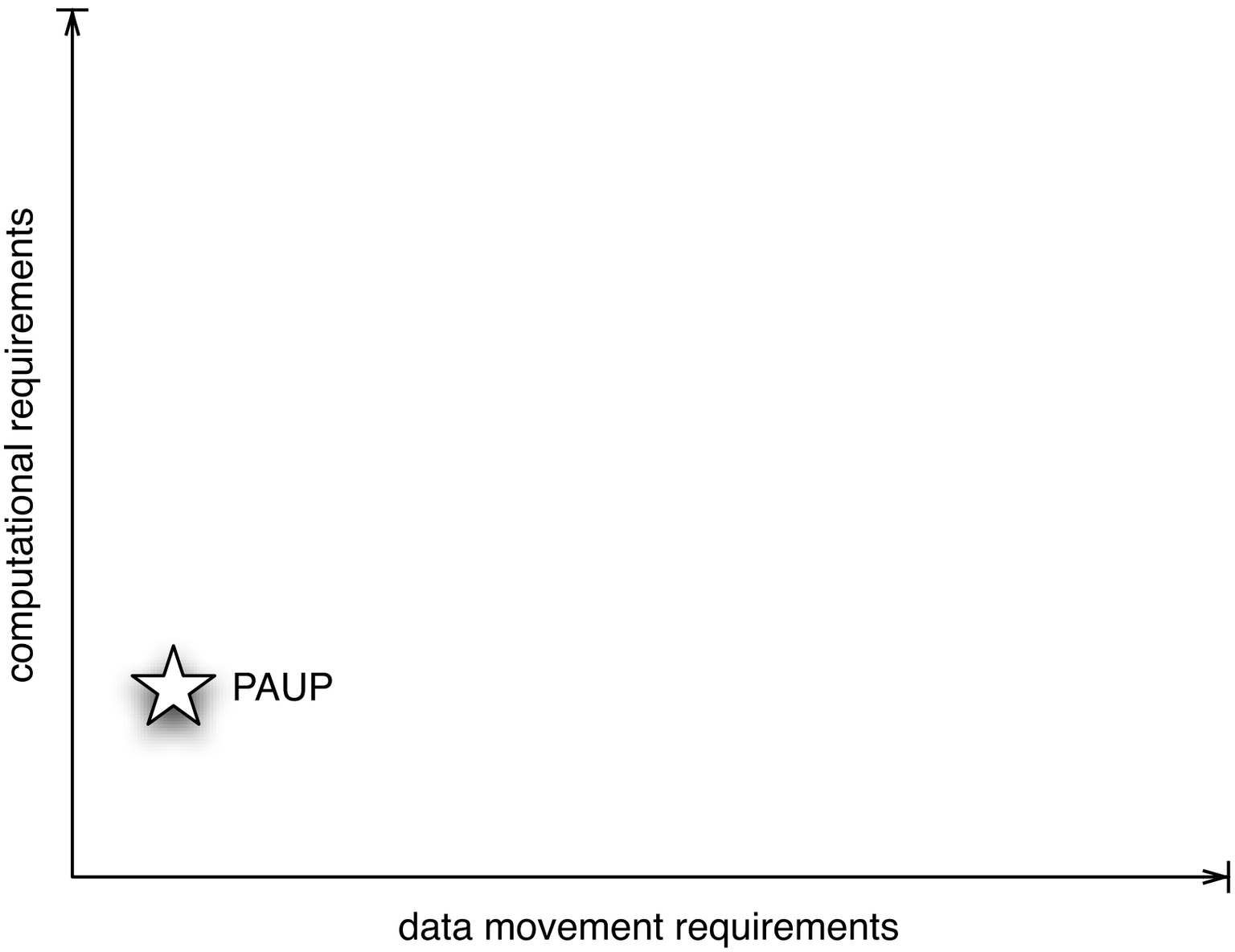
phastgrid parallelization - transform, compute, aggregate
- Input and output data is small, so we transfer it directly (no storage involved)
- PAUP is a Monte-Carlo like method, so that we can parallelize simply by splitting up the iterations and aggregate the results



result is the genetic relation of big cats - cheetah and tiger are not closely related (but beware, I'm not a biologist ;-))

UNICORE Frontend

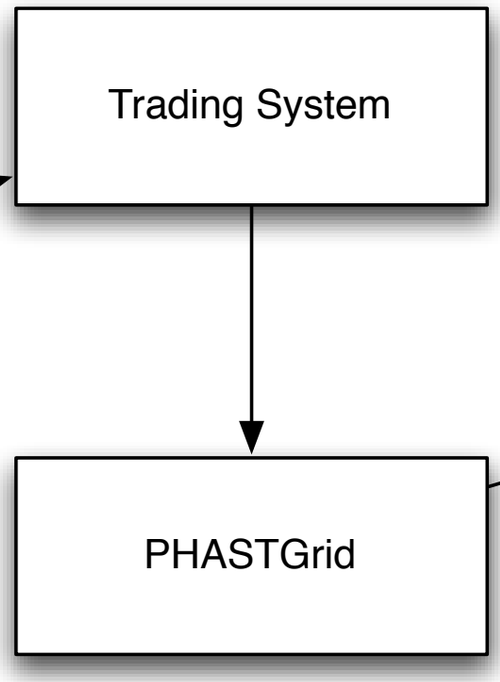
- The user uses UNICORE to access the system
- Data Staging, AAA etc. is handled by UNICORE
- We expose PHASTGrid like a batch system


computational requirements
PAUP
data movement requirements

can’t tell you more

High Troughput System - approx. 50 Jobs/s

Trading Room

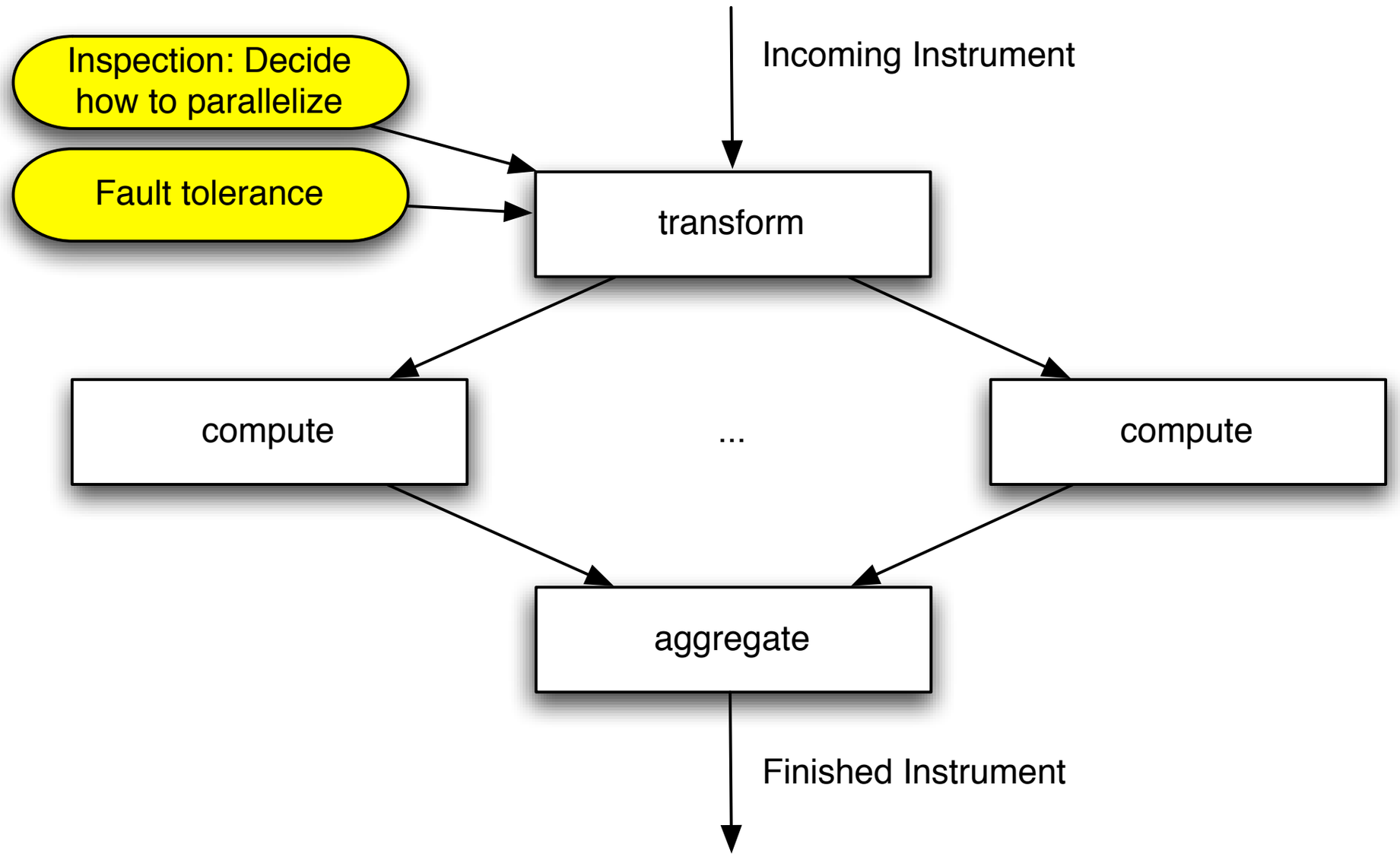


24/7 on more than 3000 CPUs

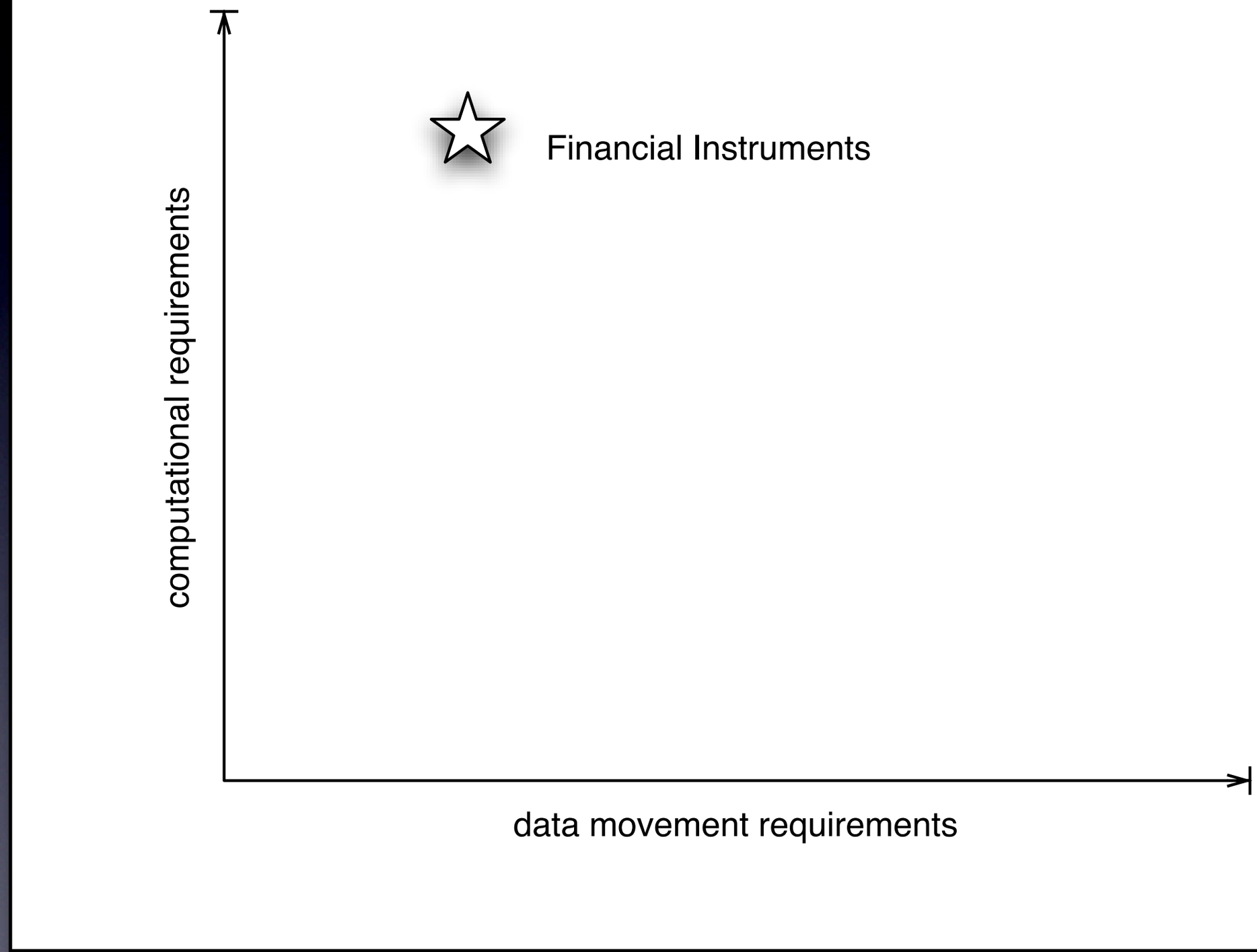
Portfolio analysis for investors in the trading room
- Trader submit a query
- Trading system submits a bunch of jobs to PHASTGrid
- The jobs are distributed among more than 3000 CPUs
- 50 Jobs/s sustained, 24/7 operation since two years.
- We even survived reboot storms etc.
- There was a memory leak (in the application) which caused nodes to crash (they simply rebooted and joined the system again)

Portfolio analysis

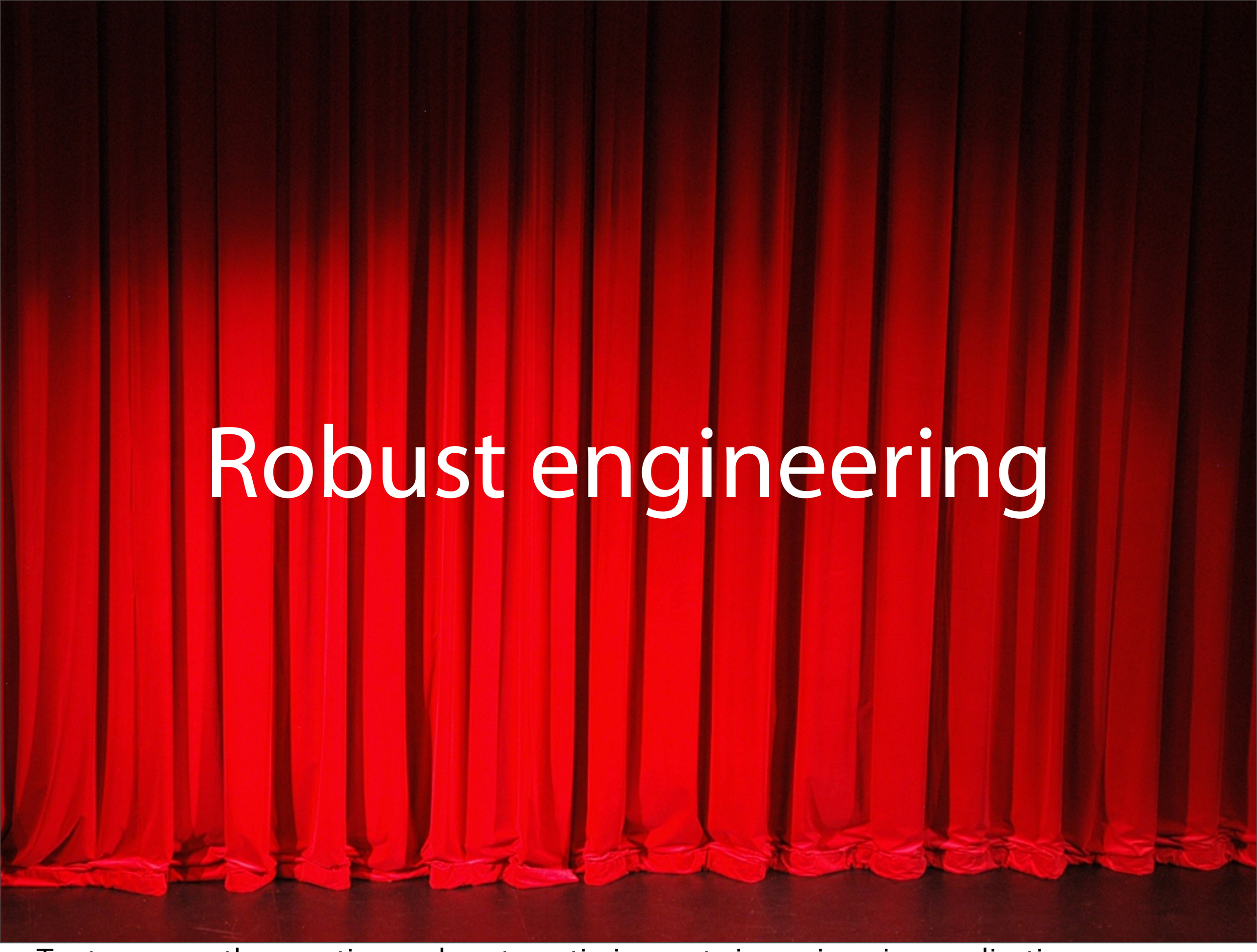
- The application is custom-built by our customer
- For the parallelization, we can investigate each job
...



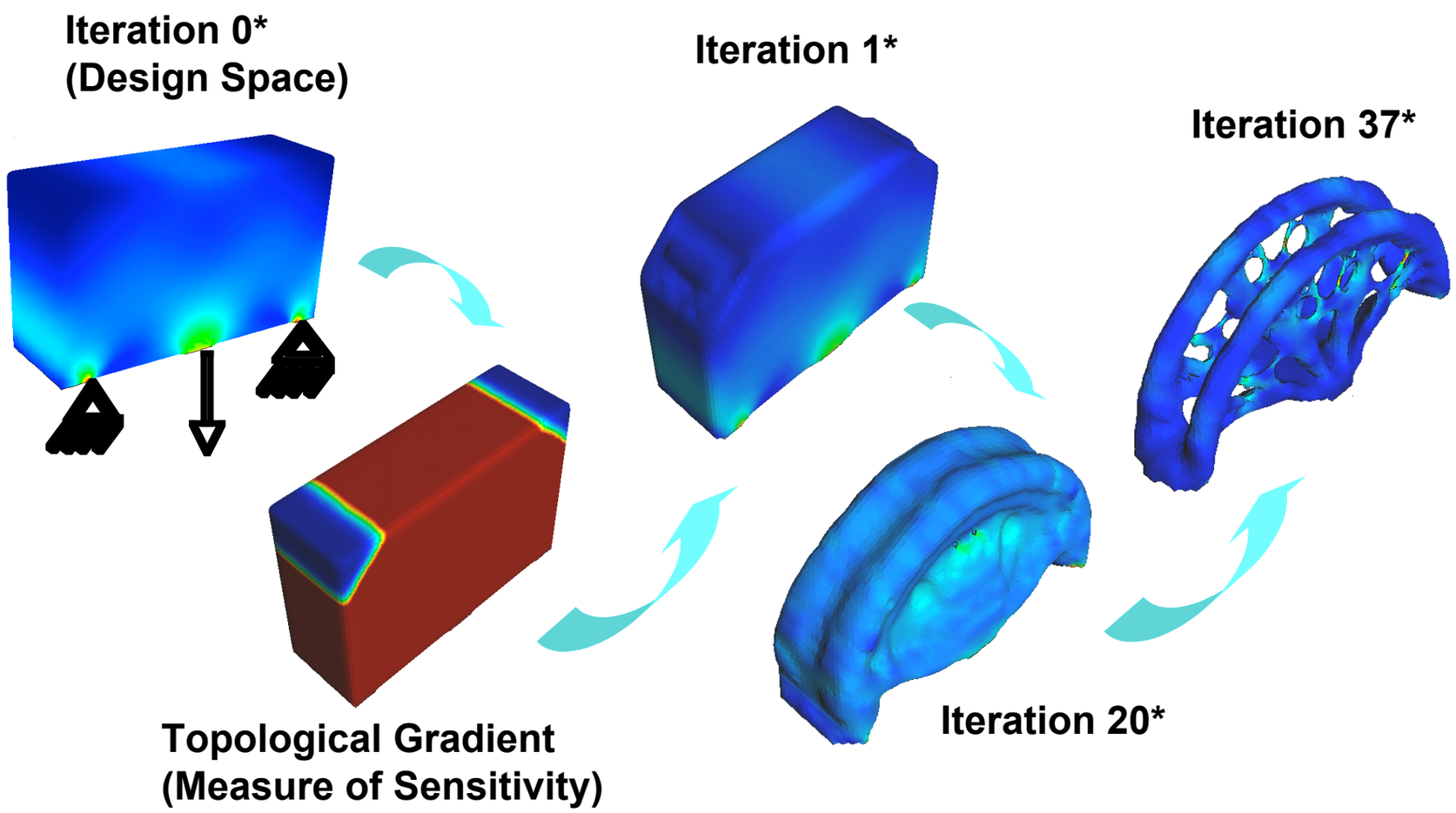
We use the PHASTGrid framework to parallelize the jobs:
- in the transform step, we can rely on the applications performance metric to decide how many compute nodes should work on an instrument
- So, sometimes, we need 100 CPUs, sometimes only 10. This allows us to keep answer times almost constant.
- In the Grid Server, we can also take care of the fault tolerance: a job will always be delivered.



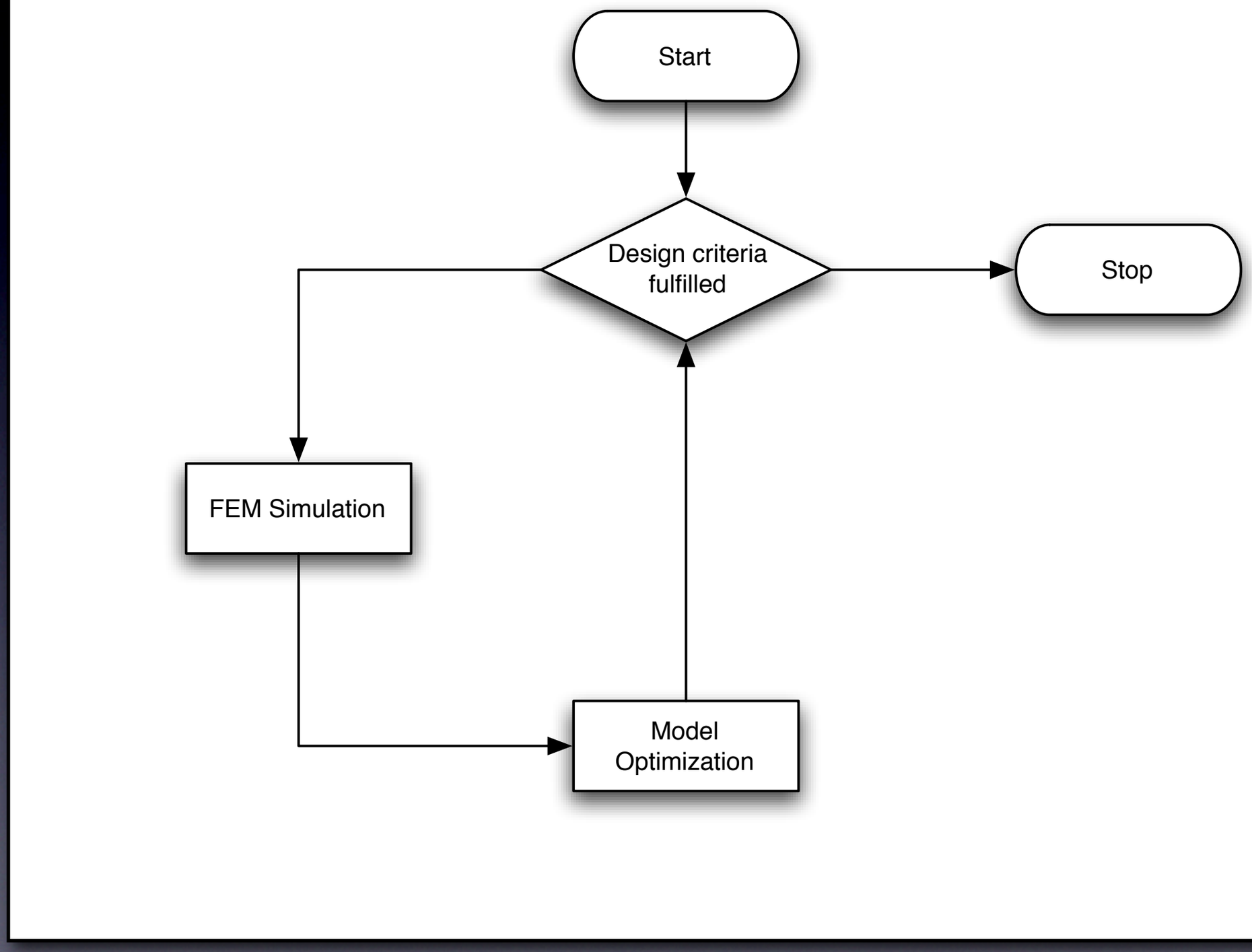
The requirements matrix looks almost similar to the PAUP application
- however, the demands on scalability and failure tolerance are much higher than with PAUP
- The customer uses its own infrastructure as a frontend to PHASTGrid (not UNICORE etc)
- our customer is very satisfied with the current deployment

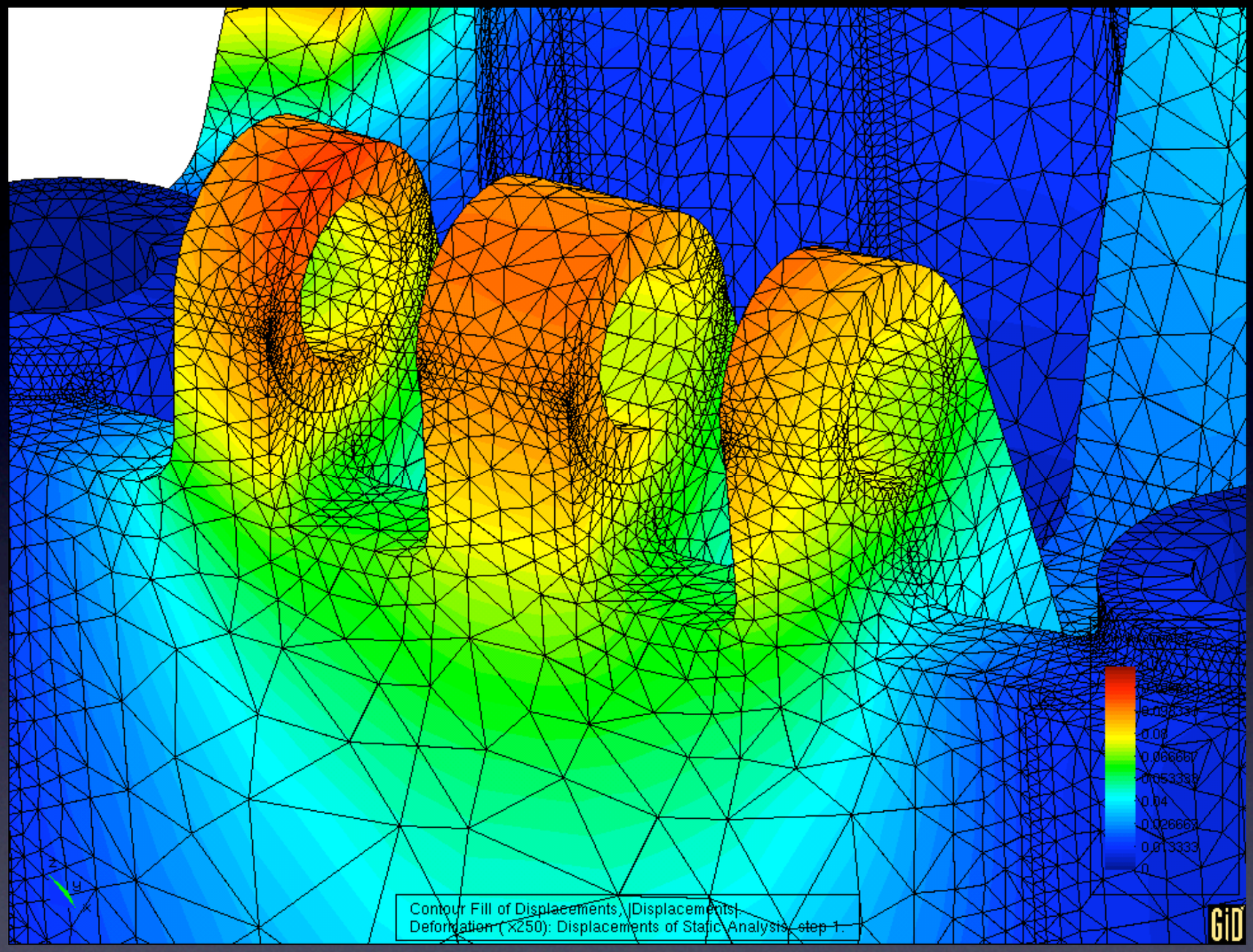


- Try to answer the question on how to optimize parts in engineering application
- e.g. how to design a part of a car in a way that its lifetime is good enough and the material cost are minimized
- while being tolerant to manufacturing mistakes etc.



Optimization within the design boundaries – you need to be sure that the element will be stable even under changing manufacturing situations.


Start
Design criteria fulfilled
Stop
FEM Simulation
Model Optimization

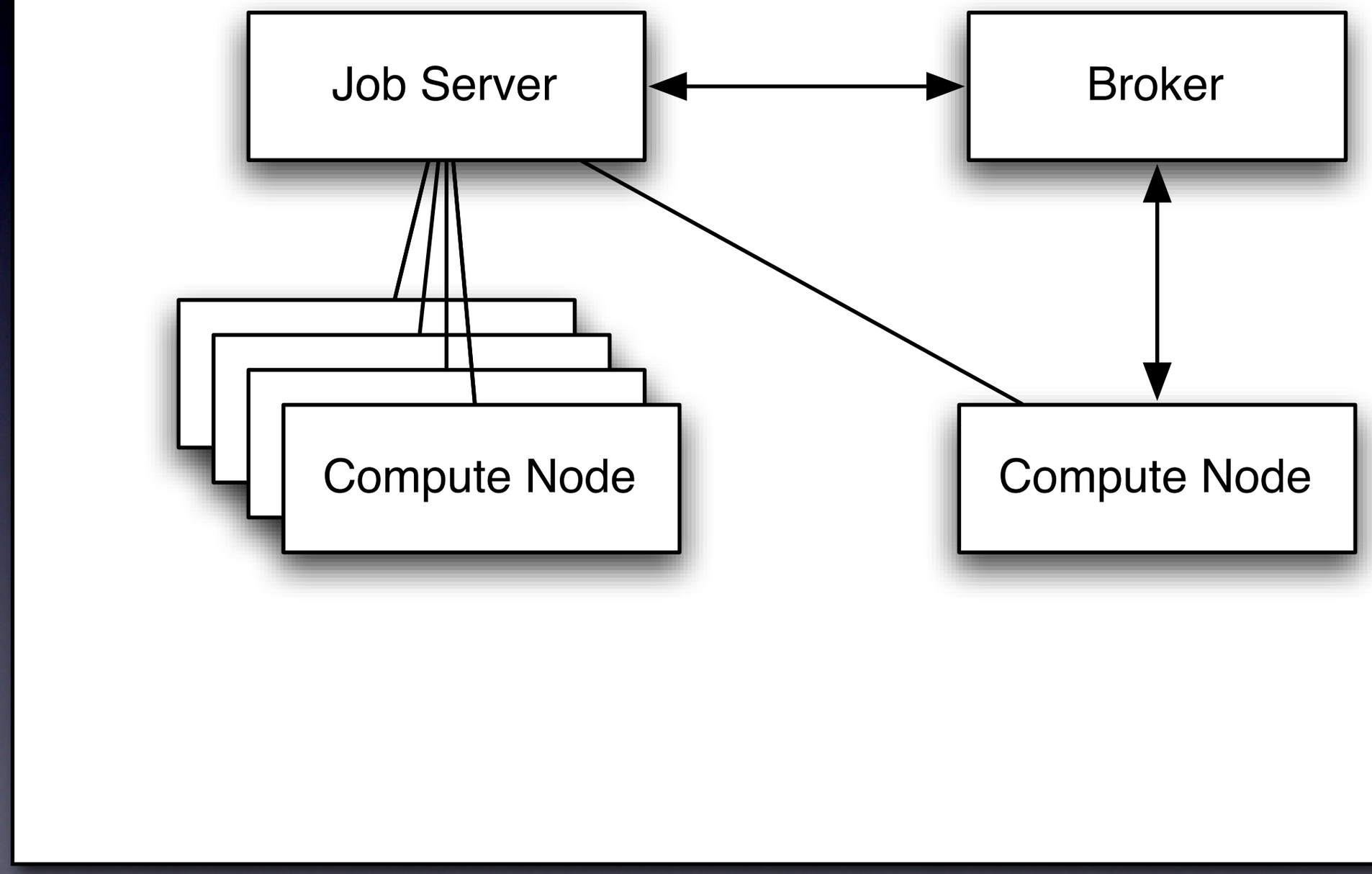


DDFEM

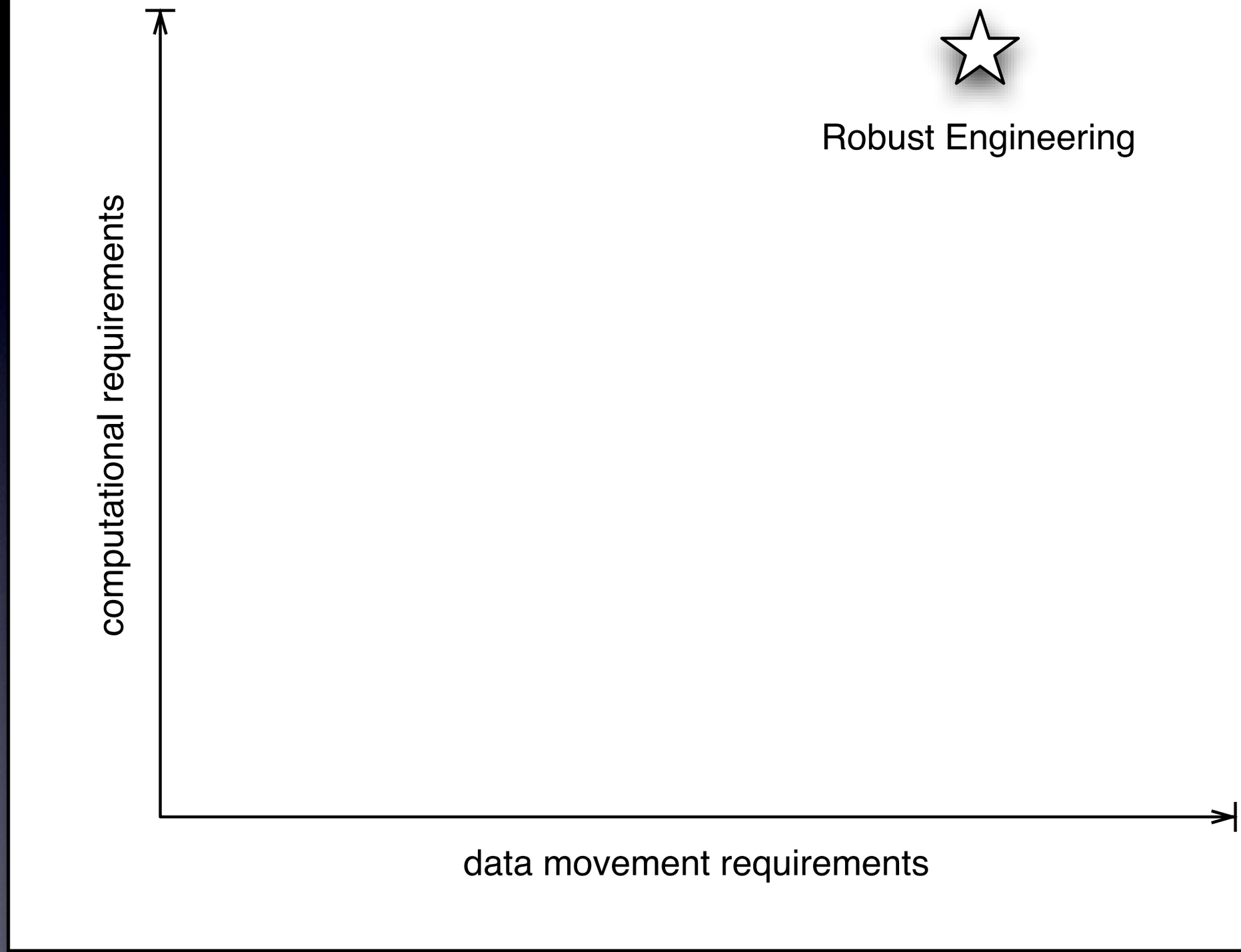
- Domain Decomposition FEM (structural analysis and optimization)
- A code developed in our Department, so that we can easily integrate this into PHASTGrid

Calana Broker

Integration in PHASTGrid:
(1) DDFEM is an MPI code – so several worker nodes need to be assigned to the job simultaneously
(2) Integration of Calana scheduler to achieve this
– Calana uses auctions to decide where to run a job
– Coallocation is also implemented
– A special case for Calana, this is a much more sophisticated system



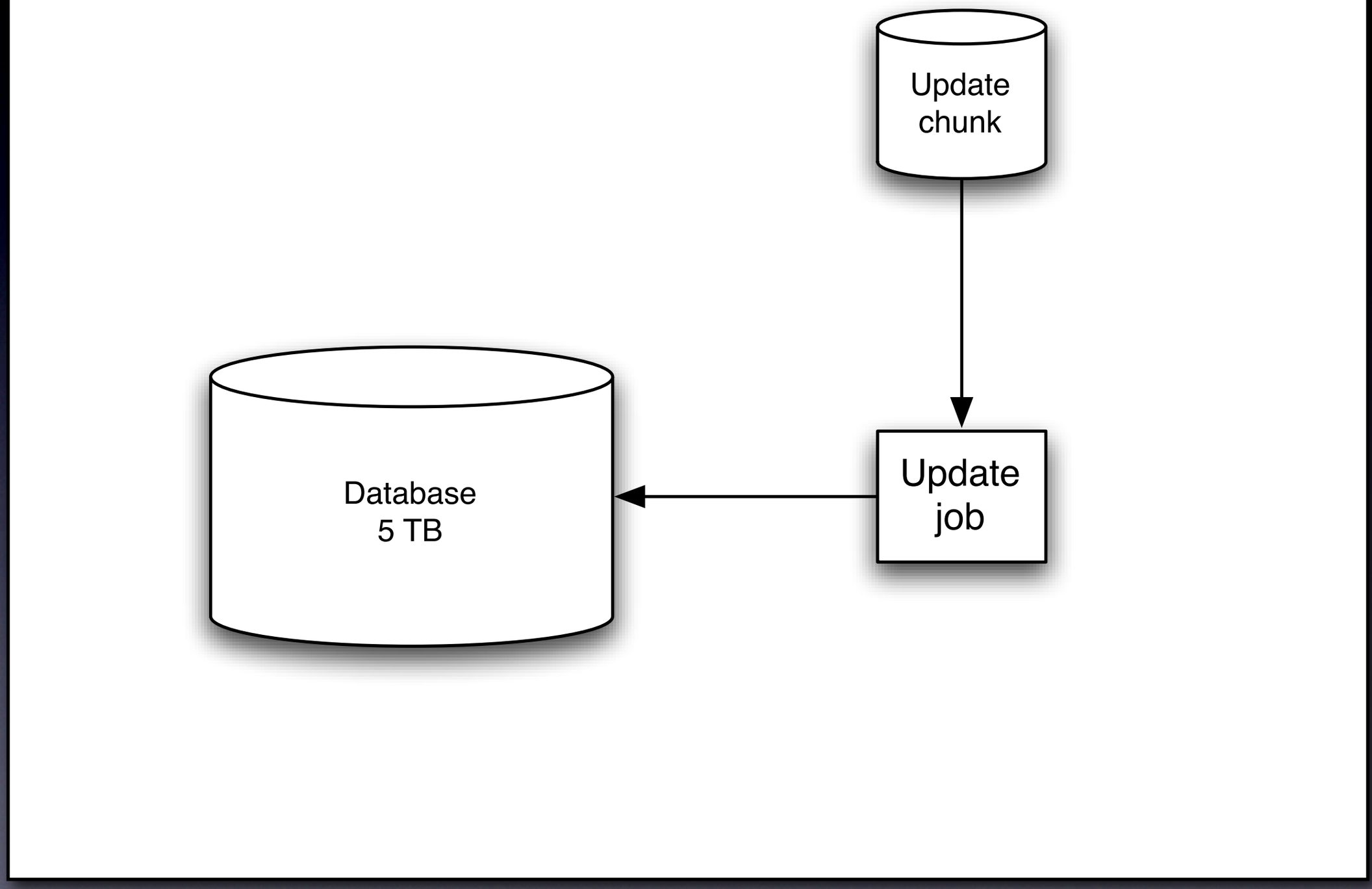
- No information system needed - either a job can be executed on the compute node or not


Robust Engineering
computational requirements
data movement requirements

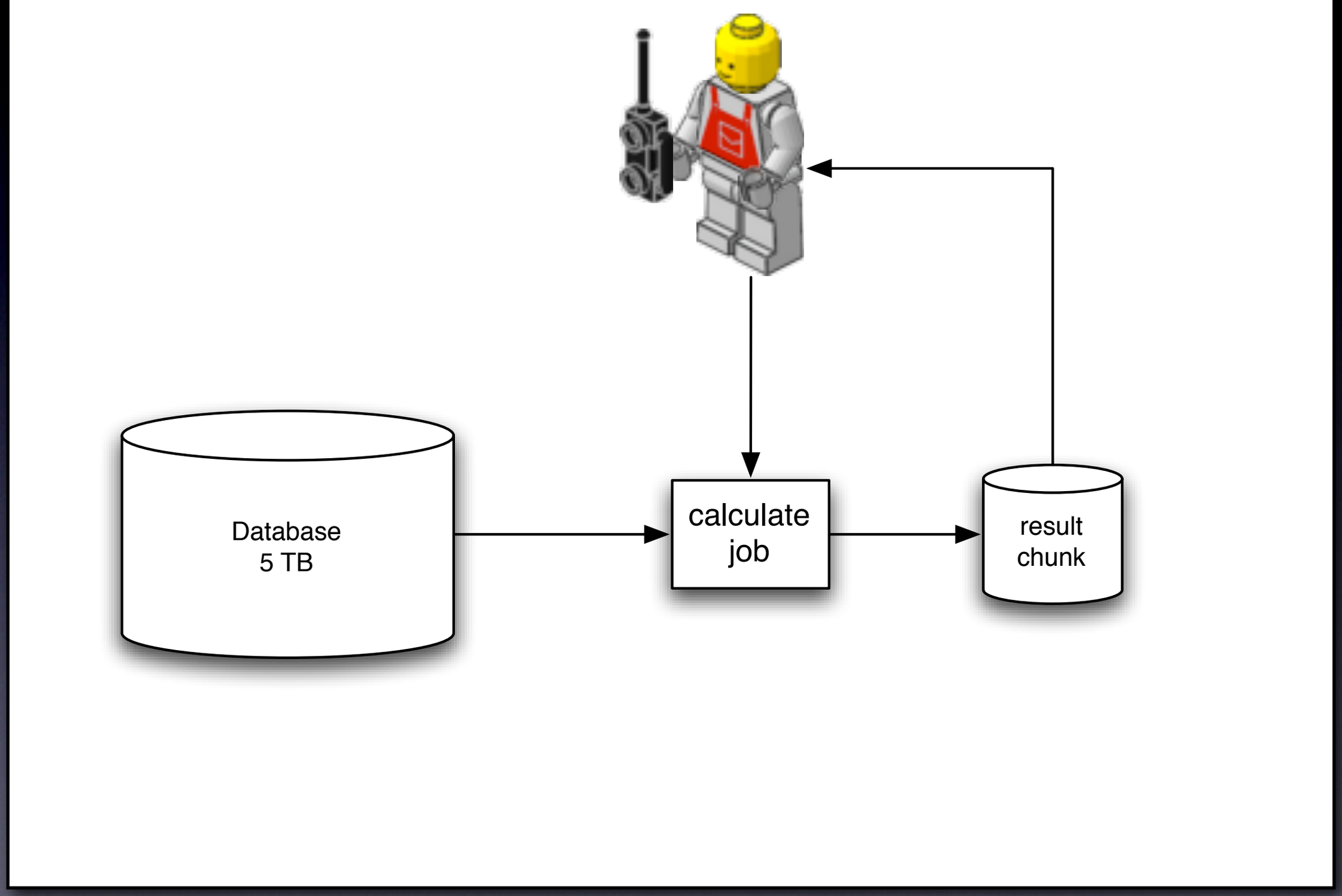
Can’t tell you what this is about, but the application is quite different – I would like to integrate it...

Customer wants to use our resources

- He provides a data analysis software which will be executed daily
- There are two kinds of jobs: ...



(1) There are database update jobs. The update chunk will be added to the database.
- It is rather big (5TB)
- Jobs will run on this database



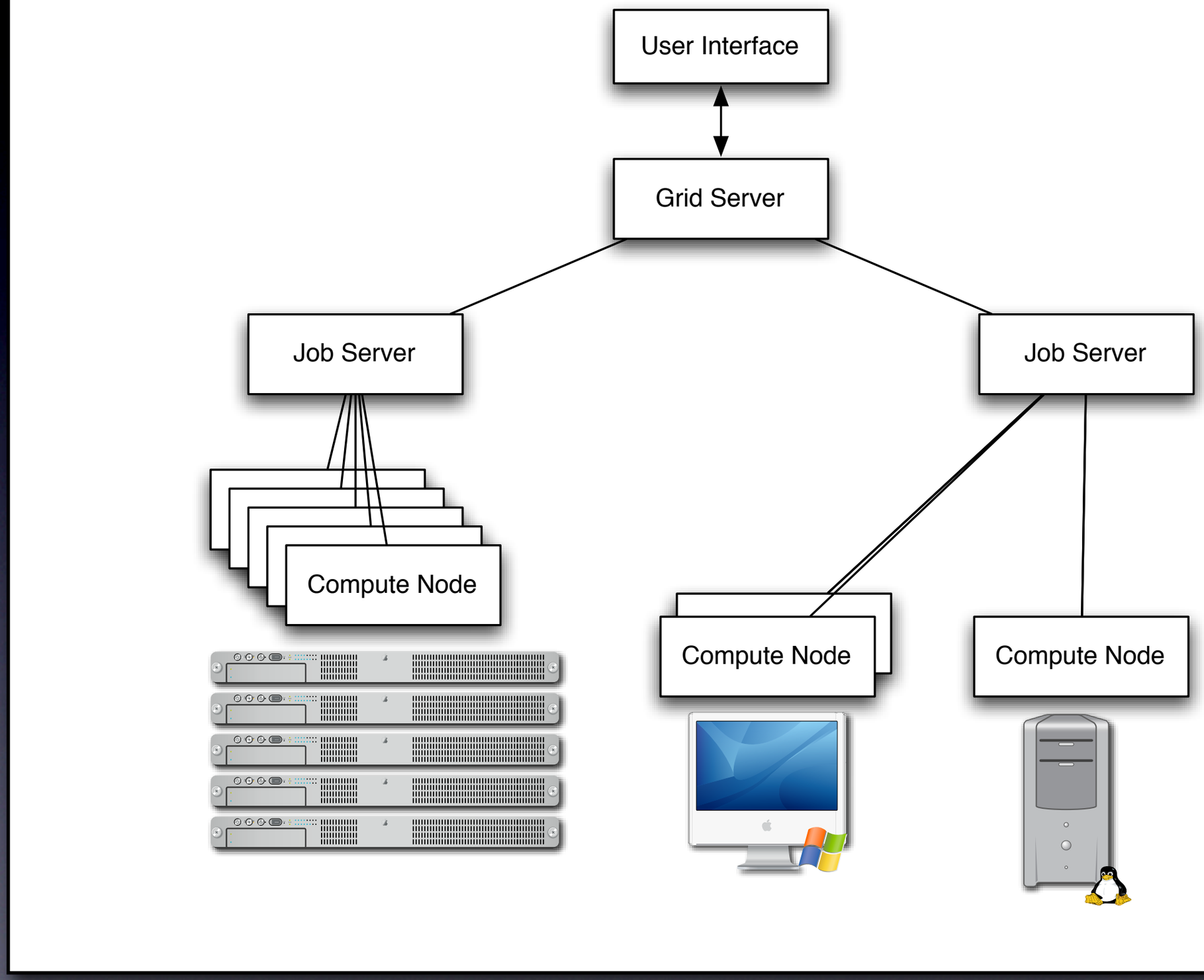
(2) The compute jobs are triggered by the user
- runs approx. in 10 minutes
- it needs approx. 10 MB of input data from the database
- the result is only a few KB

Database management

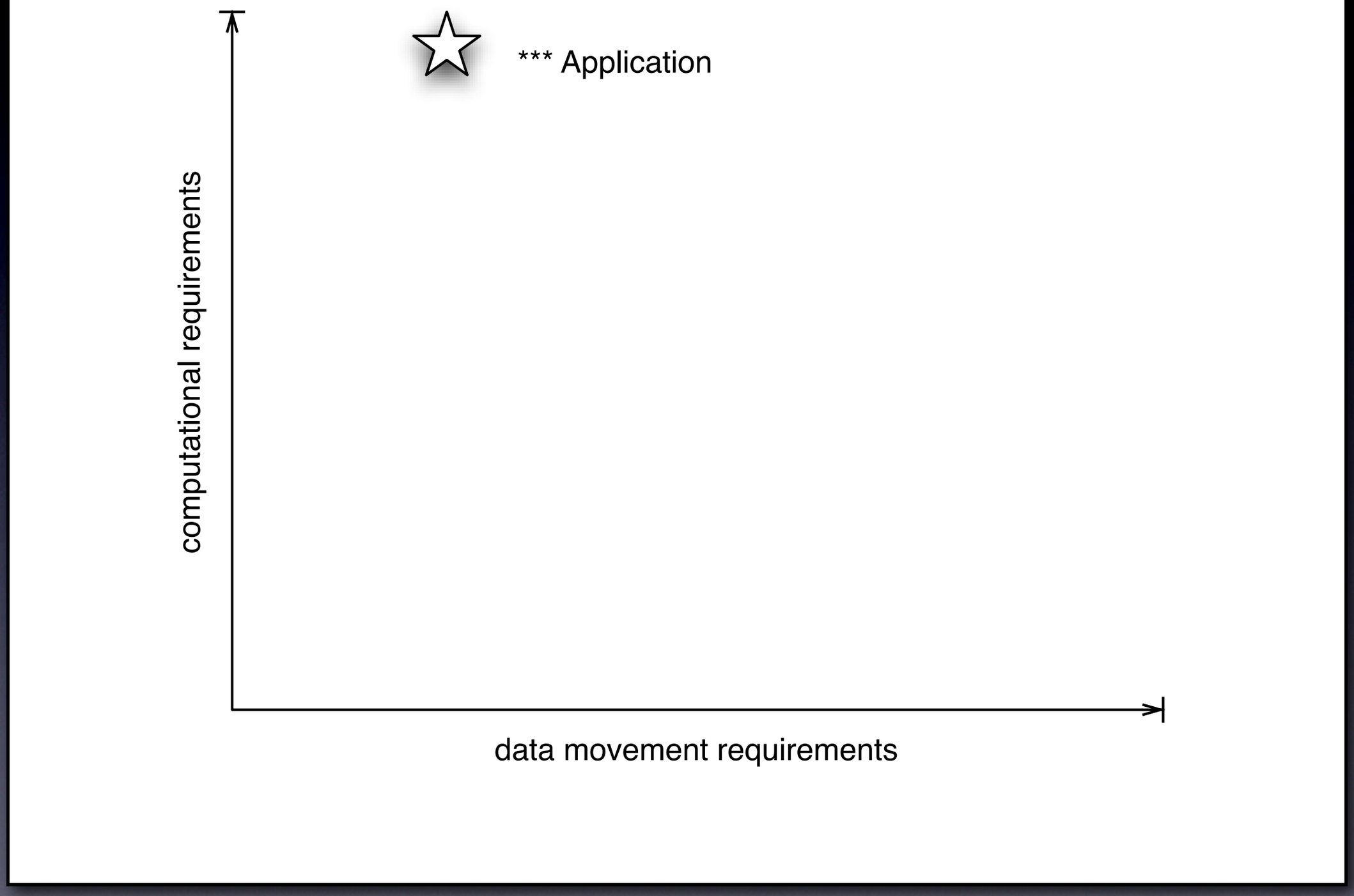
We need to maintain the database close to the execution locations (LAN)
- we will replicate the database and update all replicas simultaneously

Guaranteed execution

- Our customer forces us to present the results within 24 hours
- But within this timeframe, we have all freedom
-> Which allows us to use spare cycles.



- We use again PHASTGrid, but
(1) either we do use our cluster resources
(2) or we use resources of a dynamic job server pool - resources join the pool whenever they are capable to



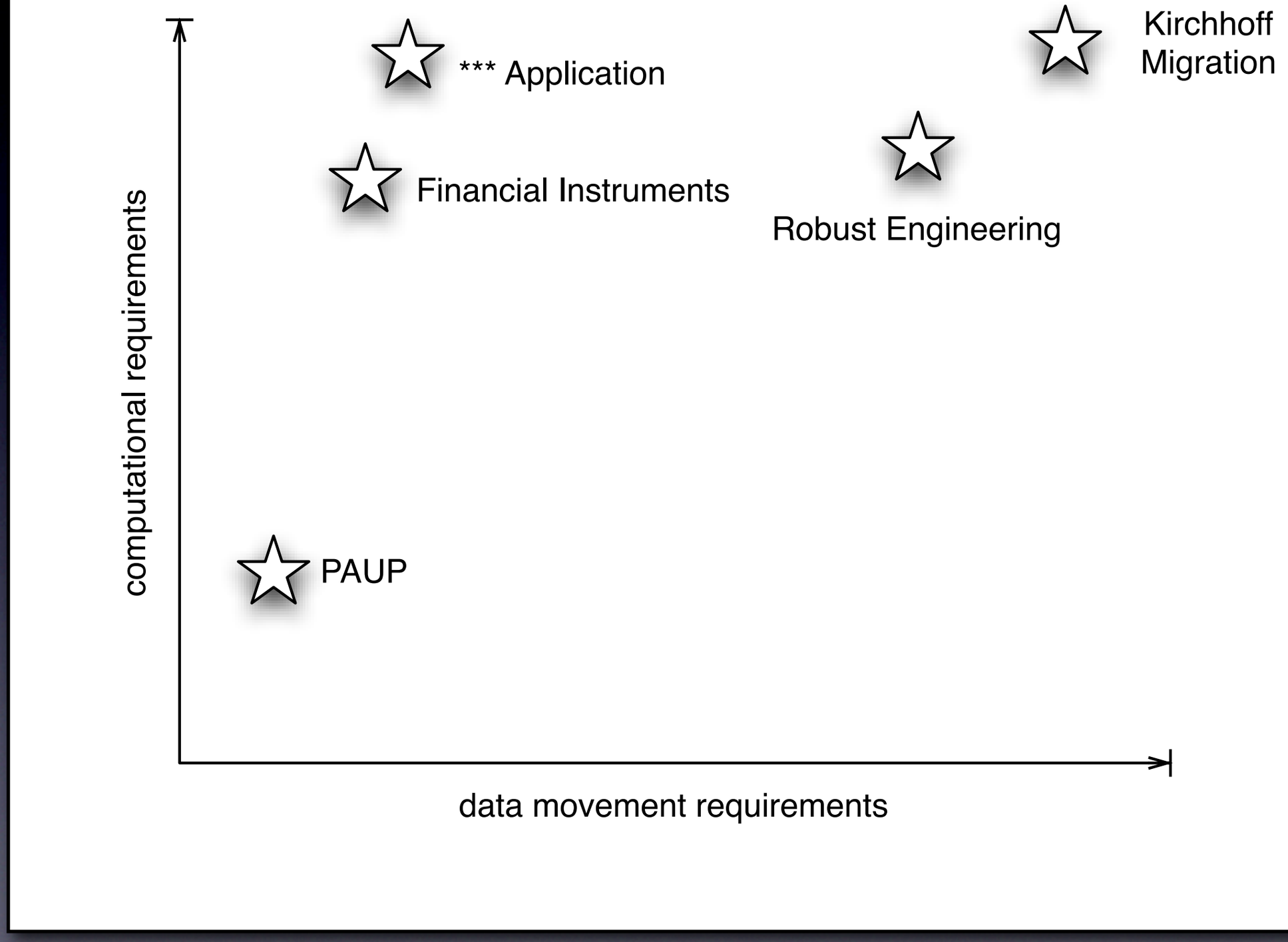
15 minutes on a single CPU, 10 MB data per job - but there can be a request to calculate 5000 jobs!

-> It is easy to parallelize

Observations

You need to understand the application.

Crucial: You need to understand the application
- Application requirements differ significantly
- Different applications can be mapped on different infrastructures



- Central question: relation of data movement vs. computing demands
-> These are just two dimensions - we need more to describe everything
- we are discussing other models to characterize the applications - but this would be out of the scope of this talk.

You need to understand the users requirements.

- How important is the reliability? What guarantees are needed?
- How do users work? What kind of interface do users need? -> Integration in the environment, like e.g. for the financial usecase

Fraunhofer

ITWM

Fraunhofer Institut
Techno- und
Wirtschaftsmathematik

come visit us, talk to me, discuss with me!