Optimisation of usage of mathematical libraries in the Grid environment

Cracow Grid Workshop, December 11-14, 2002

Maciej Brzeźniak, Norbert Meyer
Poznań Supercomputing and Networking Center

Maciej.Brzezniak@man.poznan.pl, meyer@man.poznan.pl

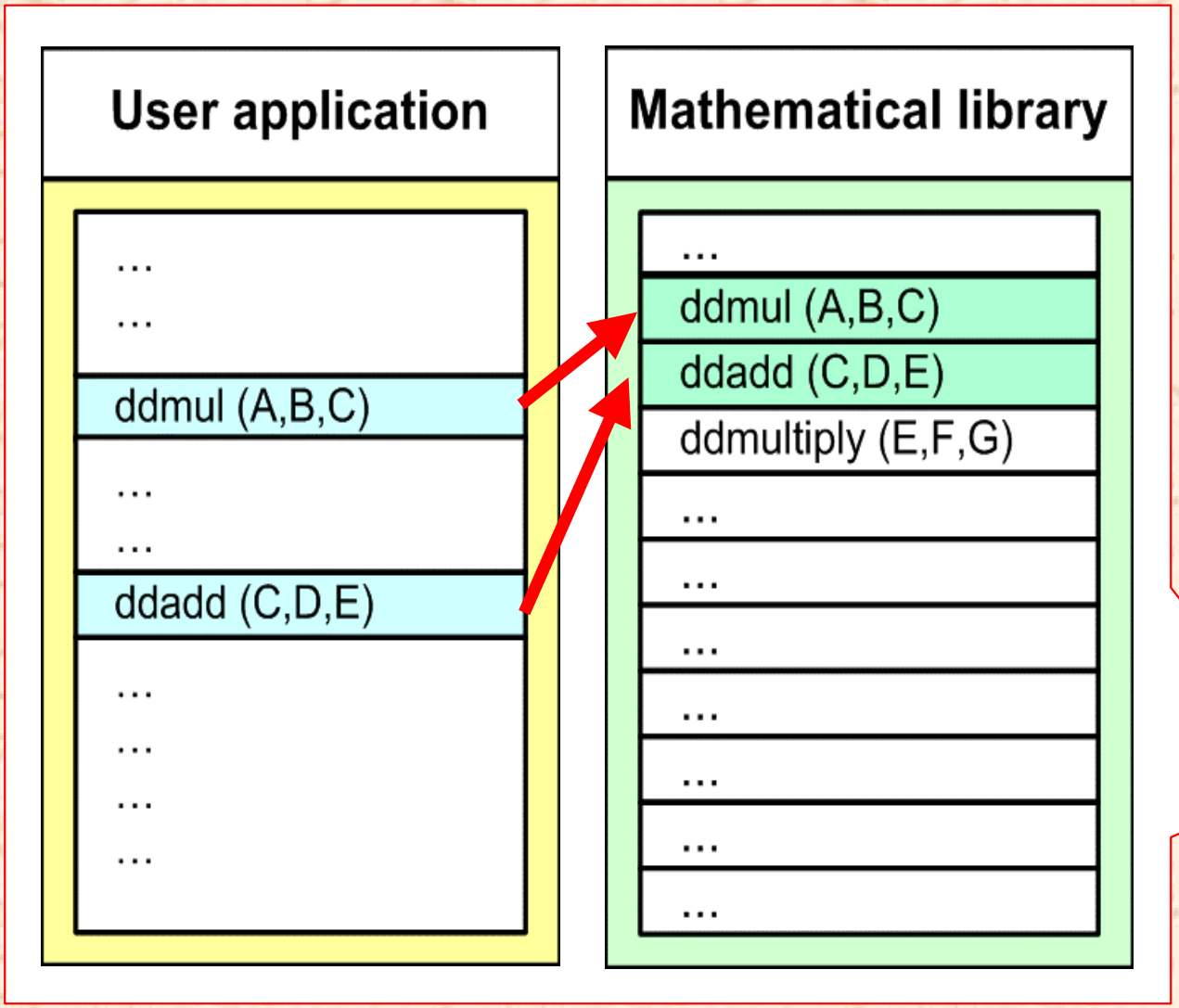
Presentation outline

- **what is the place** of mathematical libraries in the Grid environment?
- **why** is the usage of mathematical libraries to be optimized?
- what is **the state of the art** in this subject?
- what is the subject of **our work?**

Mathematical libraries in the Grid (1)

- Many scientific applications perform the complicated computations. **Computations are composed of the mathematical operations.**
- **The scientists want to perform the computations but they do not want to implement numerical algorithms.**
- The standard **mathematical operations** are implemented as the ready-to-use *C* or *Fortran* functions.
- The **libraries of the mathematical functions** are developed and delivered to the users (e.g. BLAS, LAPACK)

Mathematical libraries in the Grid (3)



User application

Mathematical library

Mathematical libraries in the Grid (4)

- There is a **need to use many computing resources** to perform the scientific computations.
- Users often needs more computing resources than the local ones i.e.:
  - user workstation
  - individual supercomputer
  - given queue system
- Computing applications must be run in the distributed Grid environment

Mathematical libraries in the Grid (5)

How to run a computing application in the Grid environment?

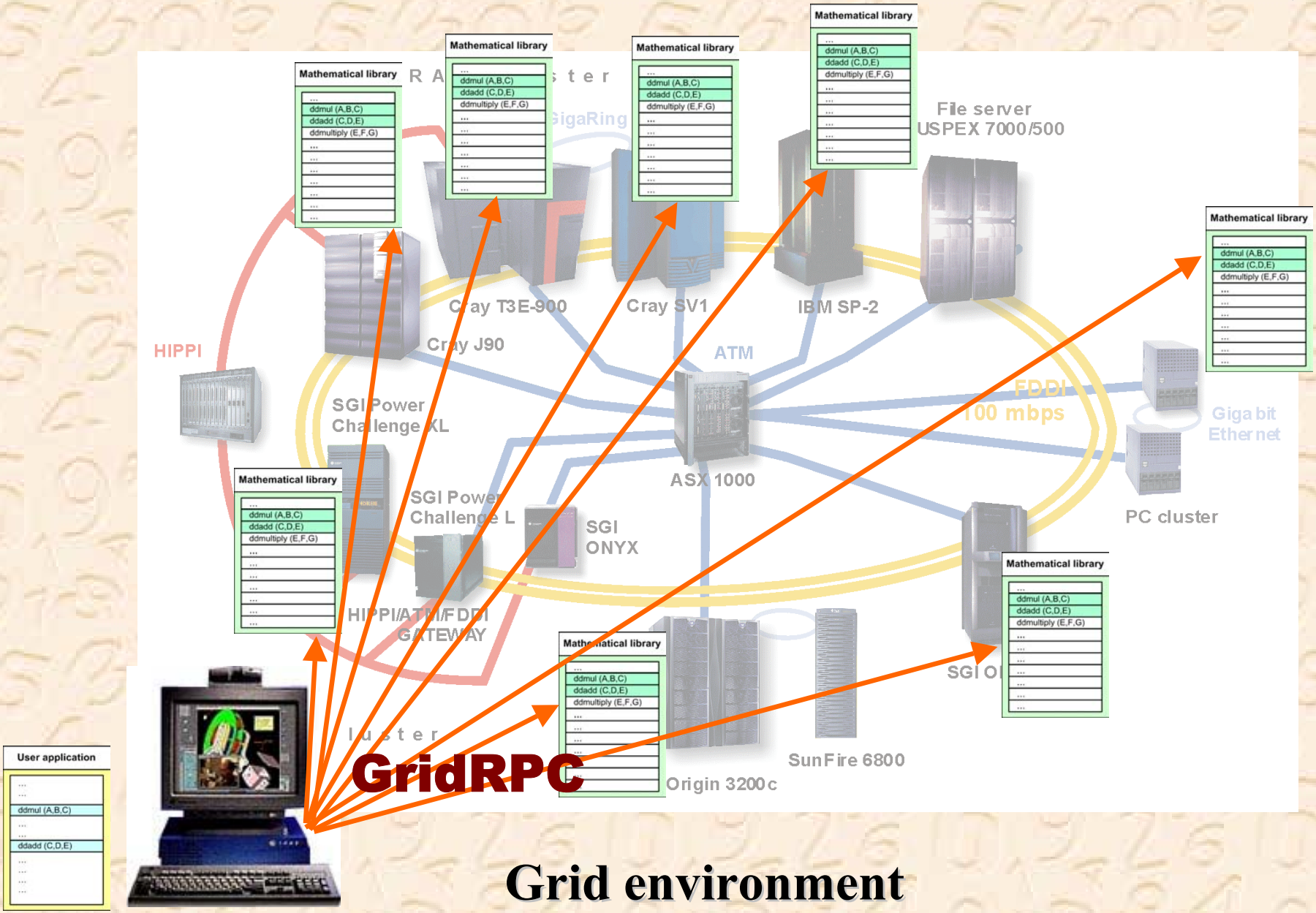
– put the user application **as a whole into the Grid system**

(e.g. run using Globus GRAM, putting into a queue system...)

– leave the user application on the user's machine but **do the calls to the remote mathematical library functions**

(e.g. RPC, Grid RPC, Ninf, NetSolve, CORBA, Java RMI)

Mathematical libraries in the Grid (8)



Grid environment

Grid RPC (1)

Grid RPC: *RPC mechanism tailored to the Grid*

- Invented by the authors of *NetSolve* and *Ninf* projects
- Status of the GridRPC technology:
  - Incorporation of the Grid RPC mechanisms into the Globus Toolkit is planned (see the GGF RGPM for details)
  - GridRPC is only the idea.
    It is implemented in Grid RPC *systems*: *Ninf*, *NetSolve*

Grid RPC's reference systems (1)

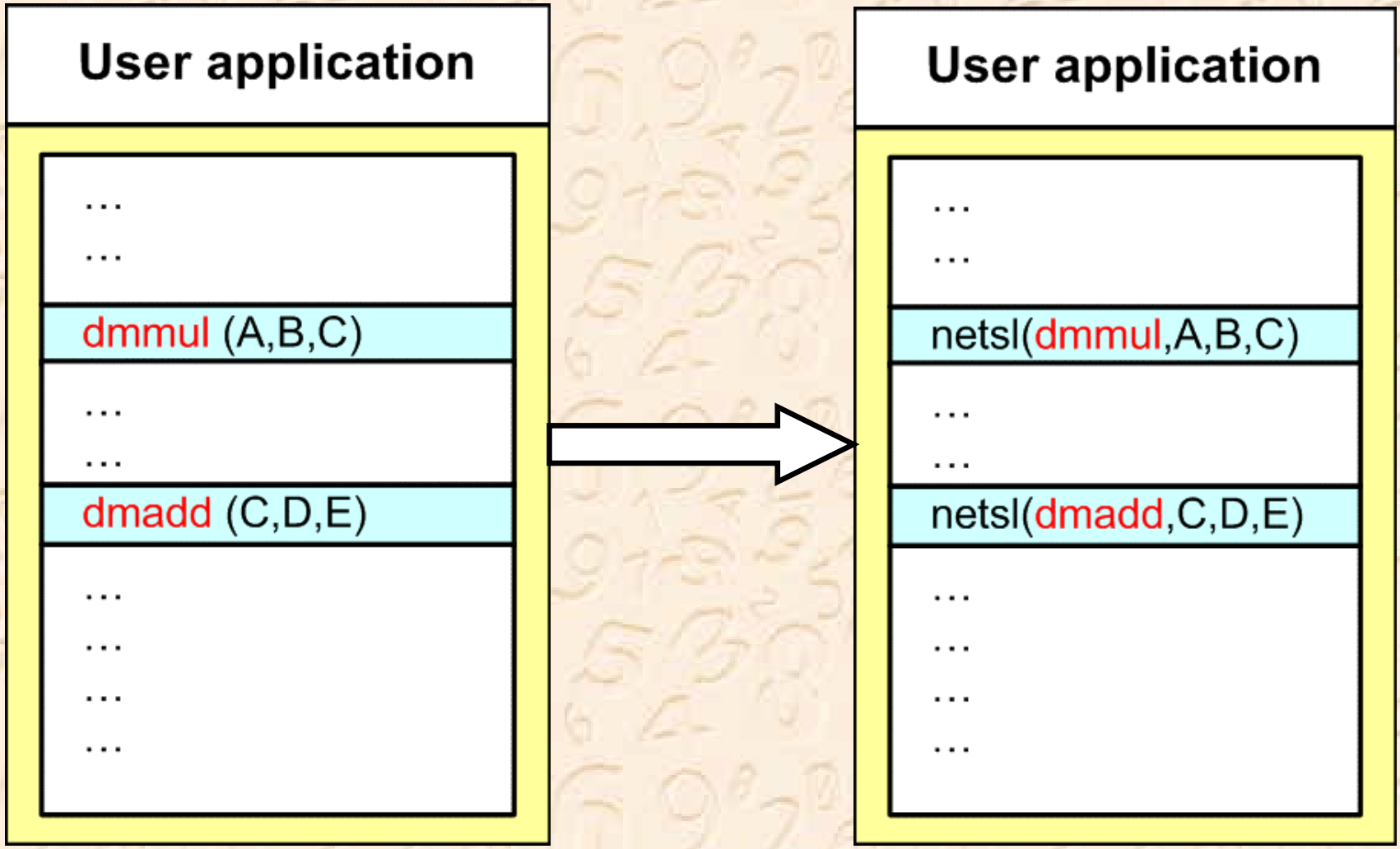
- **NetSolve:**
  - developed in the Innovating Computing Laboratory of the University of Tennessee, Knoxville, USA
  - web page at: icl.cs.utk.edu/netsolve
- **Ninf:**
  - developed by National Institute of Advanced Industrial Science and Technology, Grid Technology Research Center; Umezono, Tsukuba, Ibaraki, Japan
  - web page at: ninf.apgrid.org

Grid RPC's reference systems (2)

- **C API :**
  - **NetSolve:**
    - synchronous: `netsl (func_name, ...)`
    - asynchronous: `netsnbl (func_name, ...)`
      `netslpb (...),`
      `netslwt (...)`
  - **Ninf:**
    - synchronous: `Ninf_call (func_name,...)`
    - asynchronous: `Ninf_call_async (func_name,...)`
      `Ninf_wait (...)`
- other APIs:
  - NetSolve: C, Fortran, Matlab, Unix shell, Mathematica, Web Java GUI
  - Ninf: C, Fortran, Mathematica, Excel

Example local->remote call transition

(NetSolve example)

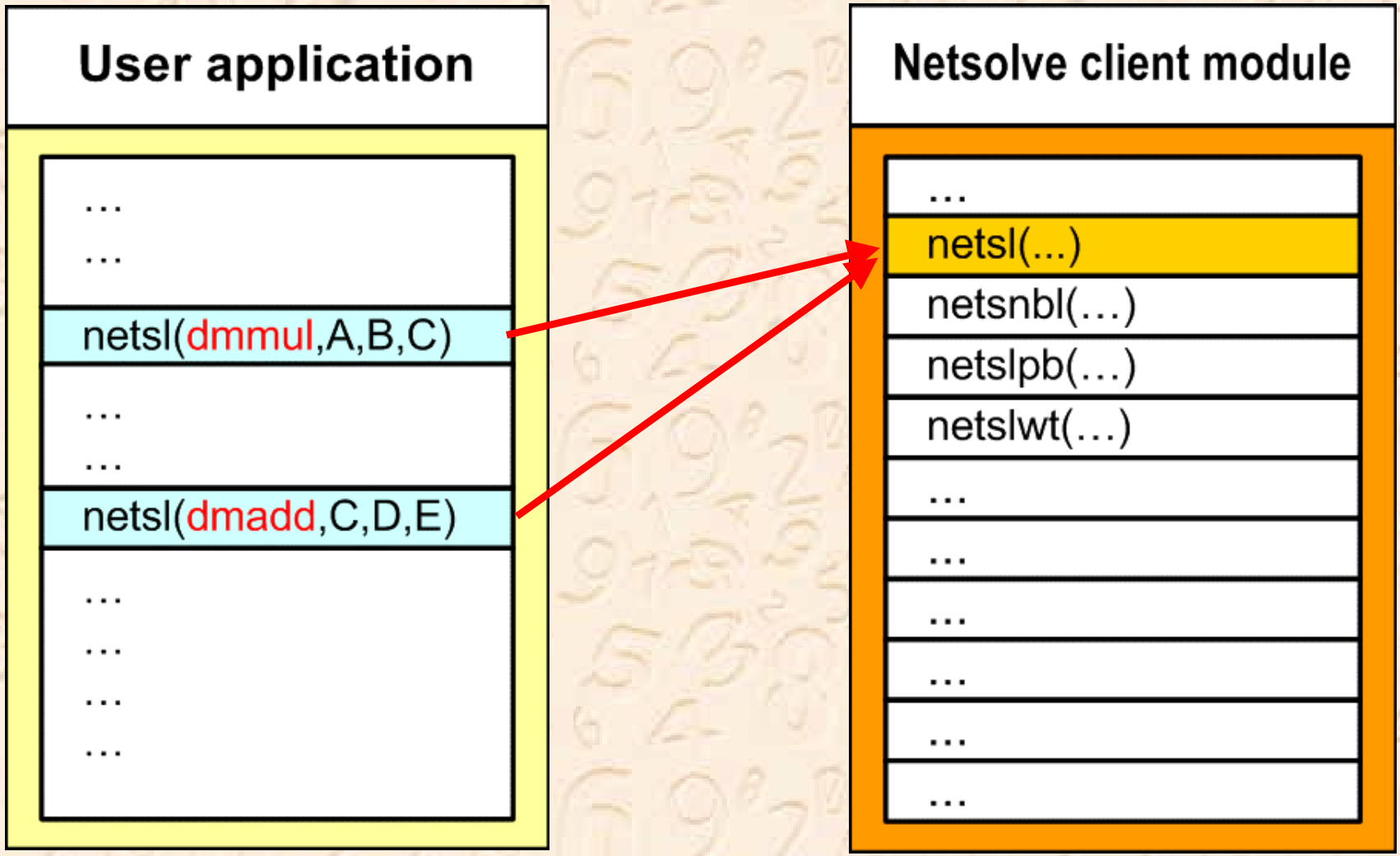


original application

Grid-enabled application

Example local->NetSolve call transition

(NetSolve example)



User application linked with NetSolve client module

How does GridRPC work? (NetSolve example)

User workstation

User application

call

reply

NetSolve client module

request

decision

NetSolve agent

Tasks running

Scheduling

remote call

task data

results

call reply

NetSolve server module

TR

NetSolve server module

TR

NetSolve server module

TR

Math library

dmmul (A,B,C)
dmadd (C,D,E)

Remote computing node

Math library

dmmul (A,B,C)
dmadd (C,D,E)

Remote computing node

Math library

dmmul (A,B,C)
dmadd (C,D,E)

Remote computing node

How does GridRPC work? (NetSolve example)

User workstation

User application

NetSolve client module

NetSolve agent

Tasks running

Resources database

Resources discovery

Scheduling

Performance database

Monitoring & prediction

NetSolve server module — TR | M & P | R R

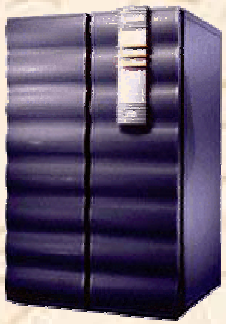
NetSolve server module — TR | M & P | R R

NetSolve server module — TR | M & P | R R

Math library

dmmul (A,B,C)
dmadd (C,D,E)

Remote computing node



Math library

dmmul (A,B,C)
dmadd (C,D,E)

Remote computing node

Math library

dmmul (A,B,C)
dmadd (C,D,E)

Remote computing node

GridRPC - how does it optimize the mathematical libraries usage? (1)

- **User's point of view**:
  - **simple client-side programming** and **code management**:
    - **no client-side management of IDLs** and/or RPC stubs
    - **high level API**
  - **no need to learning** complicated programming techniques (as in case of standard RPC, CORBA)
  - **no need to using** complicated job running techniques (as in case of Globus GRAM or queue systems)

GridRPC - how does it optimize the mathematical libraries usage? (2)

- **General point of view**:
  - **seamless, uniform and "transparent" access** to many versions and instances of libraries
  - **high degree of calls parallelism and distribution**
    - **various modes of remote calls** possible:
      - synchronous/asynchronous calls
      - callbacks
      - grouped calls ("transactions")
  - dynamic **resource discovery** mechanisms
  - simple **load-balancing** techniques (client-side) and mechanisms for **fault-tolerance** (*system agent*-side)

What is the subject of our work?

- **Easy adaptation** of the existing user applications to the remote mathematical libraries use
  - auto-adaptation techniques and tools
- **Increased fault-tolerance and performance** of the remote calls:
  - "emergency" and conditional local call execution mechanism
- **Incorporation of specific resources** to *the system*:
  - LSF and other queue systems "plug-ins"
- **Scheduling**:
  - better execution time prediction (also in the queue systems)

Auto-adaptation mechanisms - motivations

- **Many applications** that use mathematical libraries **exist** in our environment
- Currently the **source code of the existing application must be edited manually**
  to give the application access to the Grid resources through the Grid RPC technology
- This **can discourage the users** from using the GridRPC technology for the existing applications

Auto-adaptation mechanisms - solution

- **The source code pre-compiler:**
  - it will **automatically perform** the **exchanges** of local calls with Grid RPC calls **in the application source code**:

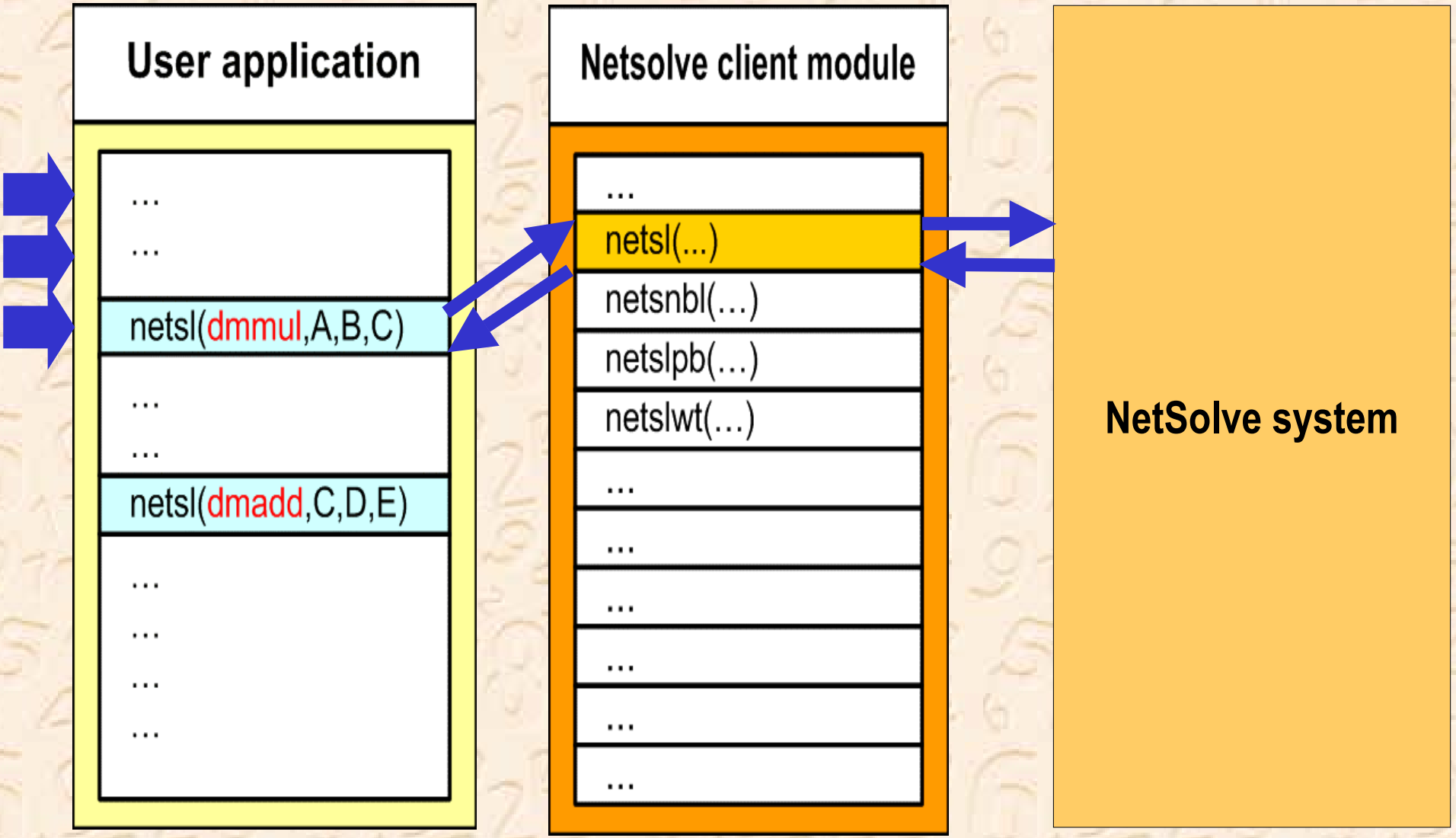
    local_call -> GridRPC_remote_call

  - the **list of known mathematical functions** will be acquired from the *system agent* database
  - each *local_call -> remote_call* exchange **may be confirmed by the user** (visual tool)
  - conversion of the local calls to *GridRPC API, Ninf API, NetSolve API* calls will be possible

Fault tolerance mechanism - motivations (1)

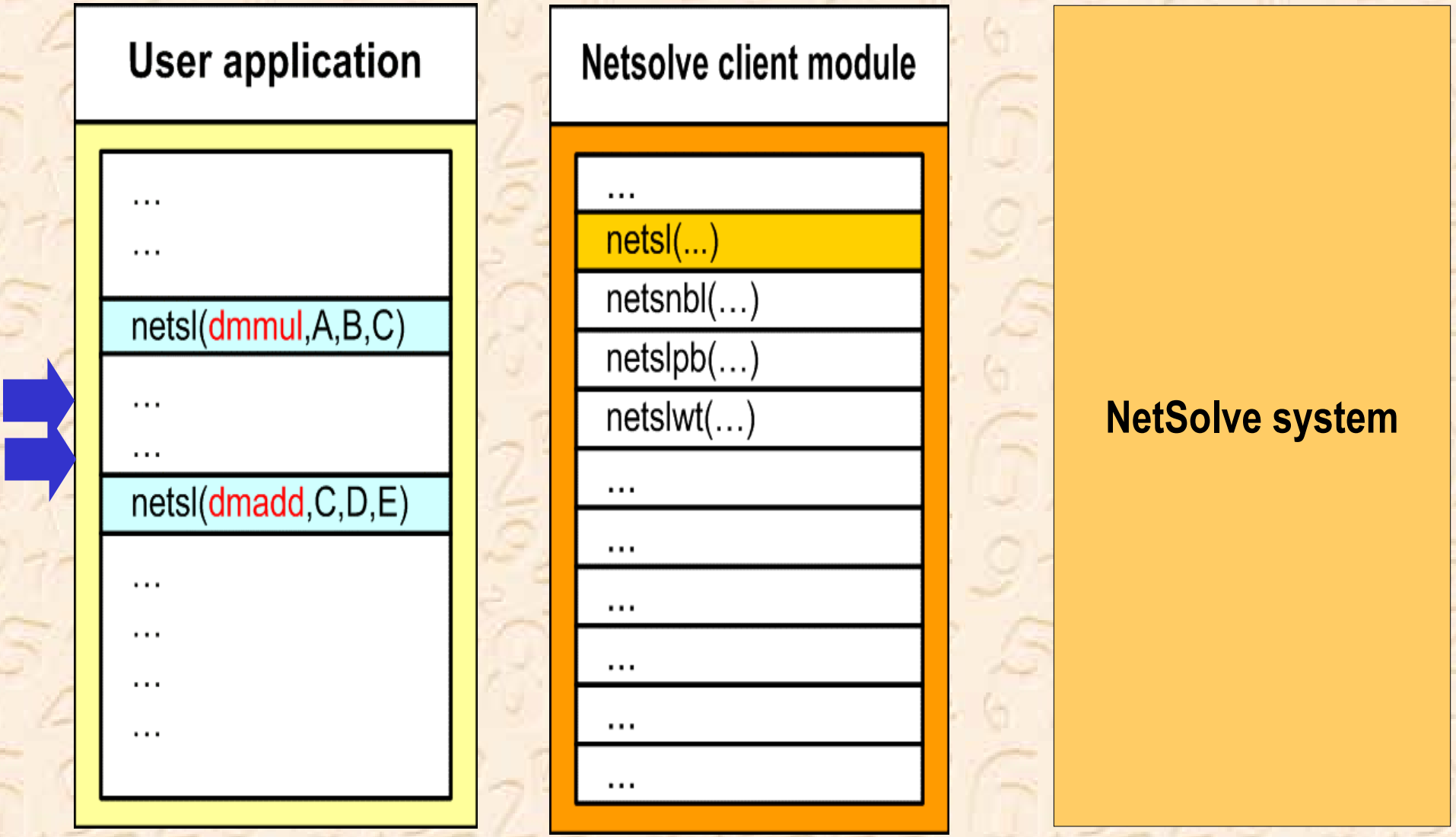
- Currently **there is no fault-tolerance mechanisms in the client module**:
  - **the decision whether** to perform the computations locally or remotely **is made once** (by the programmer)
  - if the programmer decides to perform the remote computations he possibly gains the better performance **but**:
    - **if the *system agent* fails**,
      the application execution is stopped
    - **if the network link fails**,
      (between the user's machine and the *system agent*'s machine)
      the application execution is stopped

Fault tolerance mechanism - motivations (2)



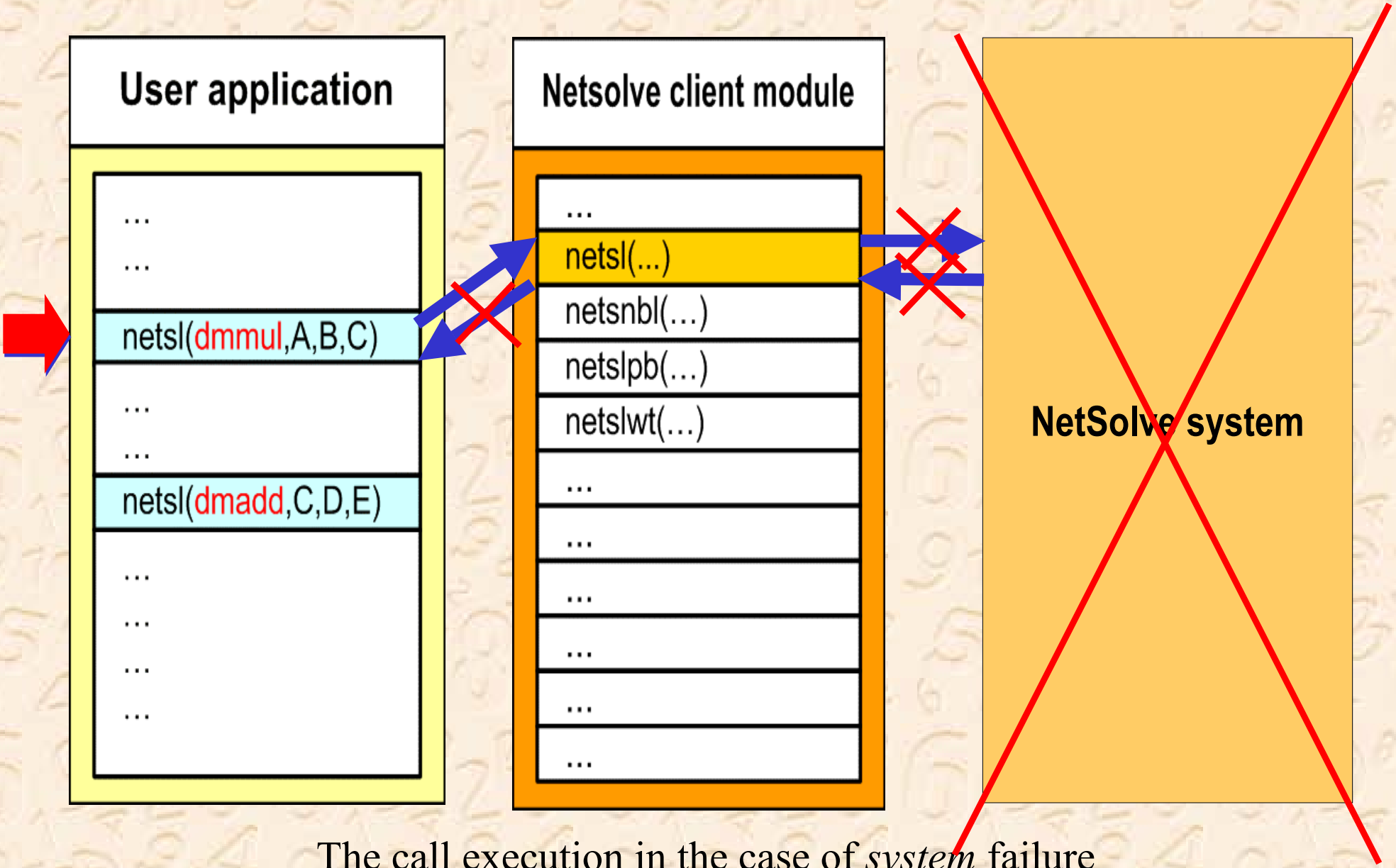
Normal call execution

Fault tolerance mechanism - motivations (2)



Normal call execution

Fault tolerance mechanism - motivations (2)



The call execution in the case of *system* failure

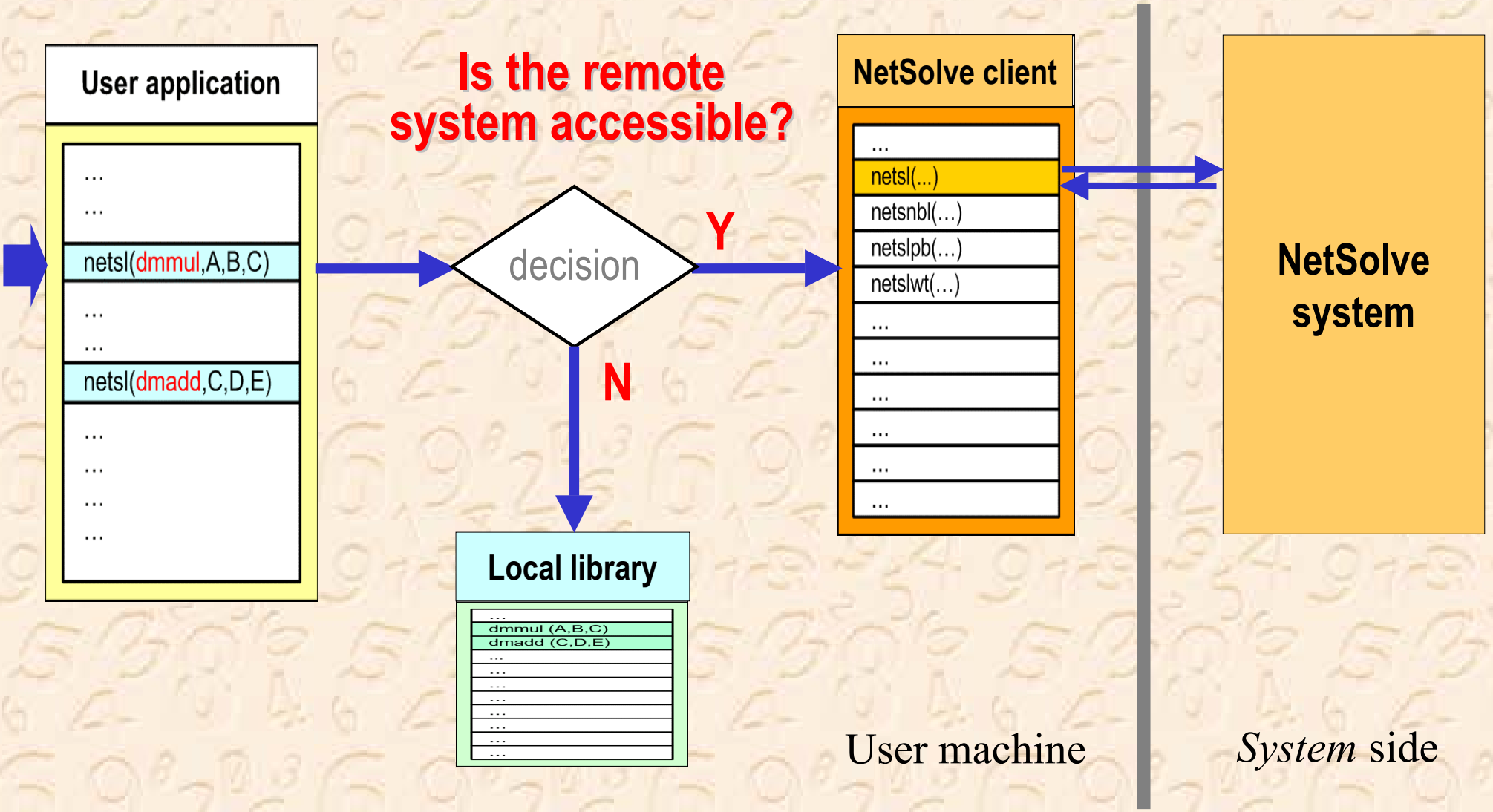
Fault tolerance mechanism - solution (1)

- **the decision** whether to perform the computation locally or remotely **should be made at the run-time**:
  - **if the *system agent* is accessible**,
    the call is scheduled by the *system agent* and the function call execution is done remotely
  - **if the *system agent* is unavailable**,
    mathematical function call is executed locally
    using the local copy of the mathematical library
  - **if the *system agent* becomes unavailable**
    while executing the remote call,
    the local function execution is launched
    (unless the local version of the given mathematical function is not available)

Fault tolerance mechanism - solution (2)

- **the "emergency" local call execution mechanism** can be exploited by linking the user application with **our own version of system client** (instead of the original *system client module*)
- the technique can be connected with the auto-adaptation mechanism or can be exploited using the calls of our **own version of API** (a robust one) instead of the GridRPC/Ninf/NetSolve API's
- we plan to implement this solution and possibly propose it as the external plug-in module to the existing GridRPC *systems*: Ninf and NetSolve

Fault tolerance mechanism - solution (3)



The remote call mechanism improved for the increased fault-tolerance

Client-side mechanisms for high-performance

motivations (1)

- **The profitability of the remote call is not considered**; currently there is an assumption made that the remote call execution is always faster:
  - **the decision whether** to perform the computations locally or remotely **is made once** (by the programmer)
  - if the programmer decides to perform the computations remotely, he **possibly** gains better performance i.e.:
    - **if the remote execution is faster** than the local one, **he gains it** (better performance)
    - **if the remote execution is slower** than the local one, **he loses it**

Client-side mechanisms for high-performance

motivations (2)

- **What could be improved?**
  - gain/lose probability is unpredictable:
    - **it depends on the configuration** and **the state** of the remote **computational resources**
    - **it depends on the state** of the **network** links
  - however **something is predictable**:
    - one may observe, analyze and predict:
      - **the "behavior" of the local function** executions (times, computational complexity etc.)
      - the **local machine state** (to predict the local computations time)
      - the **network state** (to predict the amount of time needed to transmit the input/output data)

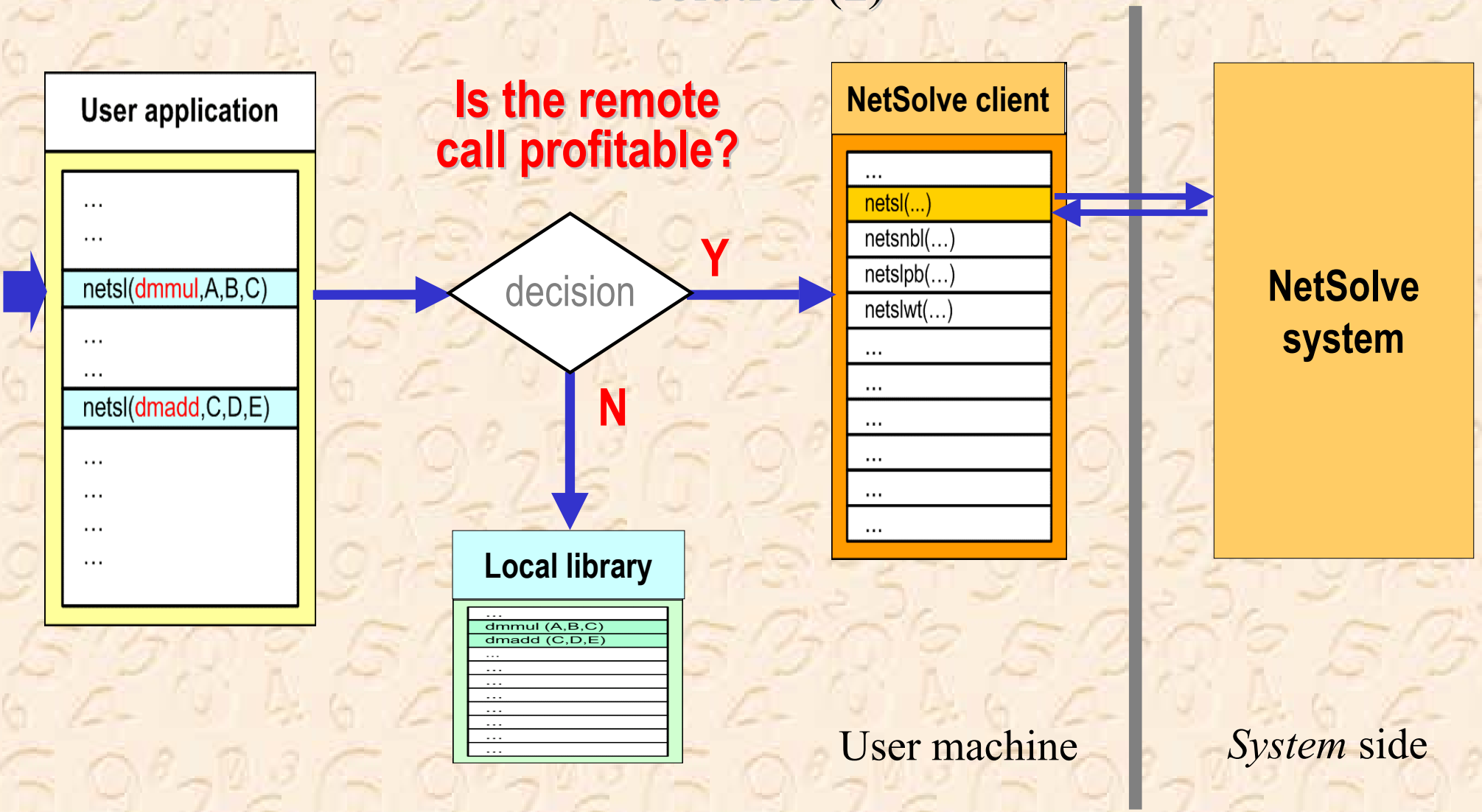
Client-side mechanisms for high-performance

solution (1)

- **the decision** whether to perform computations locally or remotely **should be made at the run-time**:
  - one may use the local copy of the library if:
    - the predicted **local execution time is shorter** than the predicted remote execution time
    - there is **no free remote resources** at the given moment (i.e. waiting for free resources is senseless)
    - the observed **network performance is poor** and the task (function call) input data are large
  - **the effectiveness of this mechanism strongly depends on the accuracy of prediction techniques**
    - we are going to implement and **test** this solution;
    - we will put it into practice only if the results will be promising

Client-side mechanisms for high-performance

solution (2)



The remote call mechanism improved for the increased performance

Client-side mechanisms - summary

- **what would be improved?**
  - **fault tolerance** of remote calls **- will be achieved**
    - i.e. the user application is going to work **regardless** of the possible *system* fails or unavailability of the remote resources
  - **performance** of the application execution **would be improved**
    - the profitability of the remote call is going to be considered:
    - the remote call would not to be made when it seems to be not profitable

Specific resources incorporation - motivations (1)

- **“to optimize”** means also **“to give the user application access to all the computational resources that can be used”**
- **a lot of the computational resources** in the polish scientific environment **work under the control of the queue systems** (e.g. LSF, NQE, PBS)
- the existing Grid RPC *systems* **have** the *plug-ins* for the Condor and Globus brokers but **have no** *plug-ins* for the queue systems
- **the queue systems *plug-ins* for Ninf and NetSolve should be developed**

Specific resources incorporation - motivations (2)

- **why not use the Globus/Legion interfaces and its interfaces to the queue systems?**
  - the Globus / Legion mechanisms bring considerable **overhead**
  - Ninf / NetSolve were designed to be simple and highly efficient
  - the **Ninf / NetSolve dedicated mechanisms** seem to be **useful**
- **why not launch the Ninf /NetSolve computations "directly" on the resources omitting "the queues"?**
  - each site has its own **resource management policies**
  - these policies **must be respected**

Specific resources incorporation - solution

- **The LSF and NQE *plug-ins* for Ninf and NetSolve will be developed**
- The *plug-ins* are going to realize:
  - mechanisms for running mathematical function calls as the queue systems' batch jobs
  - jobs execution time prediction mechanisms, including:
    - prediction of the time the task is going to spend in the queue
    - prediction of the task execution time
  - mechanisms for monitoring and predicting the state of the resources controlled by the queue systems

Specific resources incorporation - summary

- Ninf and NetSolve *plug-ins* for LSF and NQE will give the computational applications **access** to all the computational **resources** that work **under the control of the queue systems**
- **the access method will be effective**

  (more effective than the access using existing technologies - Globus, Legion etc.)

Scheduling techniques improvement - motivations (1)

- The scheduling techniques in existing *systems* are general:
  - the characteristics of the individual resource are not considered
  - simple performance models are used
  - knowledge, history, statistics... - not exploited
- (polish) Grid environment:
  - various types of machines (SMPs, MPPs, clusters, queue systems); each has a “predisposition” to solve different classes of mathematical problems
  - various implementations of mathematical libraries (open source, free vs highly optimized, commercial: e.g. NAG)

Scheduling techniques improvement - motivations (2)

- (polish) Grid environment (continued):
  - very dynamic:
    - new **resources added**
    - existing resources' **configuration changed** (number of processors, memory amount)
    - resource management **policies changed** (permanently or temporarily)
    - ...

Scheduling techniques improvement - motivations (3)

– **The execution time of the mathematical function call is much more predictable than the execution time of the "general task":**

- **computational complexity function** of the given mathematical function implementation **is known** (specified by the library supplier) **or can be discovered** (by performing a series of computing experiments) (accurate methods)
- **the size of the problem is known or can be determined** (basing on the input data size)
- **the convergence coefficient is known or can be discovered** (approximate methods)

Scheduling techniques improvement - solution (1)

- **The scheduling techniques could be improved:**
  - one may **collect the historical data** concerning the individual calls execution
  - and **perform the statistical analysis of these data**
    - to **discover the trends, rules of the "behavior" of the function executions** performed **on the given resource**

      i.e. the relationship (link) between:
      - » he call execution time
      - » the problem size
      - » the current and predicted state of the resource used

    - to **discover the predisposition** of the individual resources **to solve the given math. problems** or the classes of the problems

      e.g. "resource A performs well when solves vector tasks"

Scheduling techniques improvement - solution (2)

- **What is needed to achieve that?**
  - **mechanisms for collecting the historical data:**
    - mechanisms for monitoring the state of the systems at the moment of the function execution (e.g. NWS could be used)
    - mechanism for collecting, storing and accessing the historical data
  - **mechanisms for analysing the historical data:**
    - **eliminating the influence of the temporary system state/load on the performance** gained by the computations:
      - various **existing performance models** should be implemented and tested;
      - new models should be developed (if needed)

Scheduling techniques improvement - summary

- the **new scheduling algorithms**
  based on the discussed mechanisms
  will be designed and tested
  in the real environments
- if the results of the researches will be promising,
  the developed scheduling techniques
  will be put into practice in the Polish Grid environment

Presentation summary

The state of the art:

- There are some techniques that optimize the usage of the mathematical libraries in the Grid environment
- The most promising is the Grid RPC technology
- The Grid RPC is implemented in the *Ninf* and *NetSolve* systems

The subject of our work:

- **PSNC accept these systems** as the base of our *system*, we put these systems into practice, test and evaluate them
- we also **develop new technologies** for such systems:
  - **auto-adaptation tools** for existing applications
  - **client-side mechanisms** for the **fault-tolerance** and the **high-performance**
  - the **queue system *plug-ins***
  - improved **scheduling mechanisms** (history and knowledge based)

Final remarks

- We do not want to change the existing *systems* (Ninf/NetSolve):
  - they still evolve, the implementation's details are changed
  - new features are added, new technologies are worked out
- but we want to exploit their advantages that is why:
  - we develop the tools "around" them
  - we propose *plug-ins* to them

  in order **to put effectively these systems into practice** in our environment
- The main point of our interest is the development of the new technologies for the Grid RPC mechanisms:
  - we test the concepts in real systems
  - but research results may be used in a wider perspective

Final remarks

- We don't want to change the existing systems (Ninf/NetSolve) as:
  - they still evolve, the implementation's details are changed
  - the new features are added, the new technologies are worked out
- but we want to exploit their good qualities:
  - we develop the tools "around" them(?)
  - we propose *plug-ins* to them in order to put these systems effectively into practice
- The main point of our interest is the development of the new technologies for the Grid RPC mechanisms:
  - we test the concepts in real systems
  - but the the research results may be used much more broadly (perhaps to make the execution of the individual Grid application's parts on the distributed Grid nodes efficient, robust and easy for the user)

Thanks for attention!

Questions welcome.