

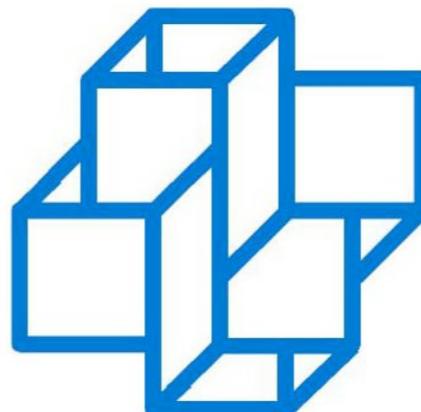
Experiments with a Multi-Language Approach to Implementing Multiscale Hybrid-Mixed Methods

Antônio Tadeu A. Gomes

Diego Paredes

Frédéric Valentin

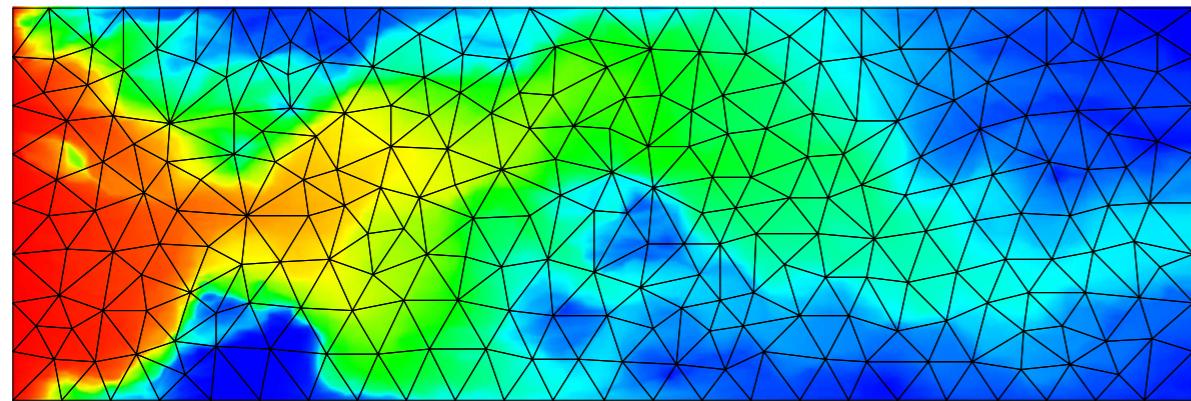
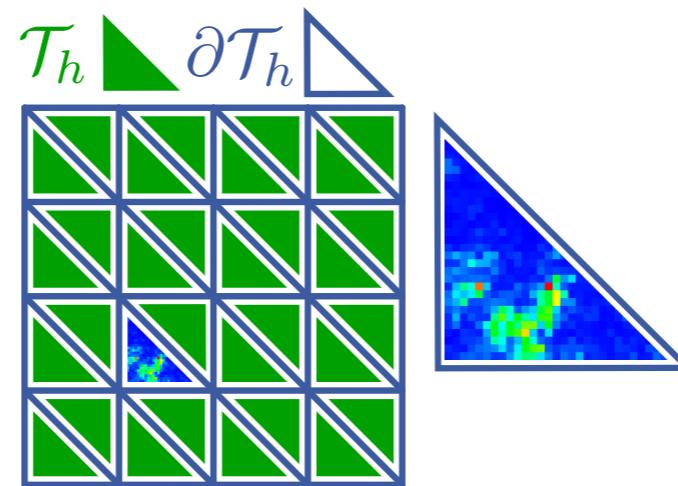
atagomes@lncc.br



Laboratório
Nacional de
Computação
Científica

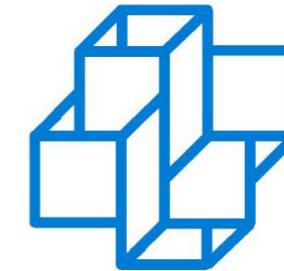
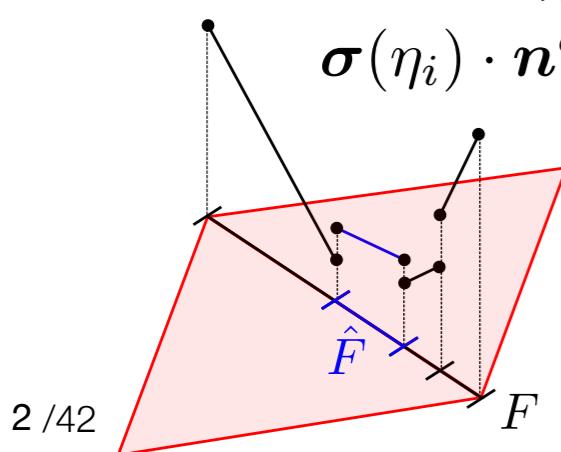


The MHM method in a (shameful) nutshell...



Find η_i such that

$$\left. \begin{array}{l} \mathcal{L}\eta_i = 0, \text{ in } K \\ \boldsymbol{\sigma}(\eta_i) \cdot \mathbf{n}^{\partial K} = \psi_i, \text{ on } \partial K \end{array} \right\}$$



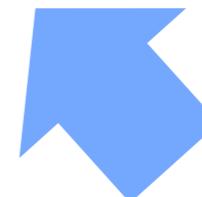
Find $\lambda_h \in \Lambda_h$ such that

$$(u^{\lambda_h}, \mu_h)_{\partial\mathcal{T}_h} = -(u^f, \mu_h)_{\partial\mathcal{T}_h}$$

for all $\mu_h \in \Lambda_h$

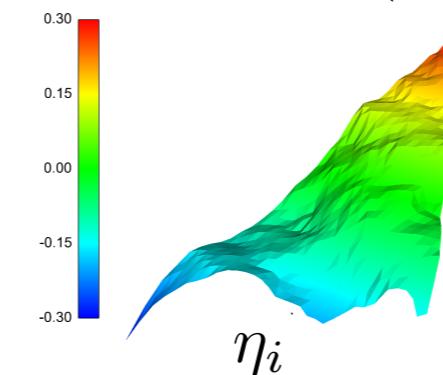


$$u_h = u^{\lambda_h} + u^f = \sum_{i=1}^{N_h} c_i \eta_i + u^f$$



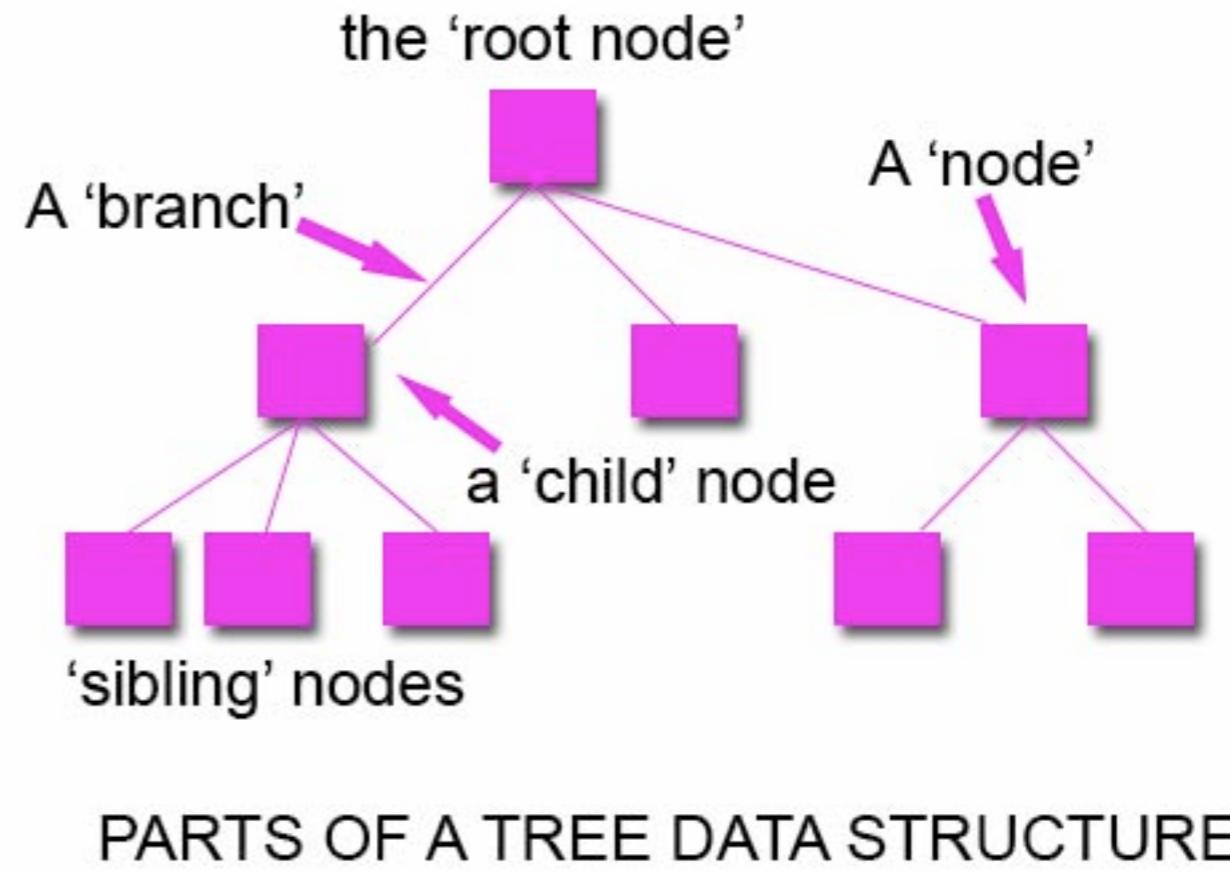
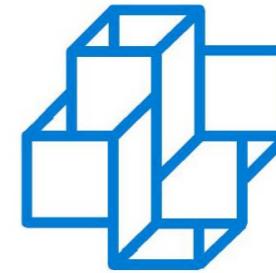
Find u^f such that

$$\left. \begin{array}{l} \mathcal{L}u^f = f, \text{ in } K \\ \boldsymbol{\sigma}(u^f) \cdot \mathbf{n}^{\partial K} = 0, \text{ on } \partial K \end{array} \right\}$$

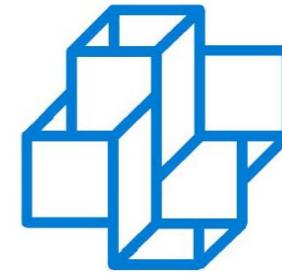


Difficulties

- Remembering SPiNMe
(Lua+NeoPZ...)
- Implementing the specific family of MHM methods
 - NeoPZ's architectural drift
- Implementation of a new FEM library specifically crafted for exploring the loosely-coupled strategy of MHM methods to solve global and local problems



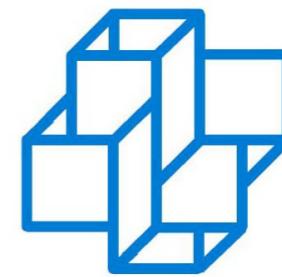
Source: www.teach-ict.com



Key design principles

- Multi-language approach:
 - Erlang for distribution, communication and fault tolerance
 - C++ for number crunching
 - (In the future) Lua for rapid prototyping and high-level configuration





Key design principles

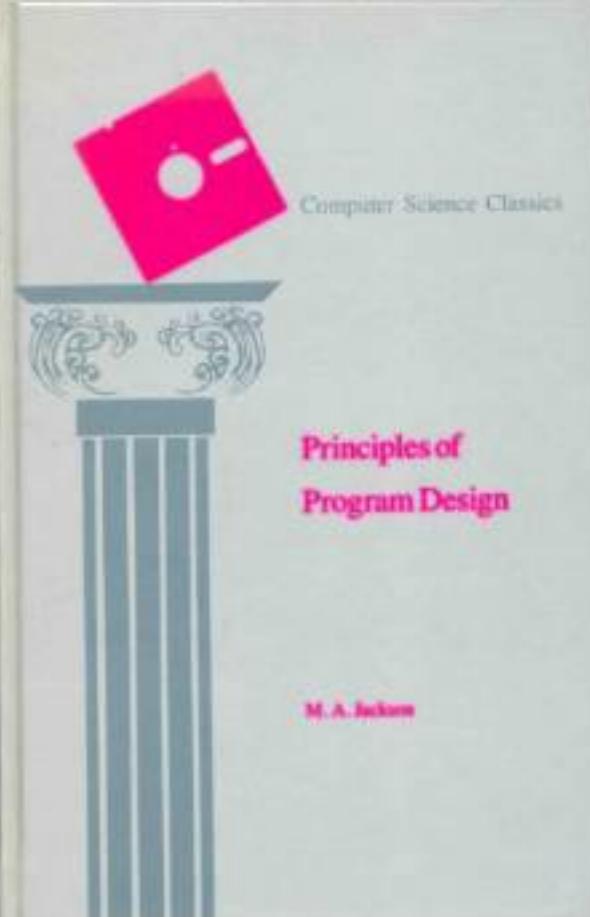
<https://quotabl.es>

The First Rule of Program Optimization: Don't do it. The Second Rule of Program Optimization (for experts only!): Don't do it yet.

— Michael A. Jackson

<http://mcs.open.ac.uk/mj665/>





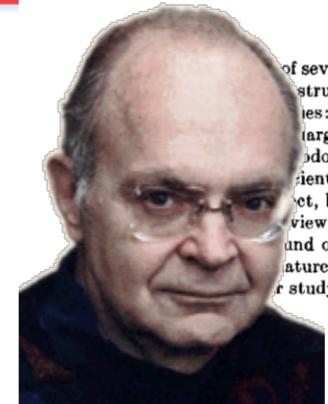
Premature optimization is the root of all evil.

— Donald Knuth in 1974

<http://www-cs-faculty.stanford.edu/~uno/>

Structured Programming with go to Statements

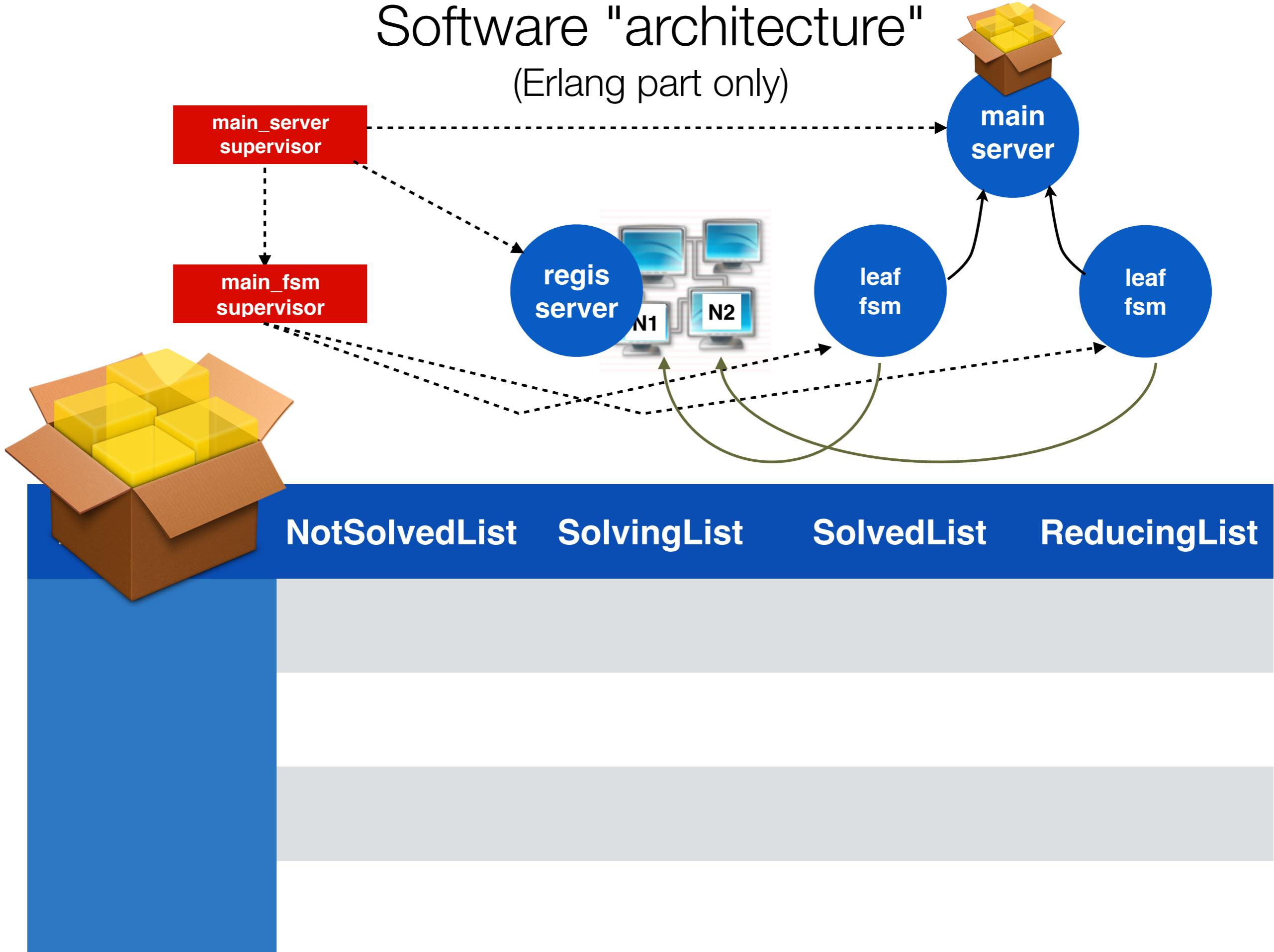
DONALD E. KNUTH
Stanford University, Stanford, California 94305

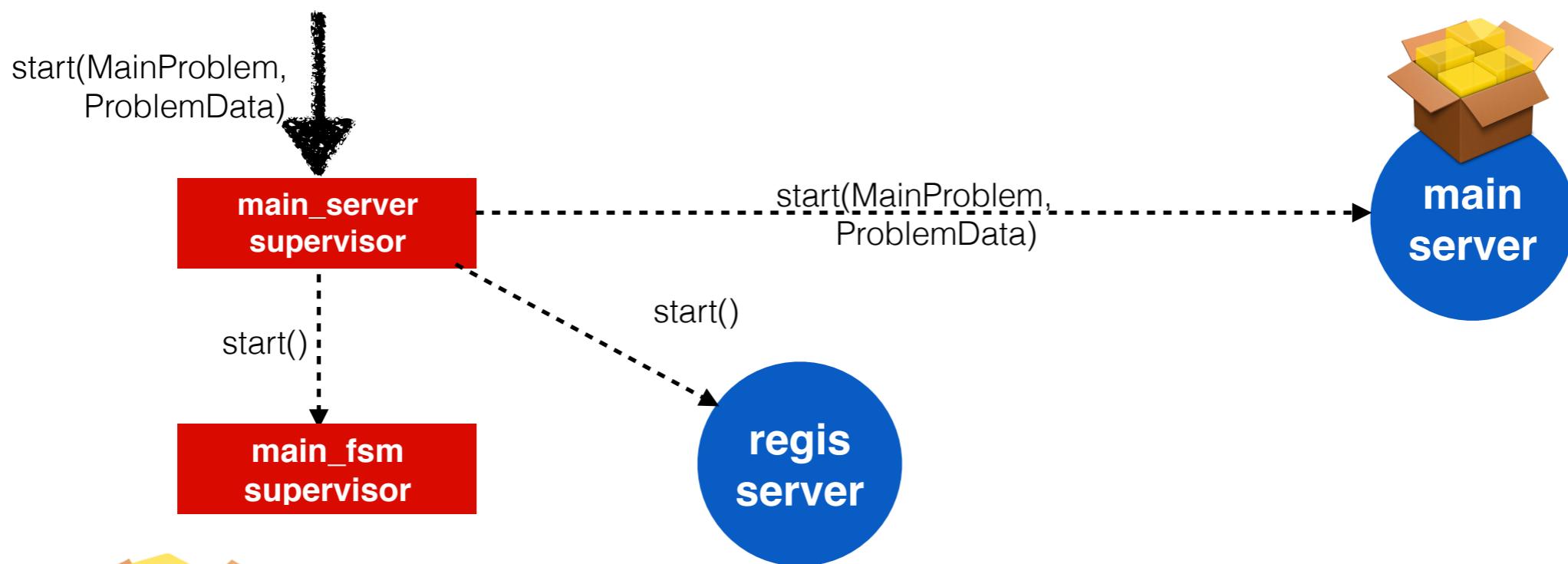


of several different examples sheds new light on the problem of creating structured programs that behave efficiently. This study focuses on: (a) improved syntax for iterations and error exits, making it easier to write a larger class of programs clearly and efficiently without `go to` statements; (b) a methodology of program design, beginning with readable and correct, efficient programs that are systematically transformed if necessary into shorter, but possibly less readable code. The discussion brings out open questions about whether or not `go to` statements should be abolished; and on both sides of this question. Finally, an attempt is made to summarize the nature of structured programming, and to recommend fruitful directions for future research.

Software "architecture"

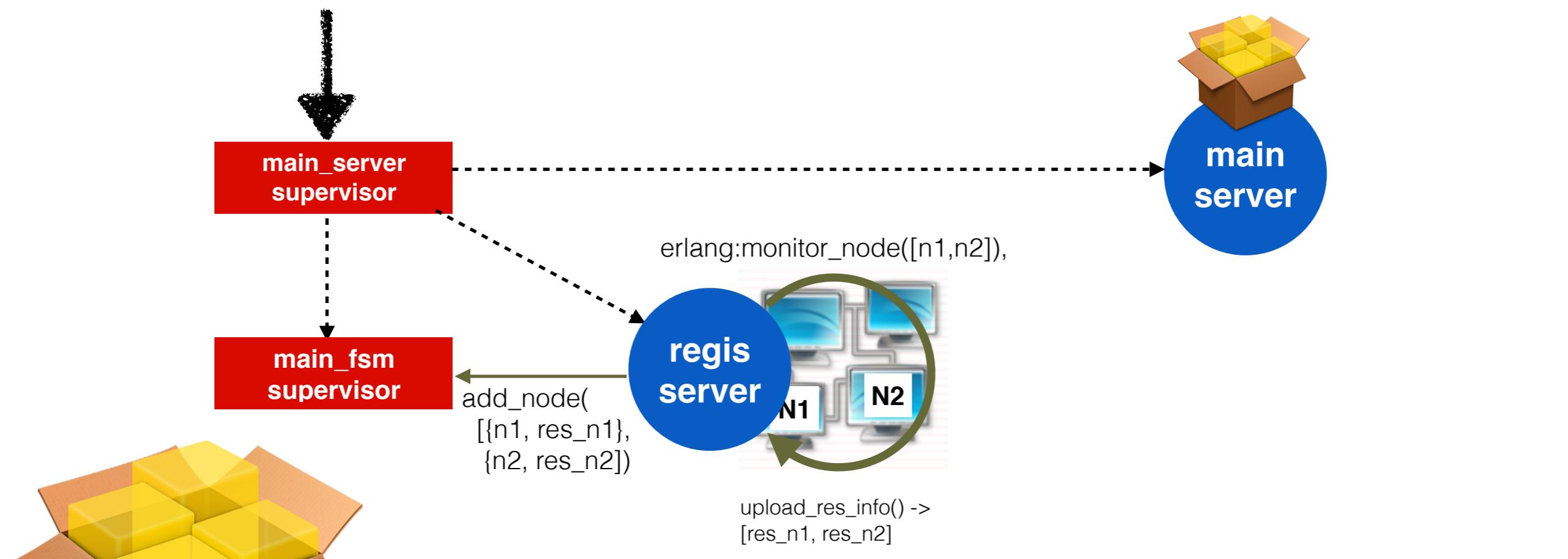
(Erlang part only)



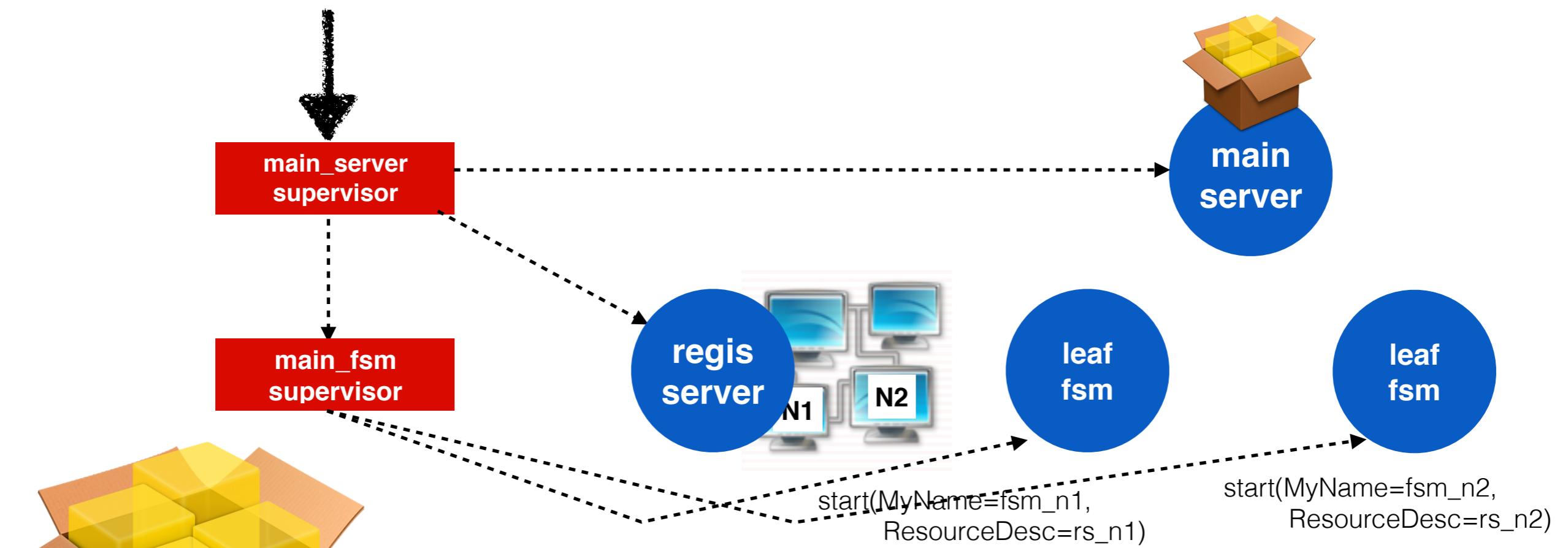


	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem	[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]	[]	[]	[]

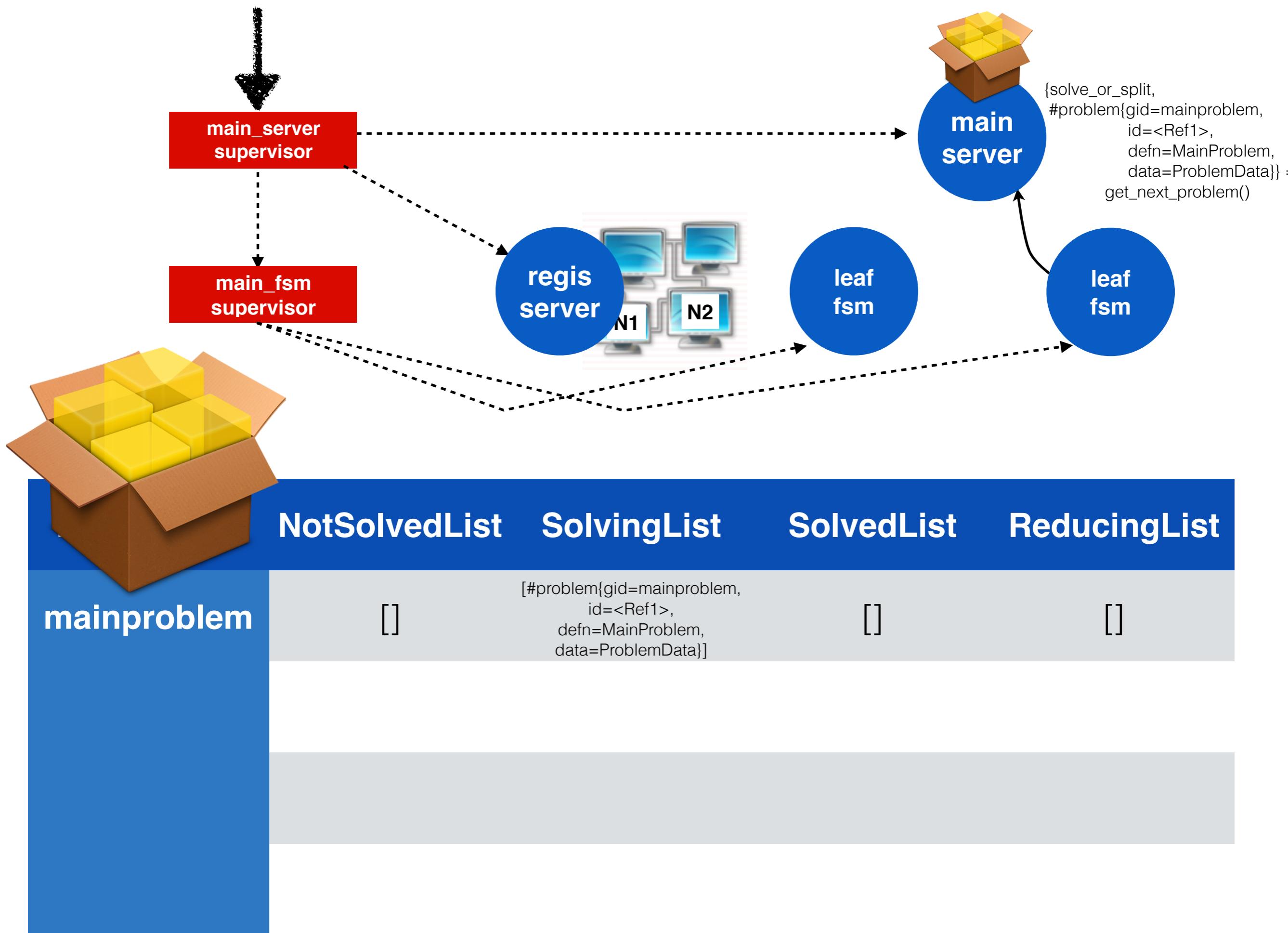
The table shows the initial state of the `mainproblem` component. It contains four lists: `NotSolvedList`, `SolvingList`, `SolvedList`, and `ReducingList`. The `NotSolvedList` contains one entry, which is a problem definition with `gid=mainproblem`, `id=<Ref1>`, `defn=MainProblem`, and `data=ProblemData`. The other three lists are currently empty.

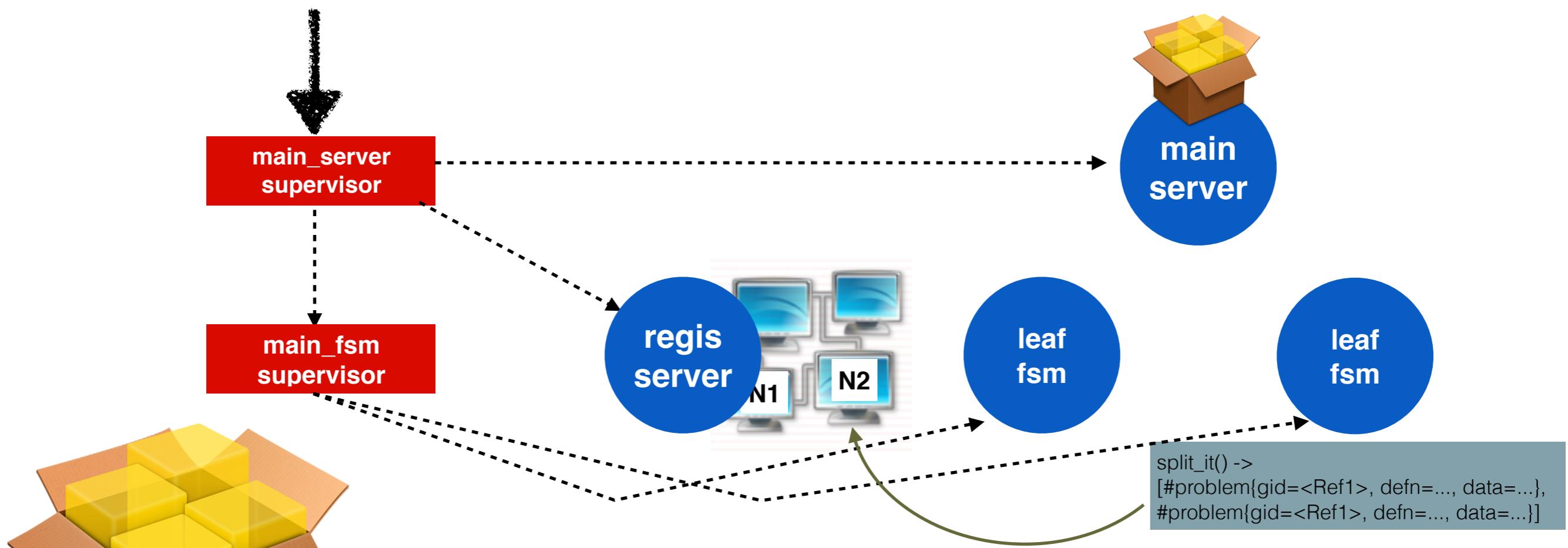


	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem	<pre>[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]</pre>	[]	[]	[]

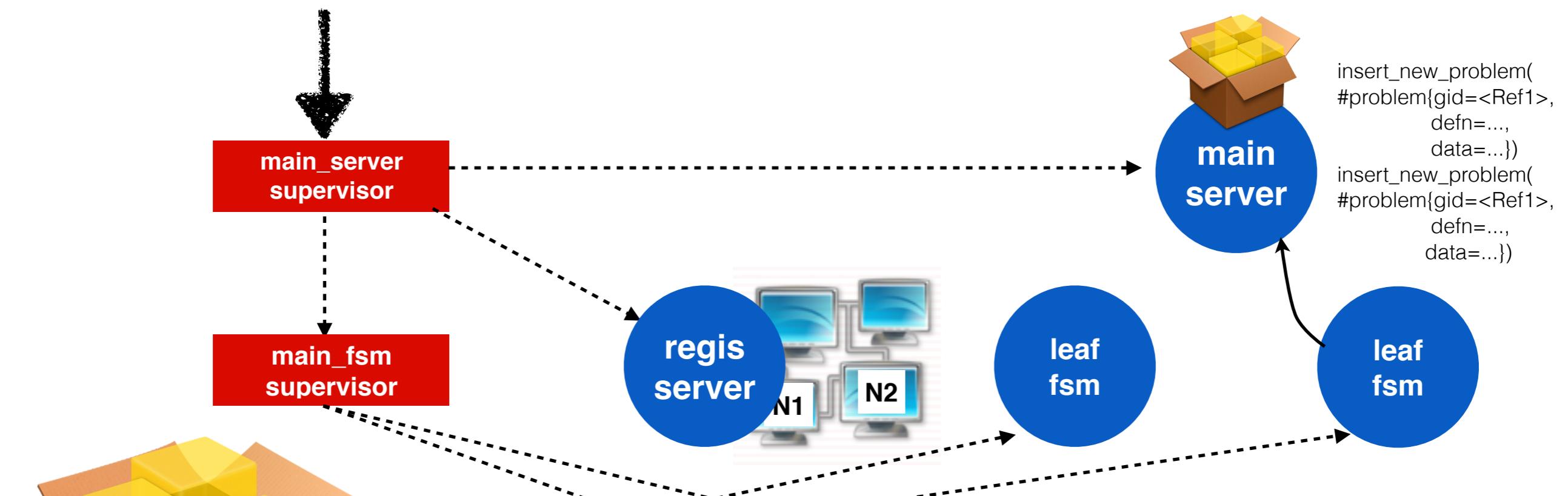


	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem	[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]	[]	[]	[]

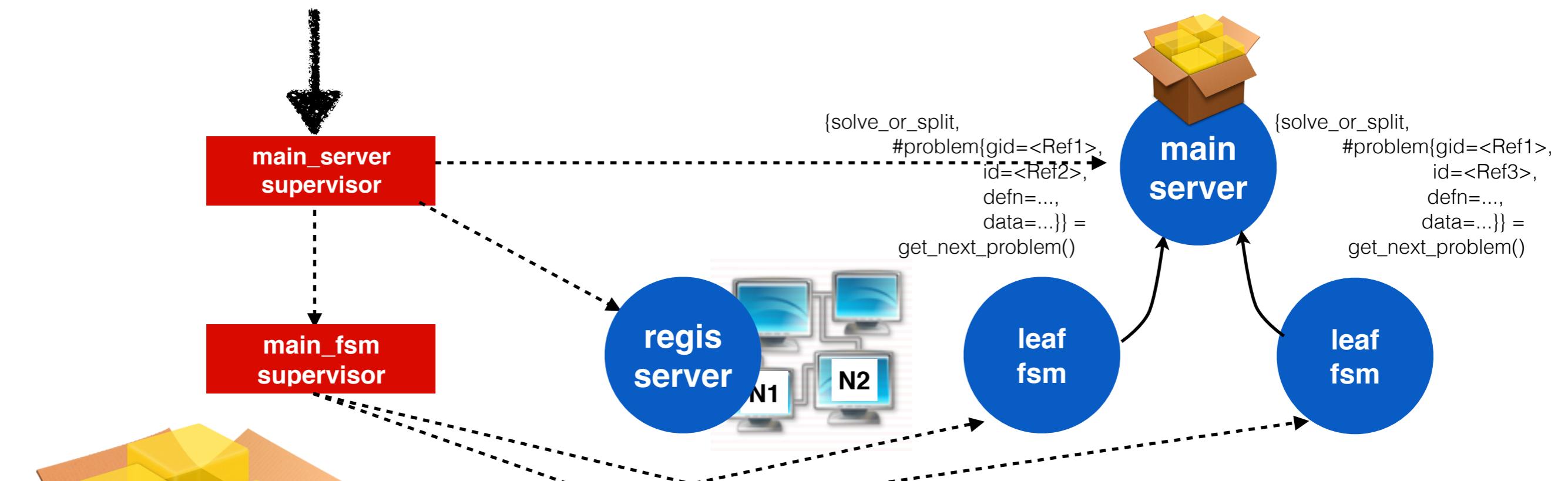




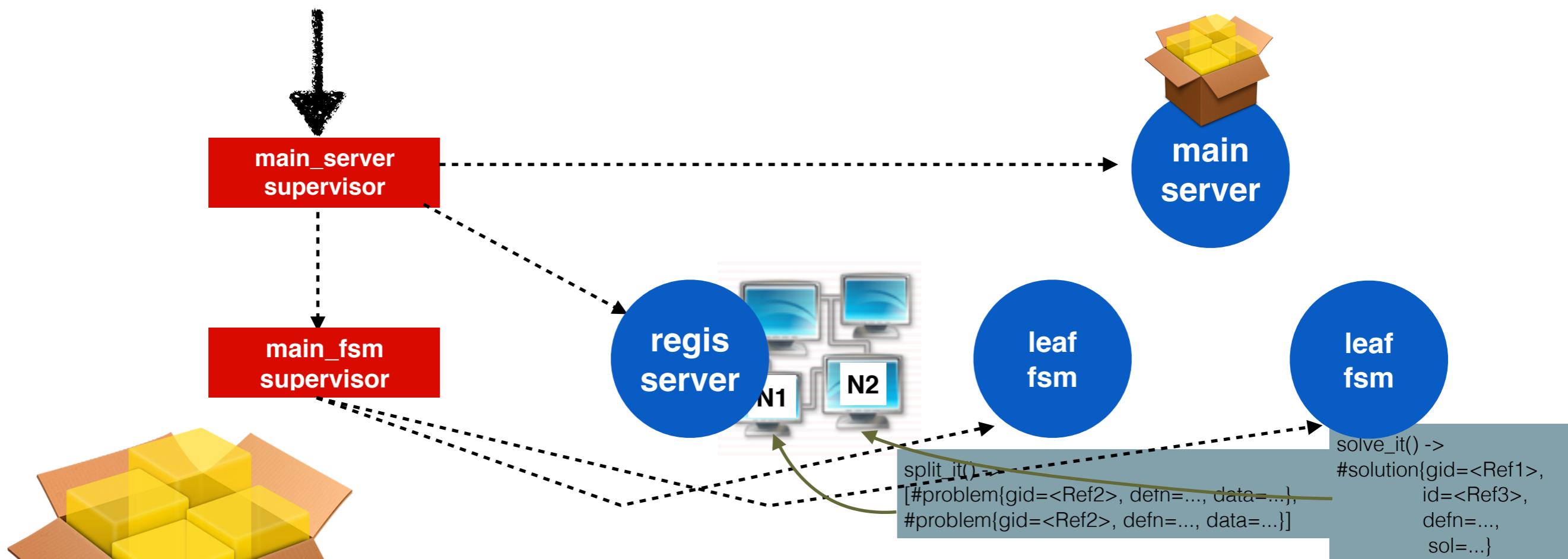
	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem	[]	[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]	[]	[]



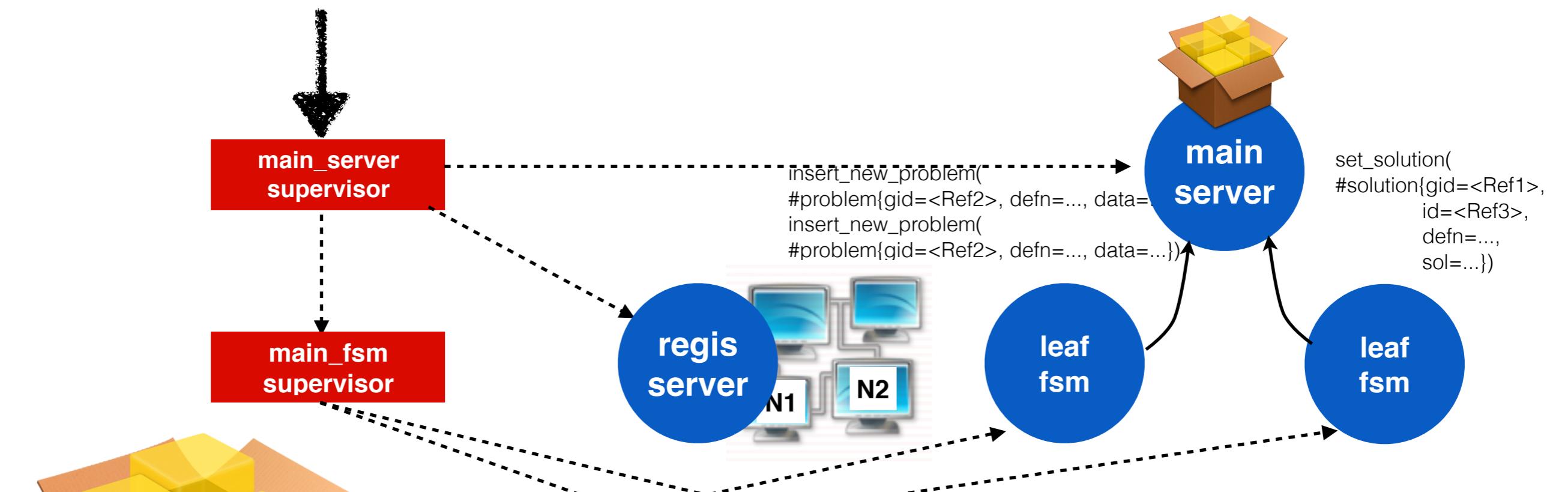
	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem				
<Ref1>	<pre>[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]</pre> <pre>[#problem{gid=<Ref1>, id=<Ref3>, defn=.., data=..}, #problem{gid=<Ref1>, id=<Ref2>, defn=.., data=..}]</pre>	<pre>[]</pre>	<pre>[]</pre>	<pre>[]</pre>



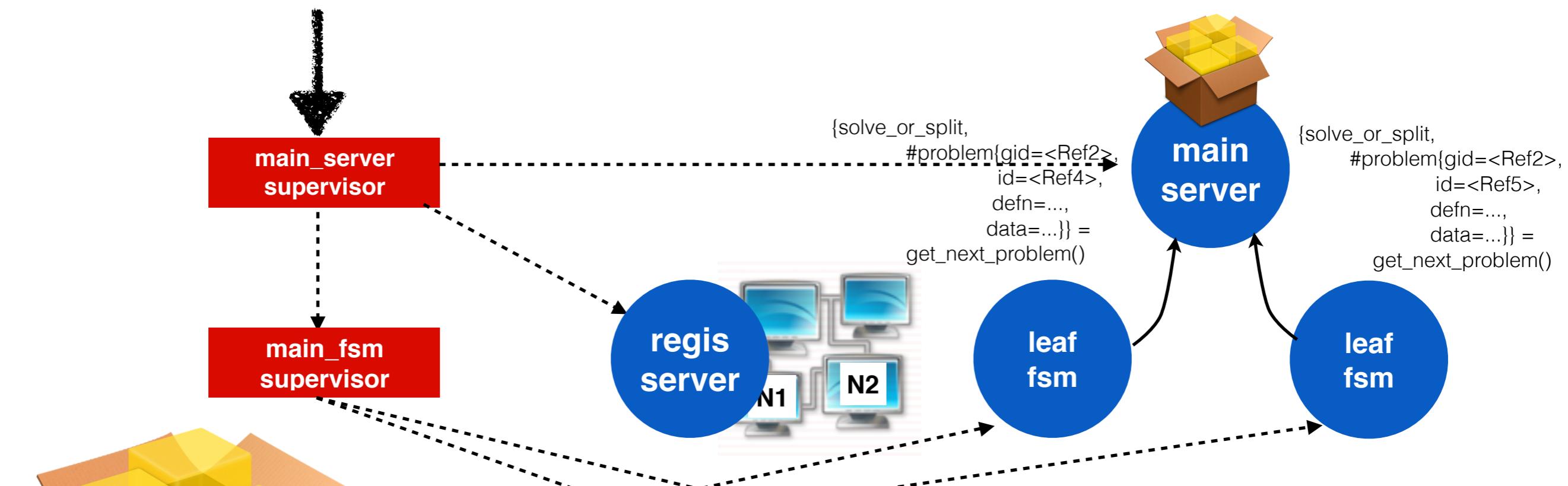
	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem		[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]		
<Ref1>	[]	[#problem{gid=<Ref1>, id=<Ref2>, defn=.., data=..}, #problem{gid=<Ref1>, id=<Ref3>, defn=.., data=..}]	[]	[]



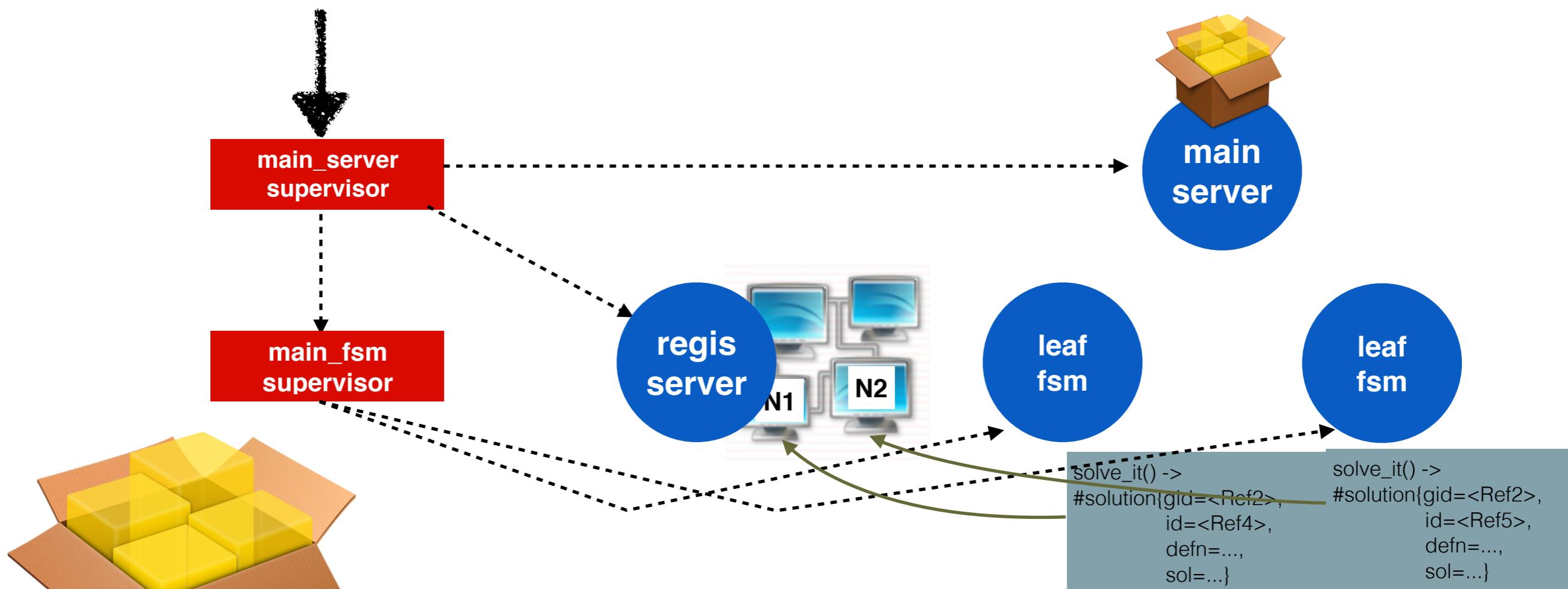
	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem	[]	[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]	[]	[]
<Ref1>	[]	[#problem{gid=<Ref1>, id=<Ref2>, defn=.., data=..}, #problem{gid=<Ref1>, id=<Ref3>, defn=.., data=..}]	[]	[]



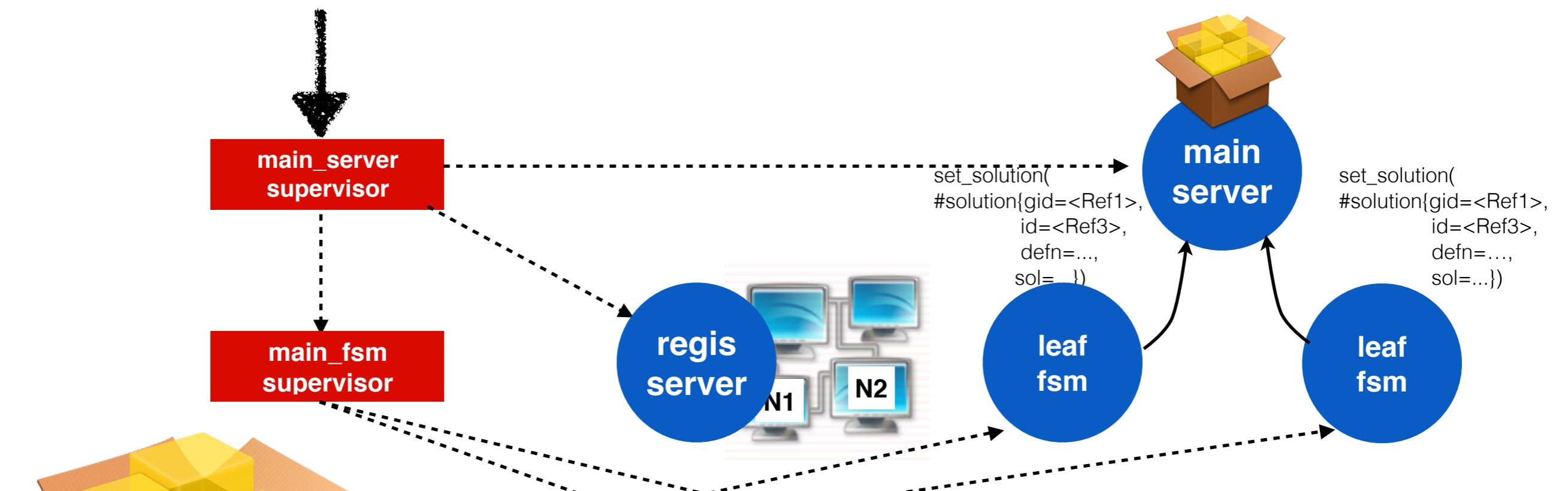
	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem	[]	[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]	[]	[]
<Ref1>	[]	[#solution{gid=<Ref1>, id=<Ref2>, defn=..., sol=...}]	[#solution{gid=<Ref1>, id=<Ref3>, defn=..., sol=...}]	[]
<Ref2>	[#problem{gid=<Ref2>, id=<Ref5>, defn=.., data=..}, #problem{gid=<Ref2>, id=<Ref4>, defn=.., data=..}]	[]	[]	[]



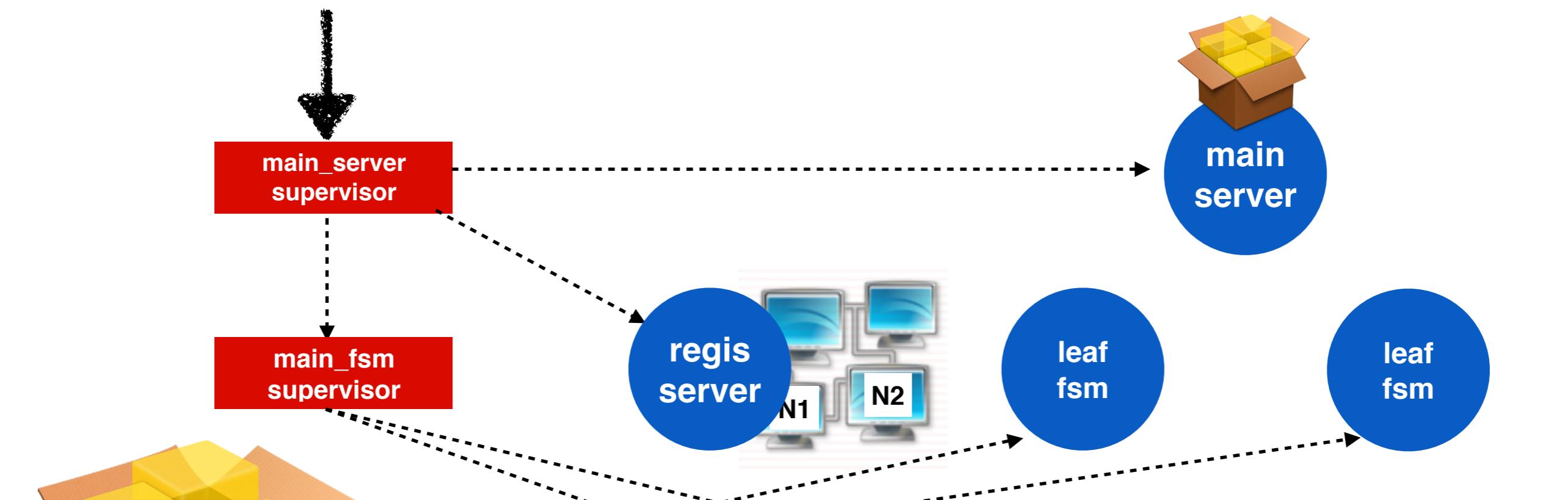
	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem		[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]		
<Ref1>	[]	[#solution{gid=<Ref1>, id=<Ref2>, defn=.., sol=..}]	[#solution{gid=<Ref1>, id=<Ref3>, defn=.., sol=..}]	
<Ref2>	[]	[#problem{gid=<Ref2>, id=<Ref4>, defn=.., data=..}, #problem{gid=<Ref2>, id=<Ref5>, defn=.., data=..}]	[]	



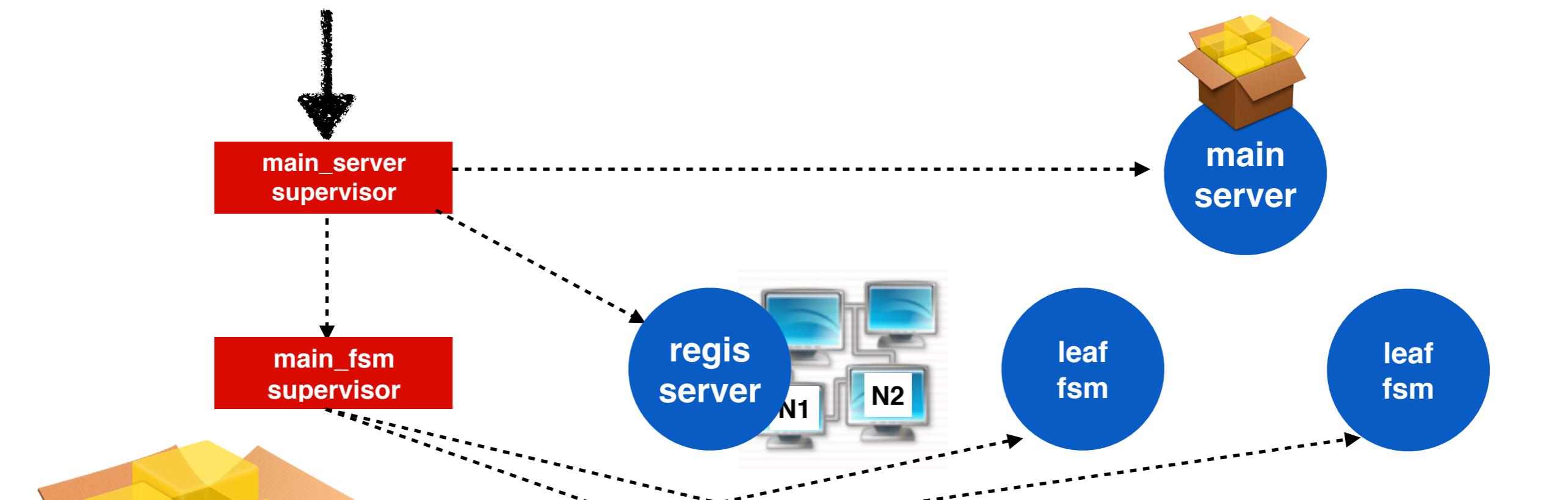
	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem	[]	[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]	[]	[]
<Ref1>	[]	[#solution{gid=<Ref1>, id=<Ref2>, defn=..., sol=...}]	[#solution{gid=<Ref1>, id=<Ref3>, defn=..., sol=...}]	[]
<Ref2>	[]	[#problem{gid=<Ref2>, id=<Ref4>, defn=.., data=..}, #problem{gid=<Ref2>, id=<Ref5>, defn=.., data=..}]	[]	[]



	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem	[]	[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]	[]	[]
<Ref1>	[]	[#solution{gid=<Ref1>, id=<Ref2>, defn=.., sol=..}]	[#solution{gid=<Ref1>, id=<Ref3>, defn=.., sol=..}]	[]
<Ref2>	[]	[]	[#solution{gid=<Ref2>, id=<Ref4>, defn=.., sol=..}, #solution{gid=<Ref2>, id=<Ref5>, defn=.., sol=..}]	[]

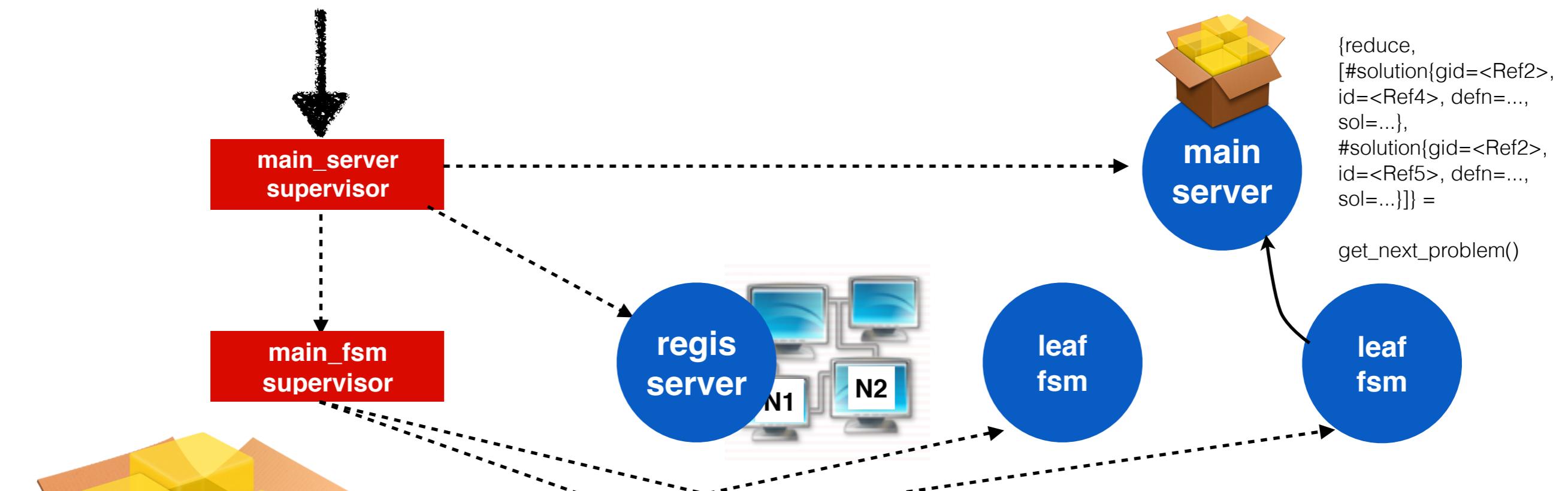


	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem	[]	[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]	[]	[]
<Ref1>	[]	There's no more items in NotSolvedList, so let's start reducing some problems!!!	[#solution{gid=<Ref1>, id=<Ref2>, defn=..., sol=...}]	[#solution{gid=<Ref1>, id=<Ref3>, defn=..., sol=...}]
<Ref2>	[]		[#solution{gid=<Ref2>, id=<Ref4>, defn=.., sol=..}, #solution{gid=<Ref2>, id=<Ref5>, defn=.., sol=..}]	[]

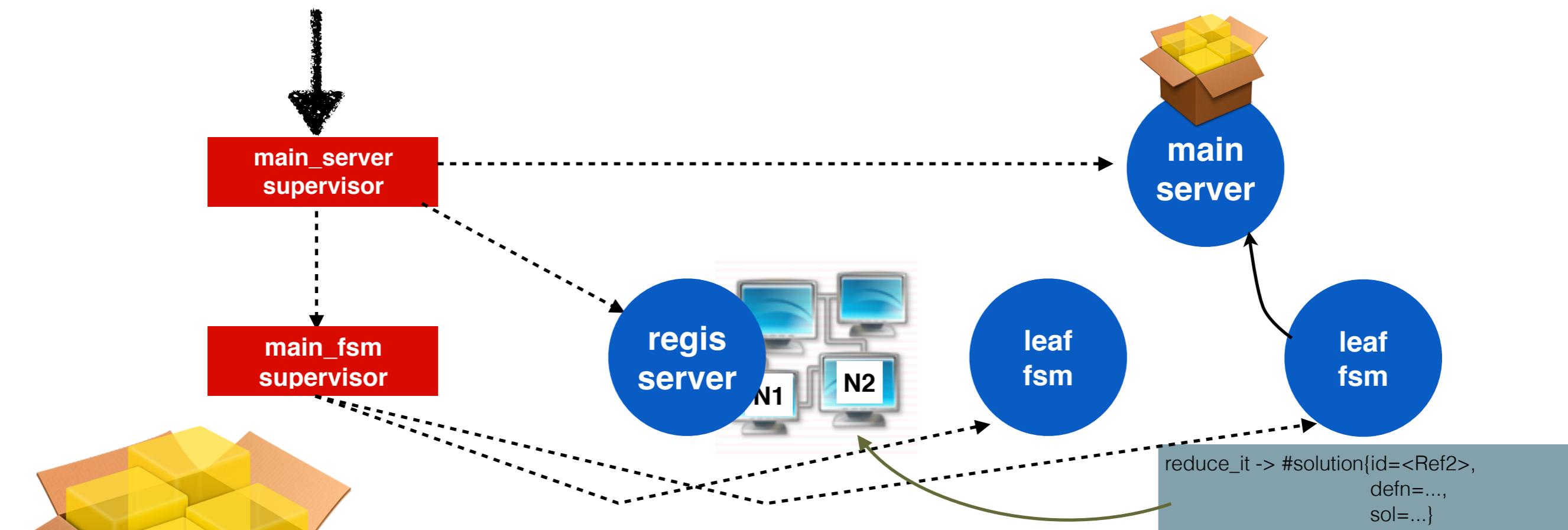


	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem	[]	[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]	[]	[]
<Ref1>	[]	[#solution{gid=<Ref1>, id=<Ref2>, defn=.., sol=..}]	[#solution{gid=<Ref1>, id=<Ref3>, defn=.., sol=..}]	[]
<Ref2>	[]	[]	[#solution{gid=<Ref2>, id=<Ref4>, defn=.., sol=..}, #solution{gid=<Ref2>, id=<Ref5>, defn=.., sol=..}]	[]

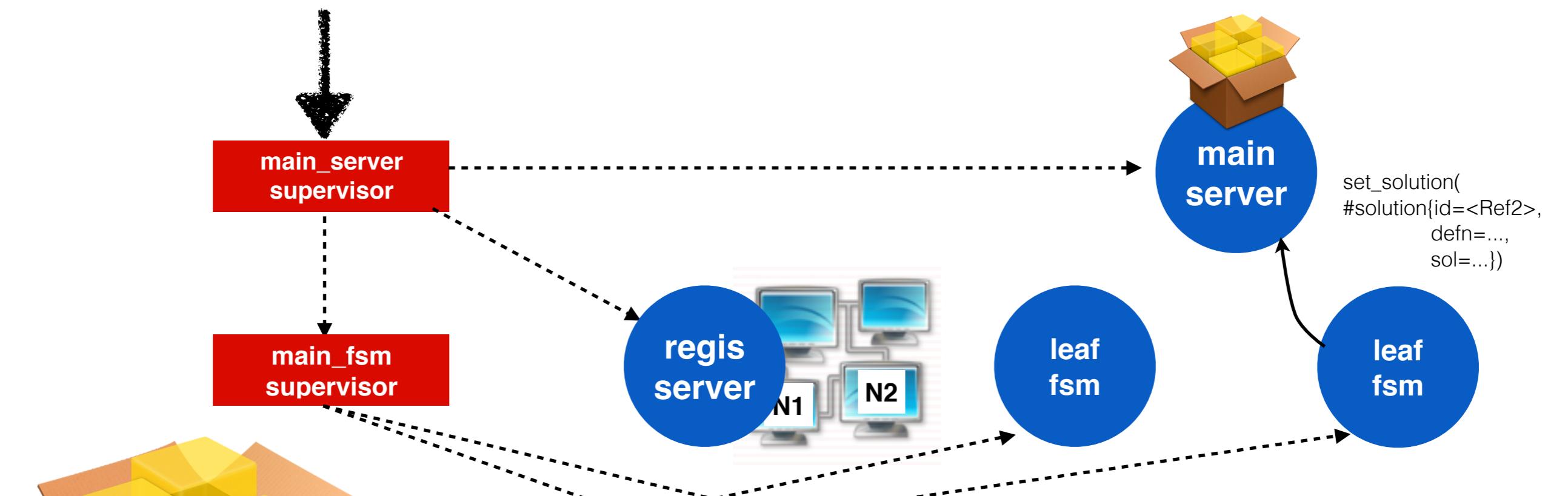
But note that only <Ref2> can be reduced,
since <Ref1> still has problems being solved!!!



	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem		[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]		
<Ref1>	[#solution{gid=<Ref1>, id=<Ref2>, defn=..., sol=...}]	[#solution{gid=<Ref1>, id=<Ref3>, defn=..., sol=...}]		
<Ref2>			[#solution{gid=<Ref2>, id=<Ref4>, defn=..., sol=...}, #solution{gid=<Ref2>, id=<Ref5>, defn=..., sol=...}]	

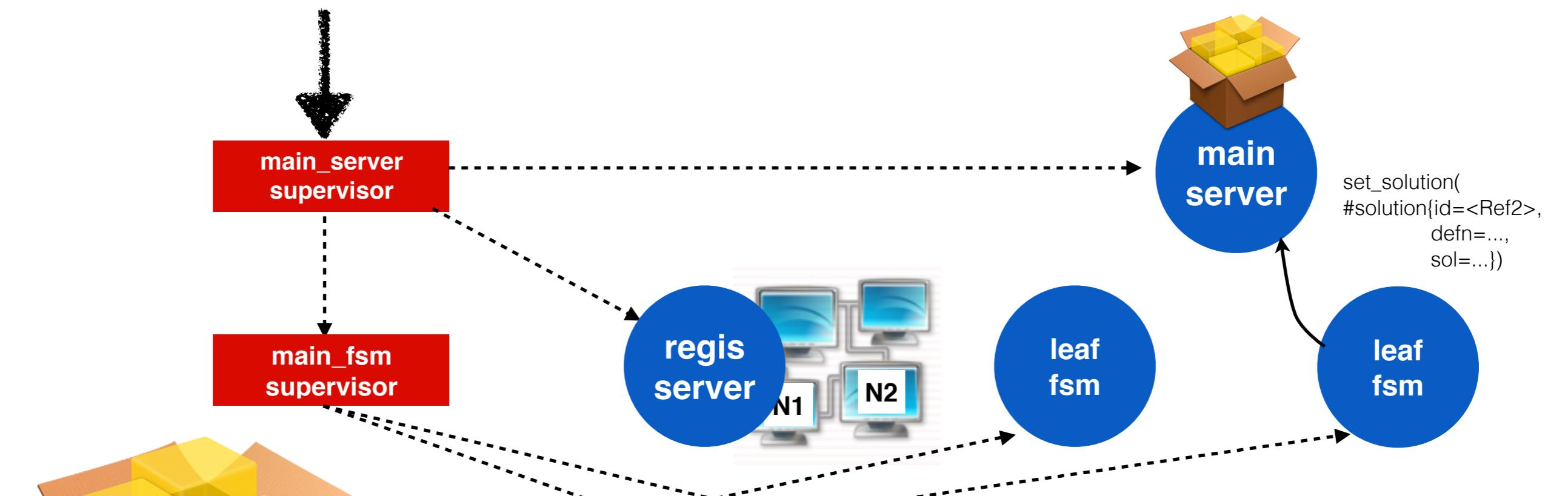


	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem	[]	[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]	[]	[]
<Ref1>	[]	[#solution{gid=<Ref1>, id=<Ref2>, defn=..., sol=...}]	[#solution{gid=<Ref1>, id=<Ref3>, defn=..., sol=...}]	[]
<Ref2>	[]	[]	[]	[#solution{gid=<Ref2>, id=<Ref4>, defn=..., sol=...}, #solution{gid=<Ref2>, id=<Ref5>, defn=..., sol=...}]

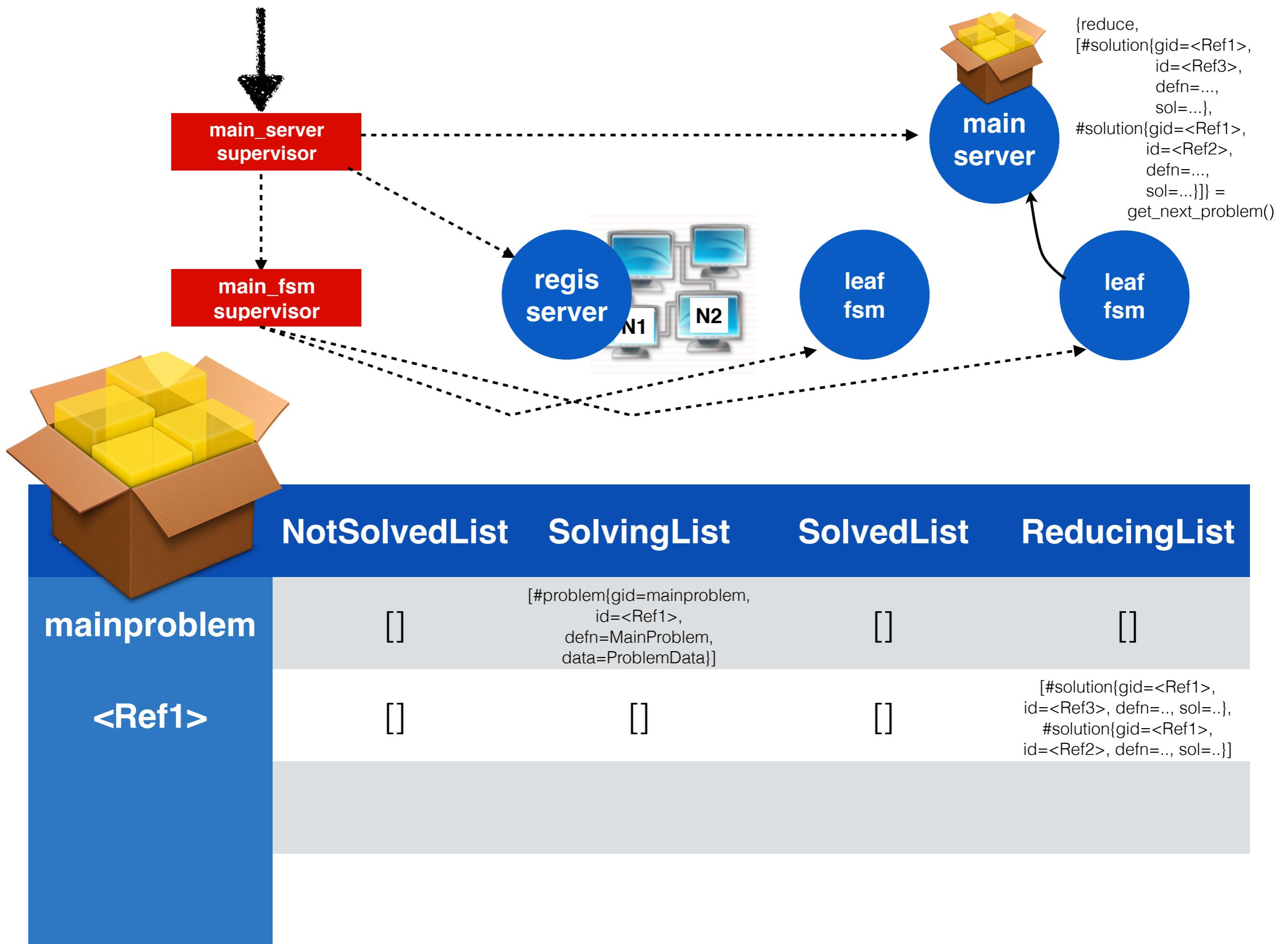


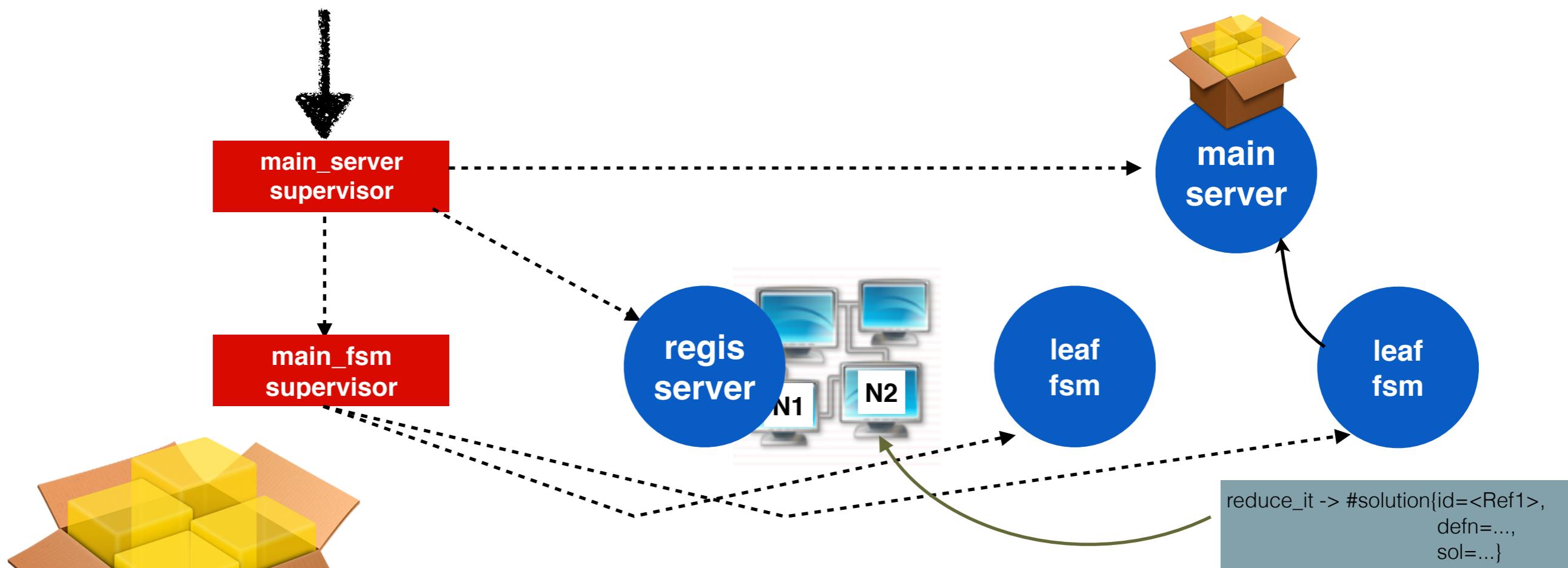
	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem	[]	[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]	[]	[]
<Ref2>	[]	[#solution{gid=<Ref1>, id=<Ref2>, defn=..., sol=...}]	[#solution{gid=<Ref1>, id=<Ref3>, defn=..., sol=...}]	[#solution{gid=<Ref2>, id=<Ref4>, defn=..., sol=...}, #solution{gid=<Ref2>, id=<Ref5>, defn=..., sol=...}]

Note that there's no gid in this reduced solution, which indicates to main_server that it's a reduction!!!

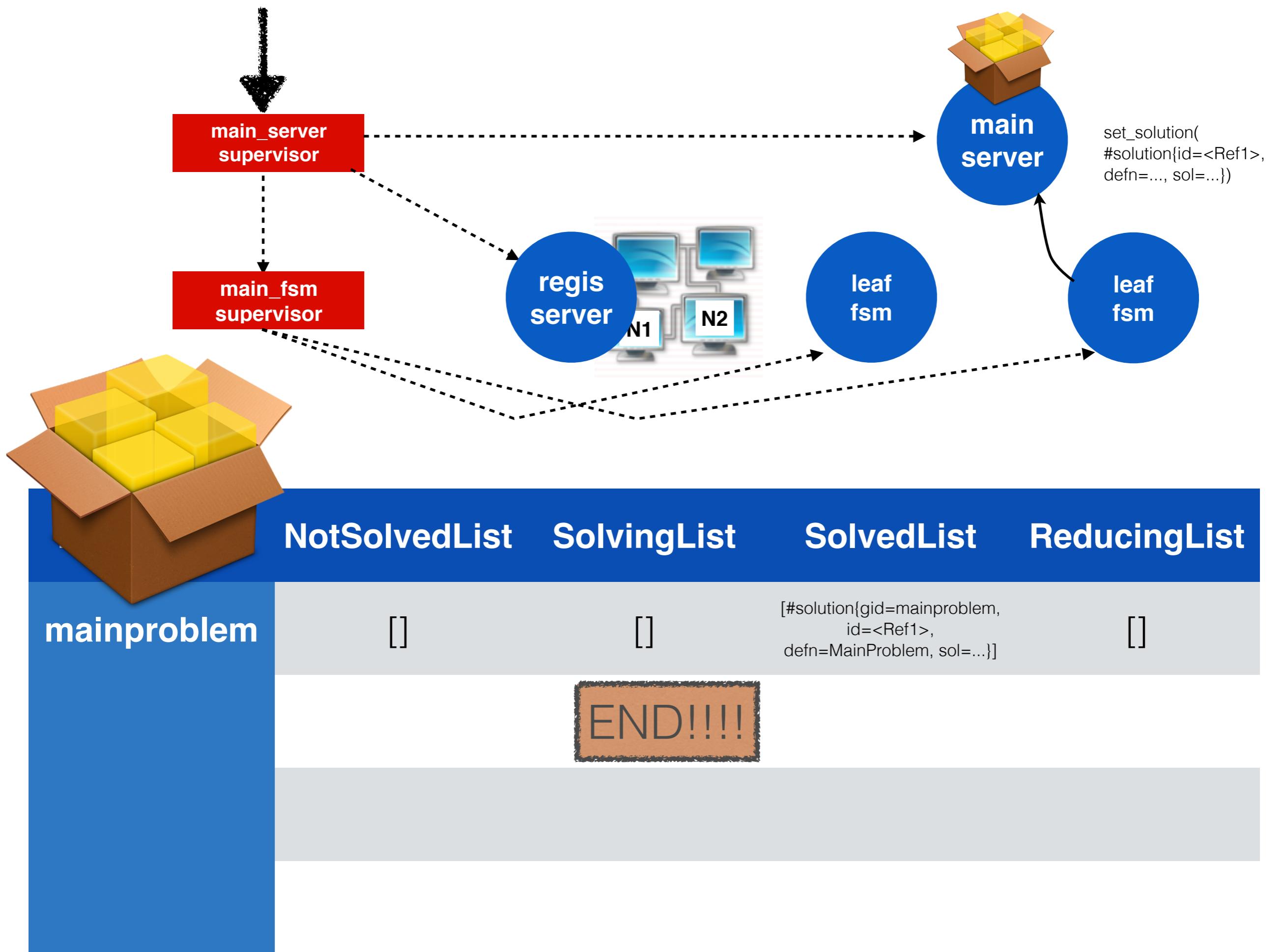


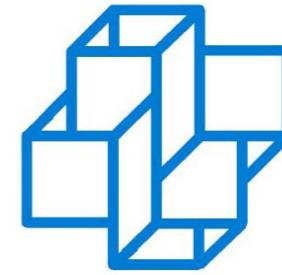
	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem		[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]		
<Ref1>	[]		[#solution{gid=<Ref1>, id=<Ref3>, defn=.., sol=..}, #solution{gid=<Ref1>, id=<Ref2>, defn=.., sol=..}]	[]





	NotSolvedList	SolvingList	SolvedList	ReducingList
mainproblem	[#problem{gid=mainproblem, id=<Ref1>, defn=MainProblem, data=ProblemData}]	[]	[]	[]
<Ref1>	[]	[]	[]	[#solution{gid=<Ref1>, id=<Ref3>, defn=.., sol=..}, #solution{gid=<Ref1>, id=<Ref2>, defn=.., sol=..}]





Preliminary tests

- Testing problem – Darcy (fluid flow through a porous medium)

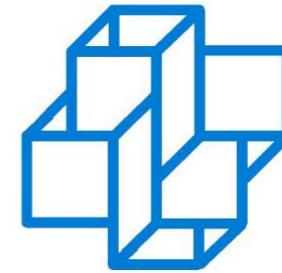
Find $p = p(\mathbf{x})$ such that

$$\begin{cases} \operatorname{div}(-K\nabla p) = f & \text{in } \Omega \\ p = 0 & \text{in } \Gamma \end{cases}$$

K is a positive definite tensor

$K(\mathbf{x})$ and $f(\mathbf{x})$ **may** have multi-scale characteristics

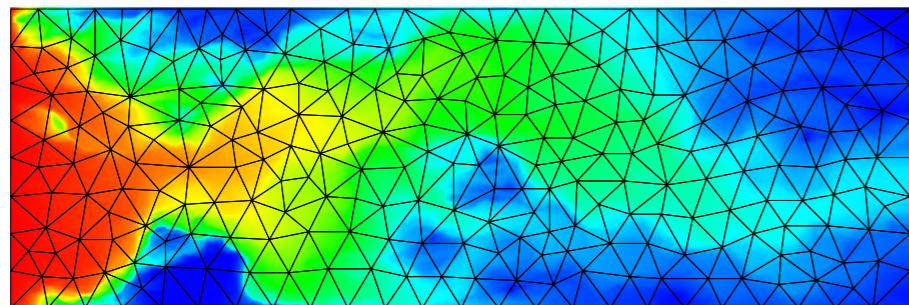
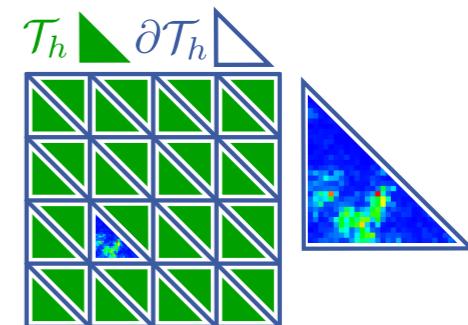
Preliminary tests



- In MHM (2D):

Find $(p_0, \lambda_h) \in V_0 \times \Lambda_h$ **such that**

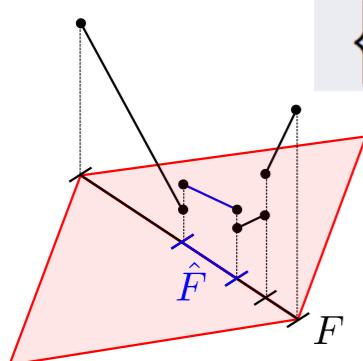
$$\begin{aligned} (\lambda_h, q_0)_{\partial\mathcal{T}_h} &= (f, q_0)_{\mathcal{T}_h} \\ (p_0, \mu_h)_{\partial\mathcal{T}_h} + (p^{\lambda_h}, \mu_h)_{\partial\mathcal{T}_h} &= -(p^f, \mu_h)_{\partial\mathcal{T}_h} \end{aligned}$$



$$p^{\lambda_h} = \sum_{i=1}^{N_h} c_i \eta_i$$

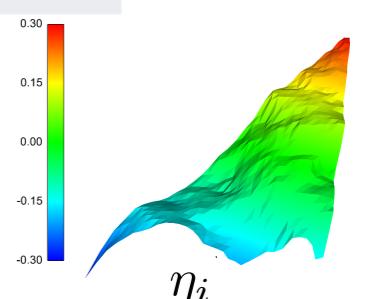
Find $\eta_i \in V_0^\perp(K)$ **such that**

$$\begin{cases} \mathcal{L}\eta_i &= (\mathbf{n} \cdot \mathbf{n}^K) c^{\psi_i} \\ -\mathcal{K}\nabla\eta_i \cdot \mathbf{n}^K &= (\mathbf{n} \cdot \mathbf{n}^K)\psi_i \end{cases}$$



Find $p^f \in V_0^\perp(K)$ **such that**

$$\begin{cases} \mathcal{L}p^f &= f^\perp \\ -\mathcal{K}\nabla p^f \cdot \mathbf{n}^K &= 0 \end{cases}$$



Total number of integration points:

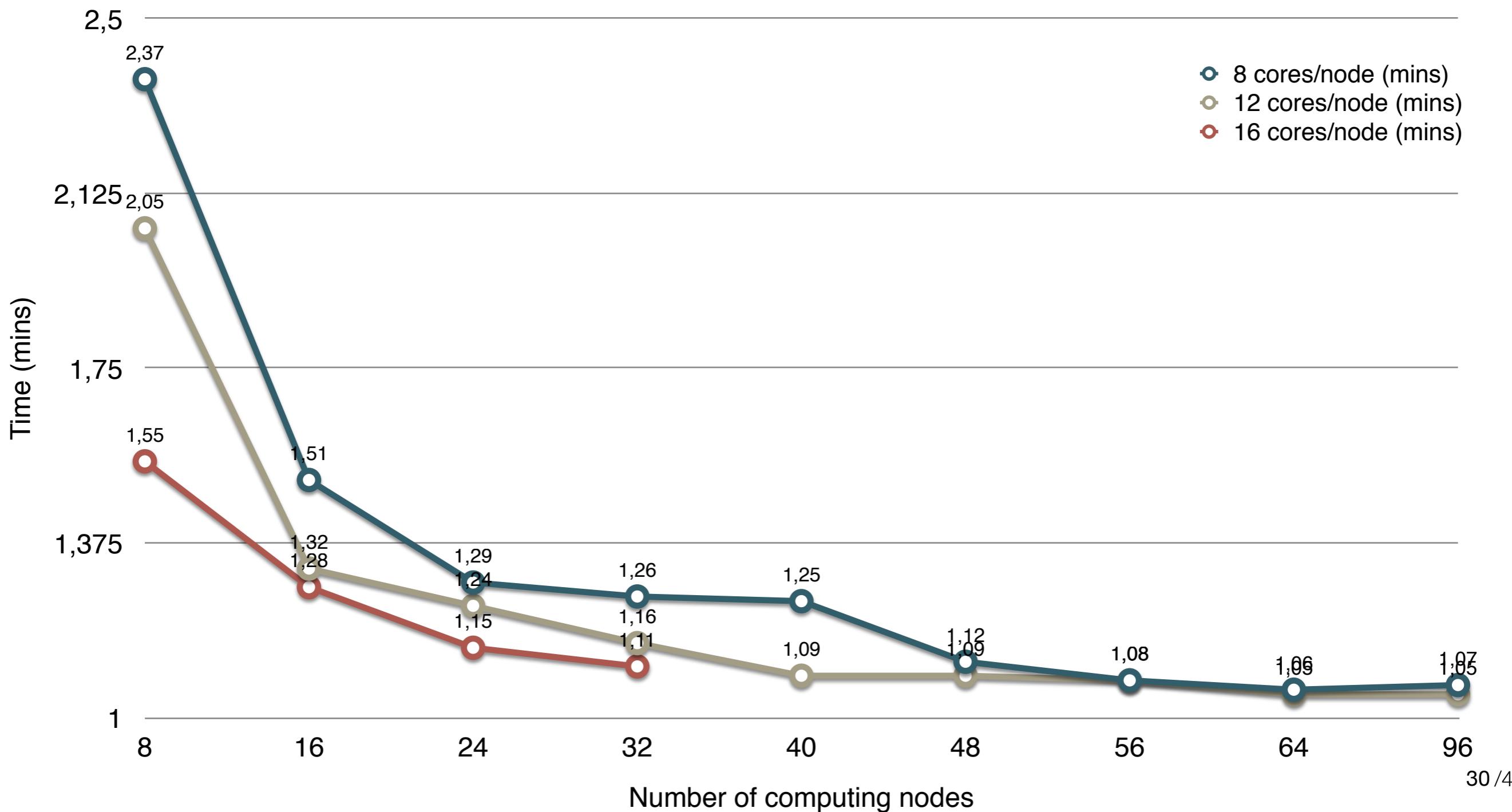
$$G * 3 * L + G * 2 * 3 * \sqrt{L}$$

within 1st level elements

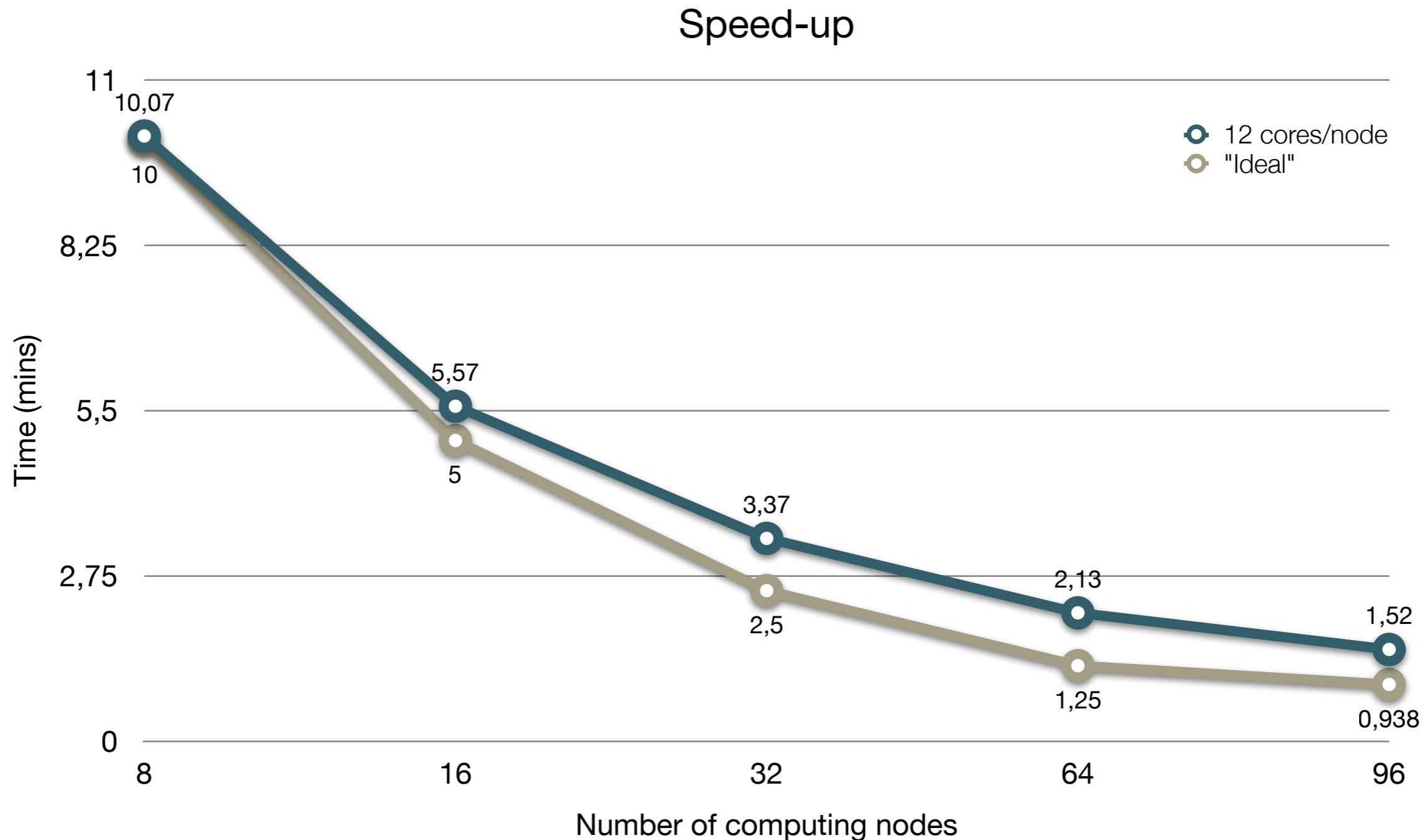
on 1st level faces

4,194,304 (G=1,024 x L=4,096) elements

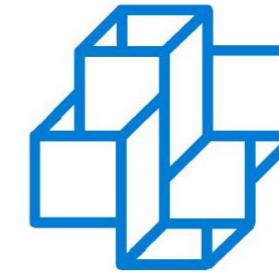
Speed-up



16,777,216 (G=1,024 x L=16,384) elements

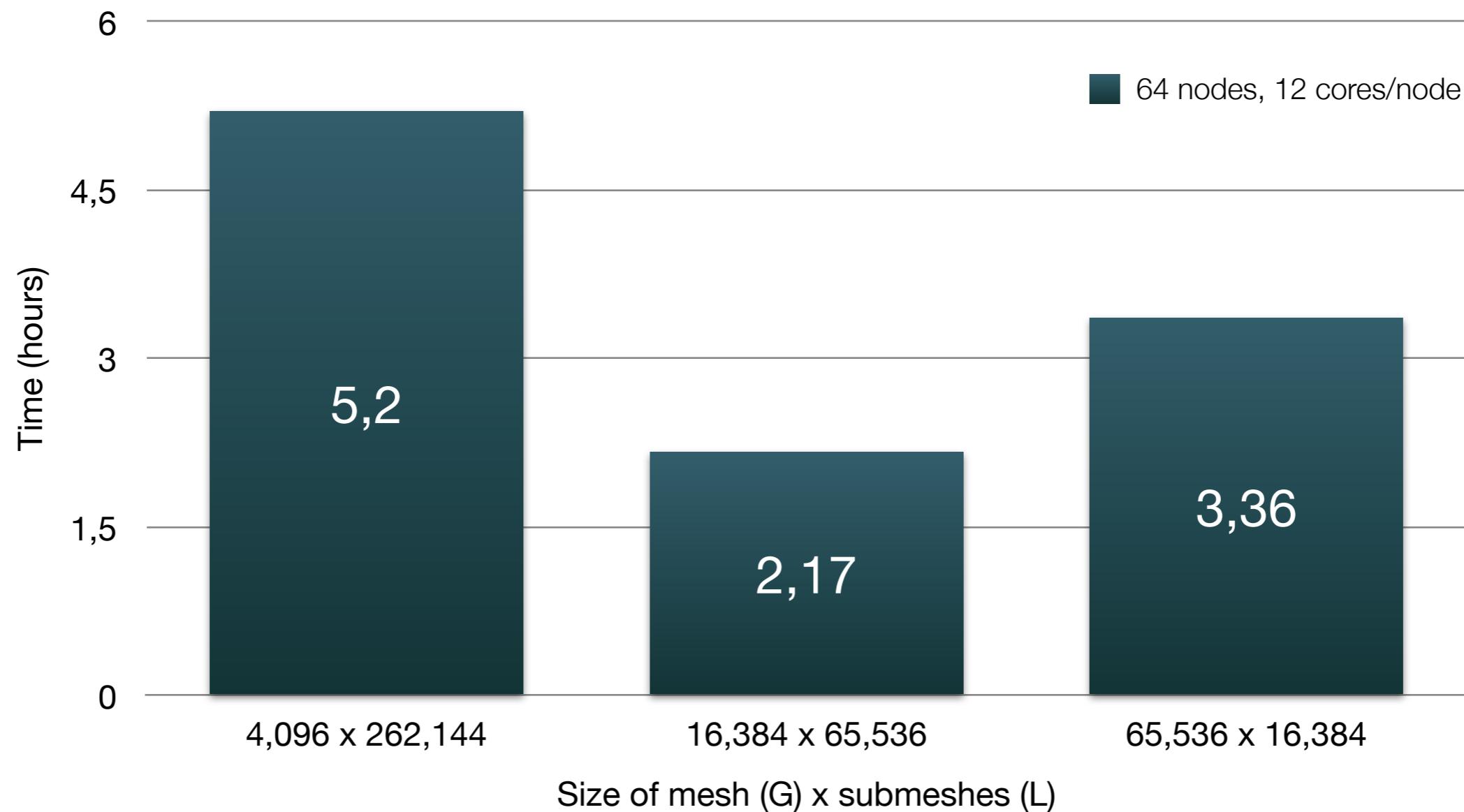


1,073,741,824 elements

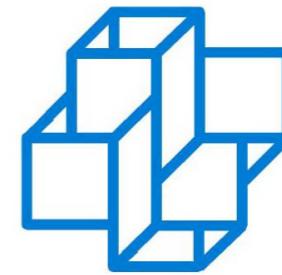


Laboratório
Nacional de
Computação
Científica

Influence of mesh-submesh relation

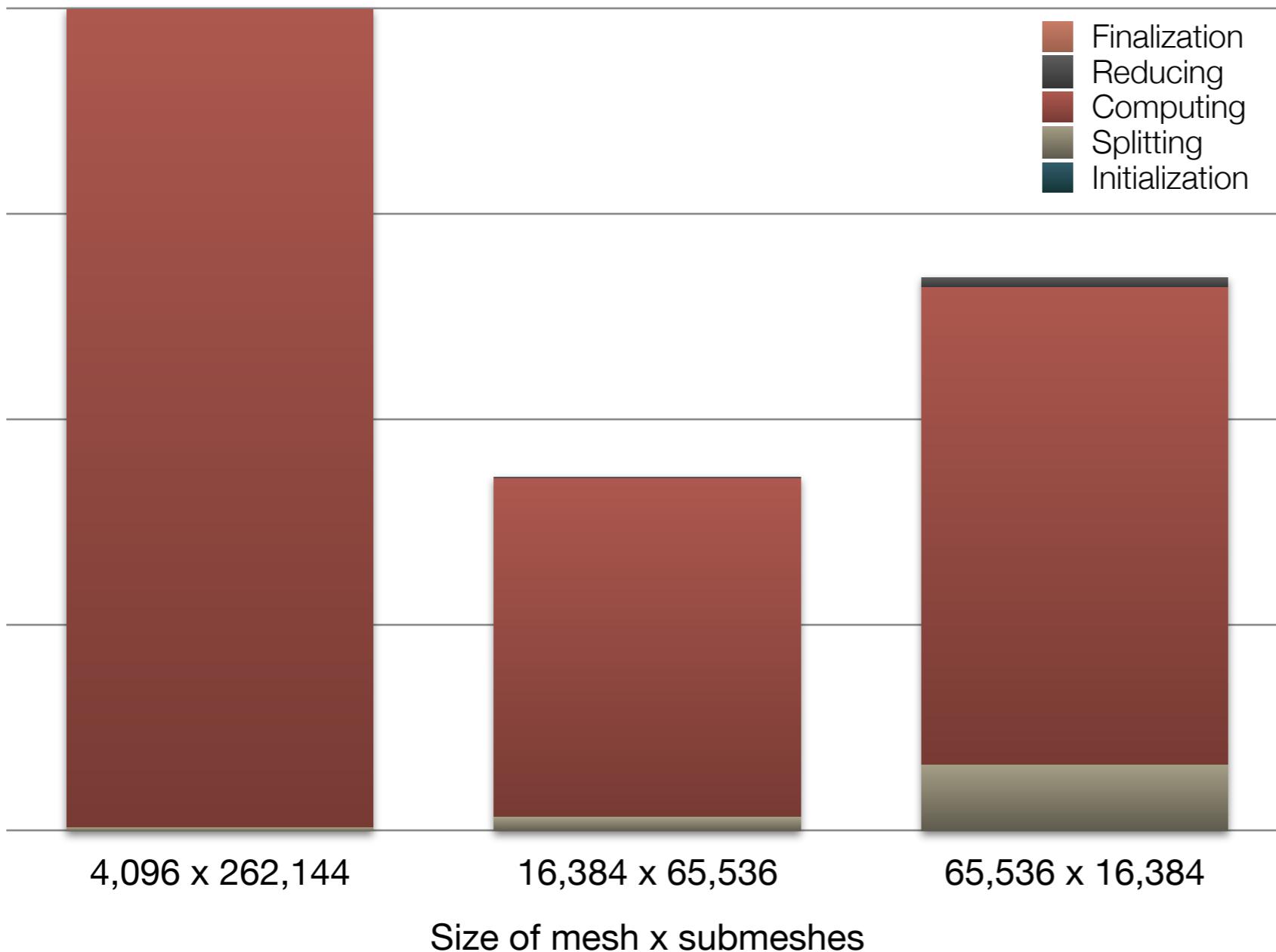


1,073,741,824 elements

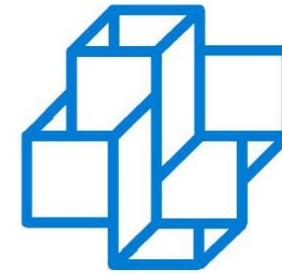


Laboratório
Nacional de
Computação
Científica

Cost share of each phase (64 nodes, 12 cores/node)

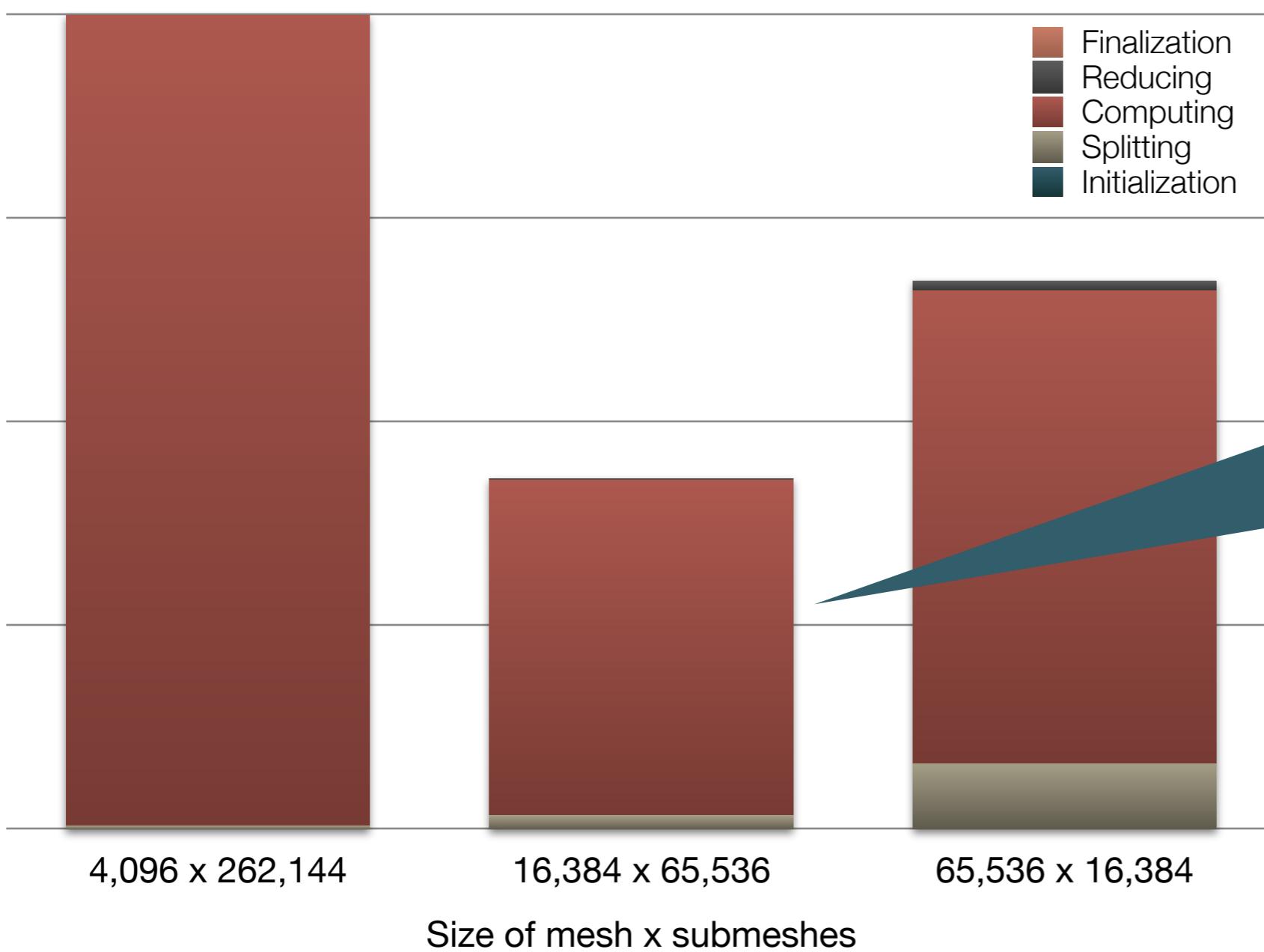


1,073,741,824 elements



Laboratório
Nacional de
Computação
Científica

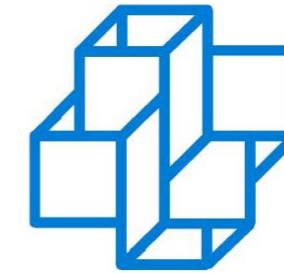
Cost share of each phase (64 nodes, 12 cores/node)



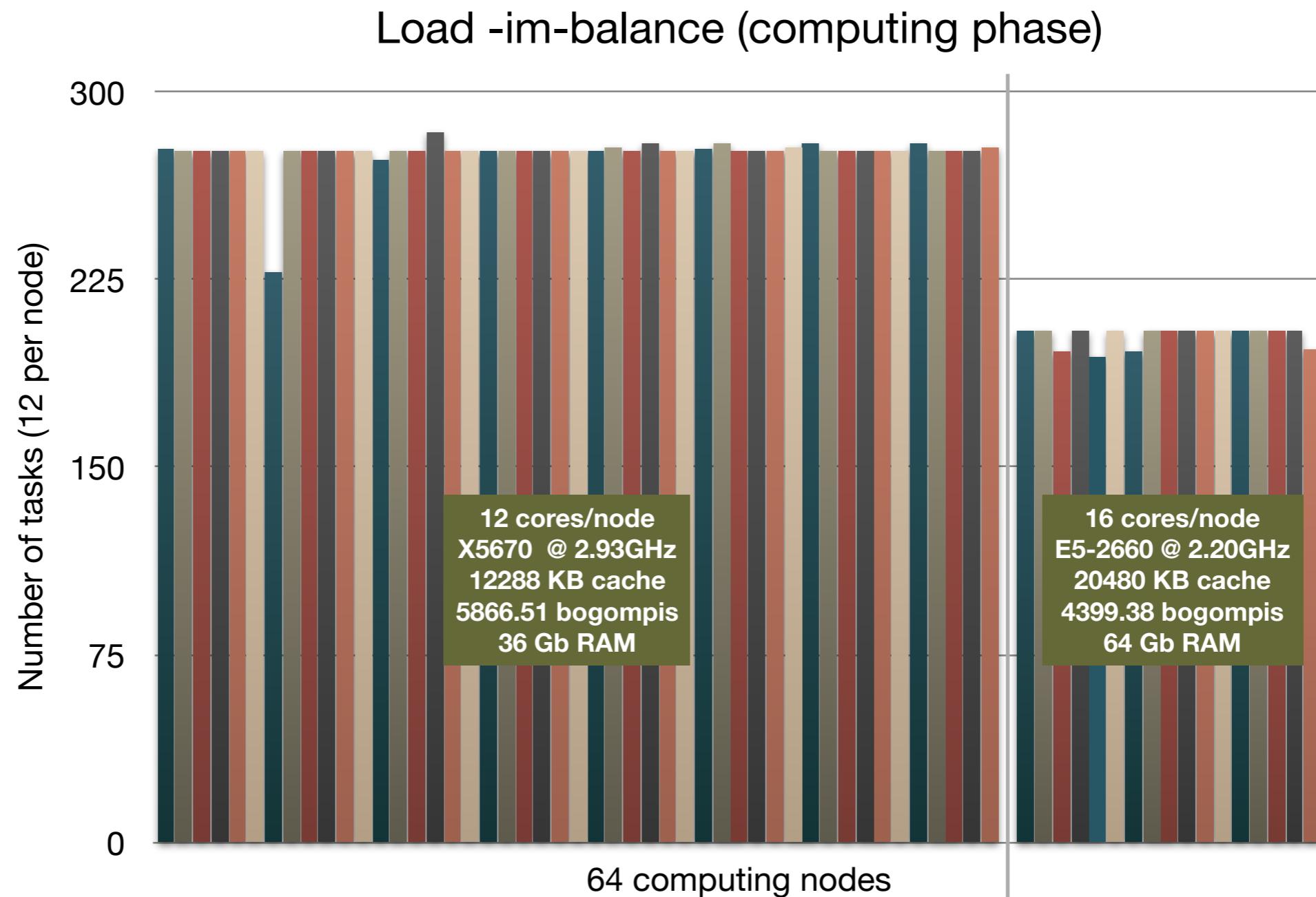
Total time:
2,17h

Erlang overhead:
< 8min

1,073,741,824 elements

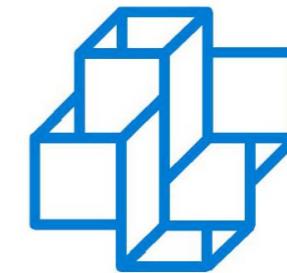


Laboratório
Nacional de
Computação
Científica

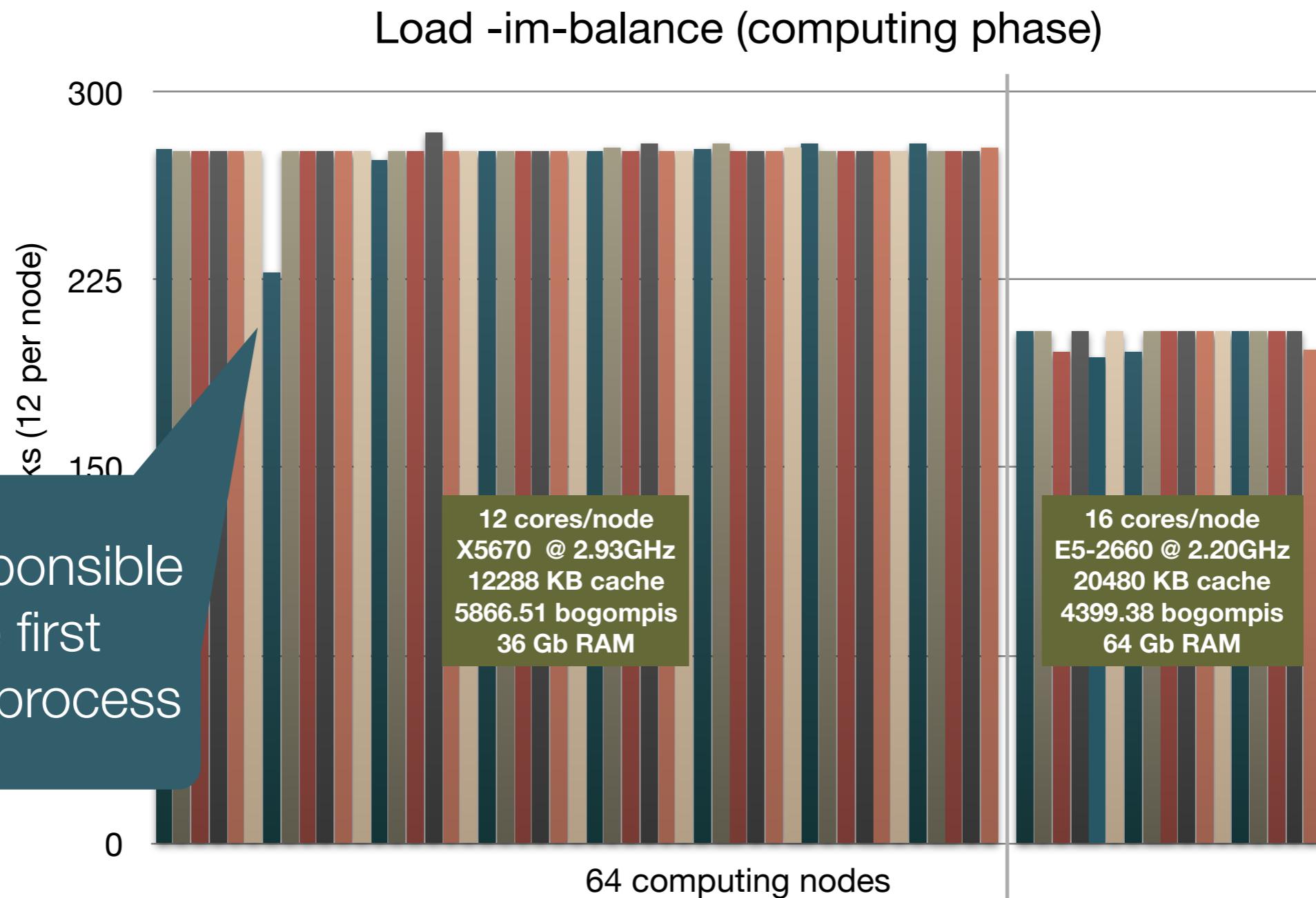


macchpc[101,104-107,109-113,123-131,133-136,159-162,164-168,173-187,219-223,225-236]

1,073,741,824 elements

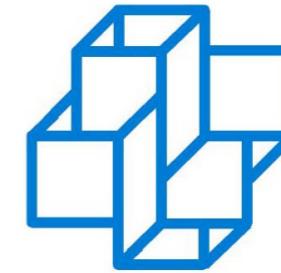


Laboratório
Nacional de
Computação
Científica

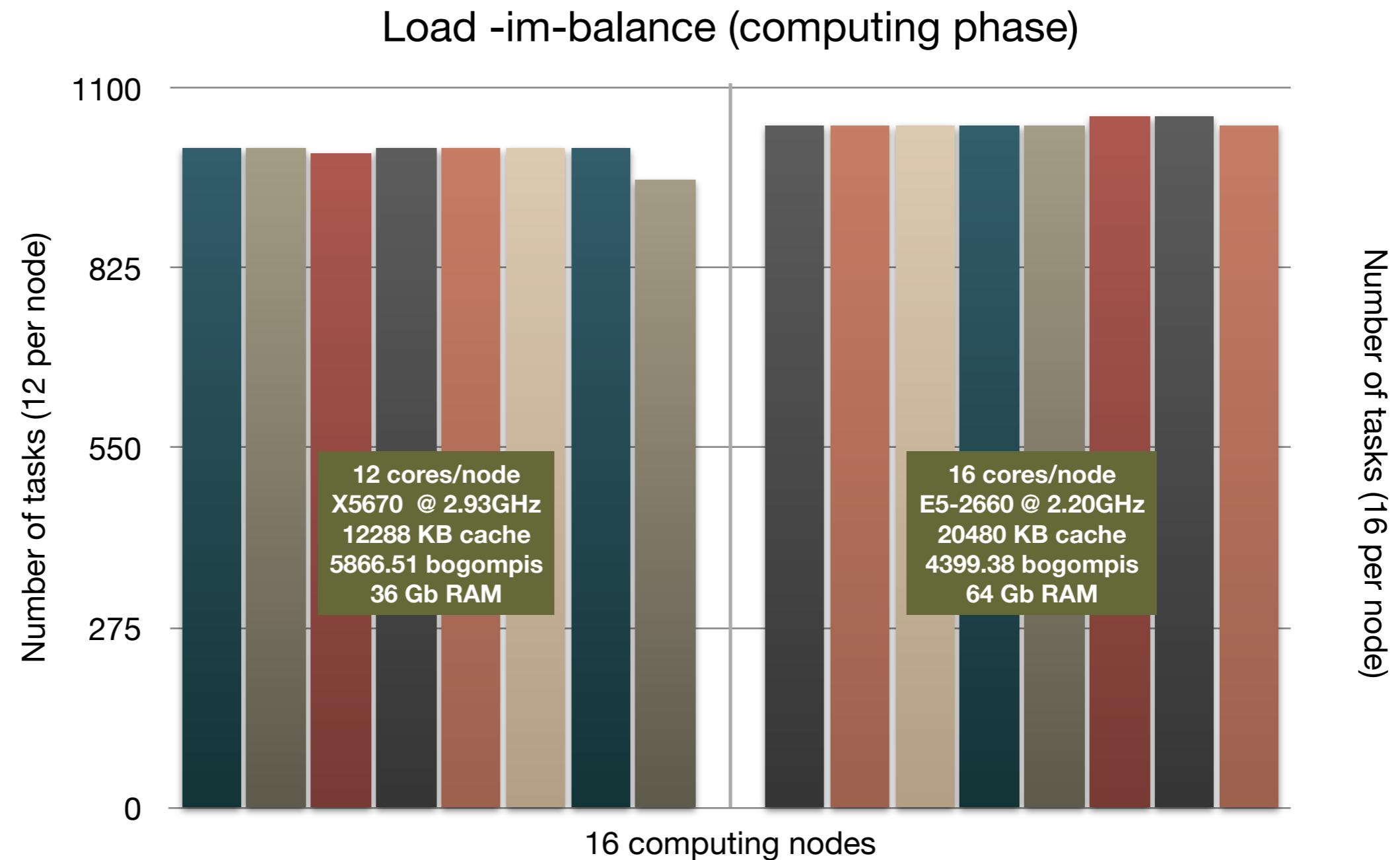


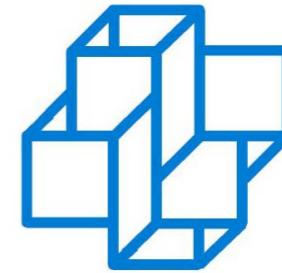
macchpc[101,104-107,109-113,123-131,133-136,159-162,164-168,173-187,219-223,225-236]

1,073,741,824 elements



Laboratório
Nacional de
Computação
Científica





Fault tolerance: let it crash!

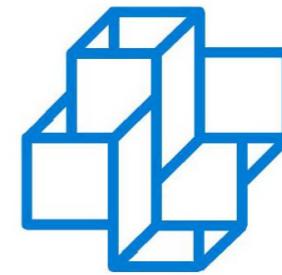
- At the worker nodes:
 - Network partitions and node failures are considered permanent
 - Related “being solved”/ “being reduced” task is rolled back to the “not solved list”/ “solved list”
- At the master node (main_server):
 - Network partitions and node failures are considered transient
 - All tasks in “being solved list”/ “being reduced” are rolled back to the “not solved list”/ “solved list”
 - Backup master node does the failover

**Fault Tolerance in
Distributed Systems**

- Perfect world: No Failures
 - We don't live in a perfect world
- Non-distributed system
 - Crash, you're dead*
- Distributed system: Redundancy
 - Should result in less down time
 - But does it?
 - Distributed systems according to Butler Lampson

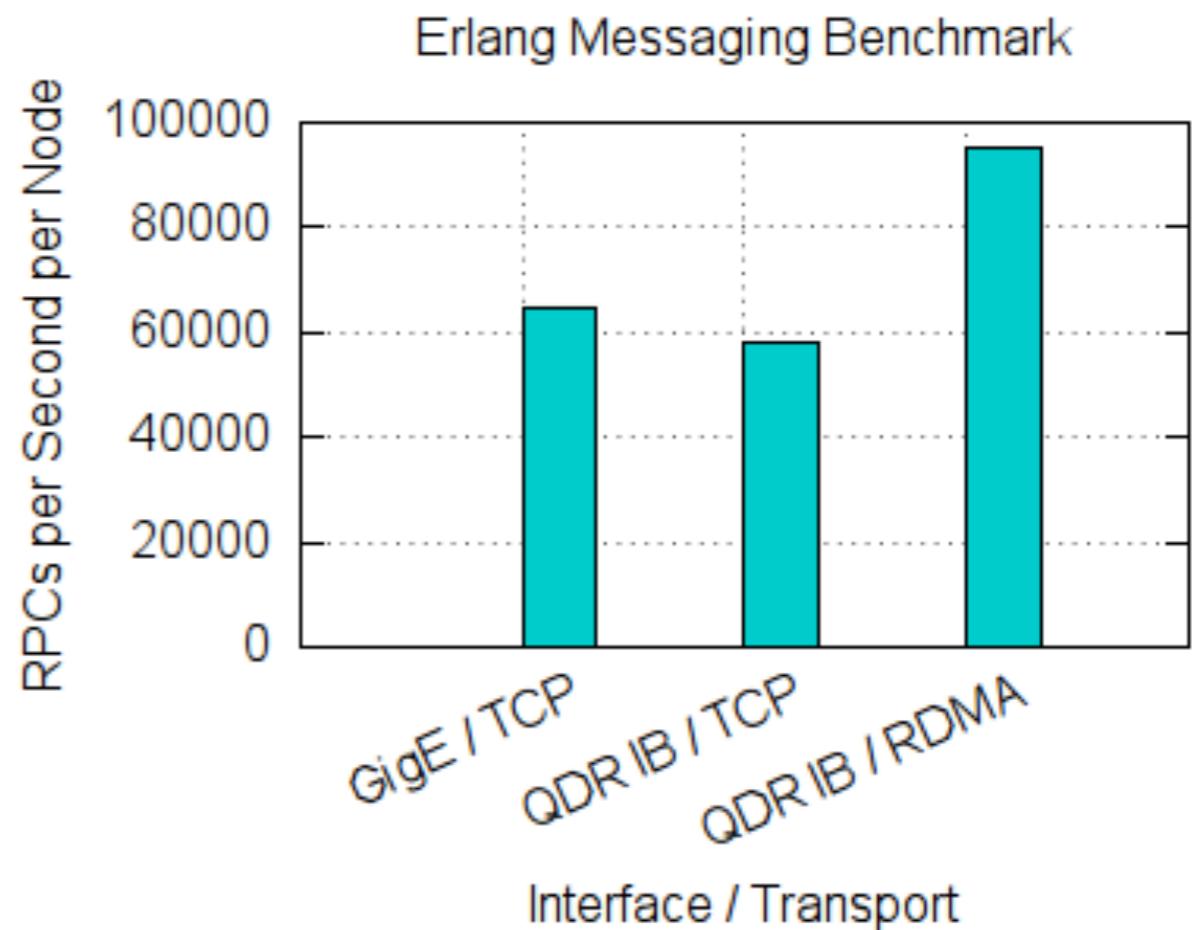
A distributed system is a system in which I can't get my work done because a computer that I've never heard of has failed.

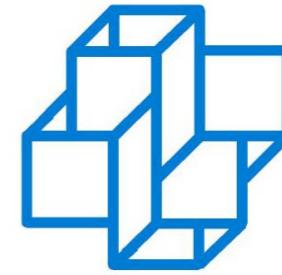
Source: <https://www.cs.purdue.edu/homes/clifton/cs603/FailureModel.ppt>



To-do... (Erlang part)

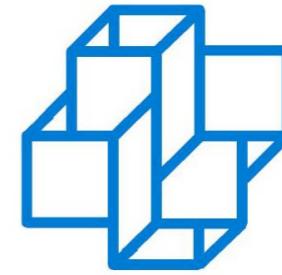
- Employ native RDMA protocol over Infiniband (https://github.com/MrStaticVoid/rdma_dist)
- Adopted message formats are too memory greedy (and unnecessarily wordy)
 - Related with slide 36...
- Do better cleanup of worker nodes in case of network partitioning





To-do... (C++ part)

- Unexpected results for $l > 0$ (degree of lagrange multiplier polynomials)
- Unexpected results for $k > 5$ (degree of polynomials at the subelement level)
- Basic optimizations at the processing phase
- Hotspots at the splitting (I/O intensive) and reducing (computing/memory intensive) phases
- Develop auto-mesher for 3D problems



To-do... (Erlang*C++ part)

- Give support to transient problems
- Make overall configuration more flexible (a lá SPiNMe)
- Provide better (if any!) visualization output
- Consider other execution environments (e.g. clouds)

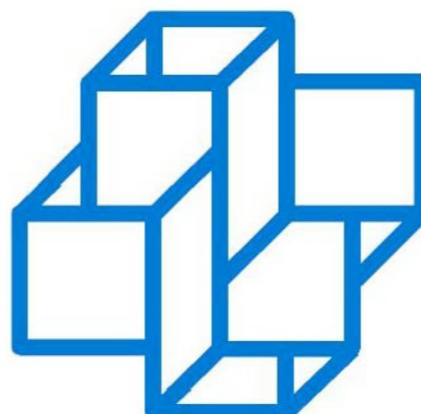
Thank you!!!

Merci!!!



Antônio Tadeu A. Gomes
Diego Paredes
Frédéric Valentin

atagomes@lncc.br



**Laboratório
Nacional de
Computação
Científica**

