The MathWorks

MATLAB® & SIMULINK®

MATLAB Based Optimization Techniques and Parallel Computing

Bratislava
June 4, 2009

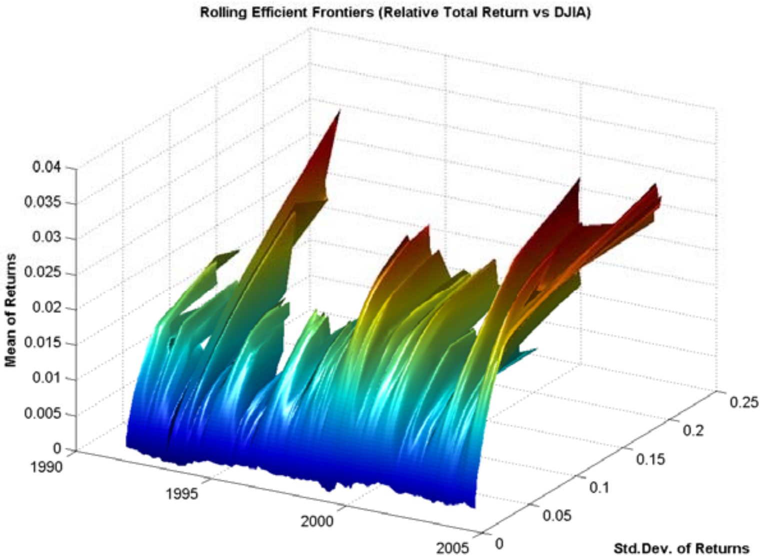
Jörg-M. Sautter

Application Engineer

The MathWorks

Agenda

- Introduction
- Local and Smooth Optimization
- Example:
  - Portfolio Optimization, part 1
  - Expected Shortfall
  - GARCH
- Global or Non-Smooth Optimization
- Example:
  - Portfolio Optimization, part 2
- Parallel Computing
- Summary



Optimization workflow

Major steps

- Define your problem
- Solve your problem $x = \arg\min_{x \in \mathbb{R}^n} f(x)$, possibly subject to constraints
- Analyze and visualize the result

Demands

- Appropriate solver
- Fast solver
- Robust solver
- Customizable solver settings
- Easy to use
- …

What you have to do

Situation

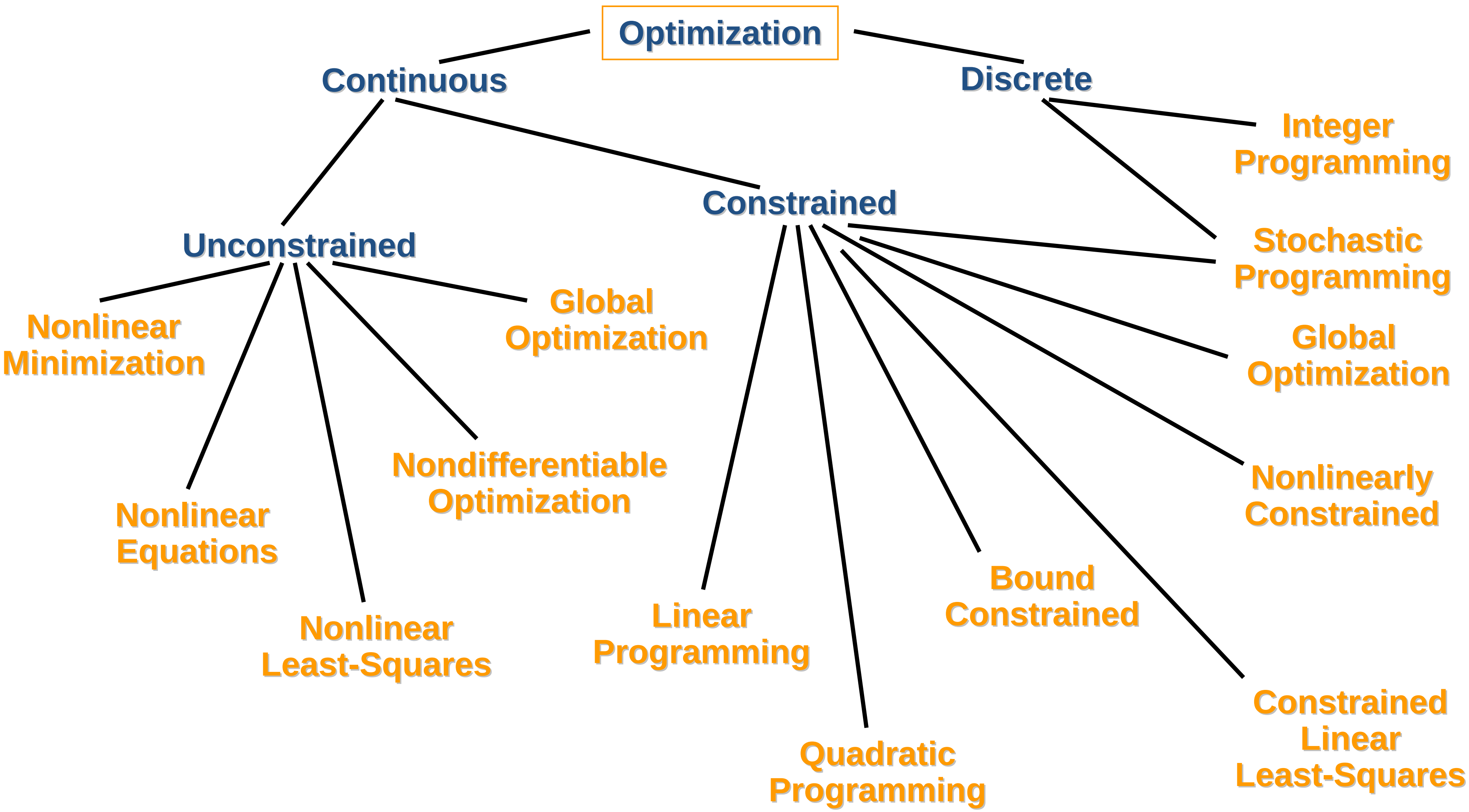
- You know your objective function
- You know the constraints (if there are any)

What you have to do

- Code your objective function (in MATLAB, C, C++, Excel,…)
- Code your constraints
- Determine the type of optimization problem
- Call the appropriate solver

Note: there are exceptions which are even easier to handle…

Classification



MATLAB Based Optimization Tools

- MATLAB
- Curve Fitting Toolbox
- Optimization Toolbox
- Genetic Algorithms and Direct Search Toolbox

as well as

- Statistics Toolbox
- Symbolic Math Toolbox

Optimization with MATLAB

- Direct methods for linear systems
  - square systems
    - LU-decomposition
    - Cholesky-decomposition
    - QR-decomposition
  - underdetermined systems
  - overdetermined systems

- Iterative methods for linear systems
  - cg-method (and variants)
  - GMRES, QMR, MINRES,…

- Polynomial approximation

- …

Agenda

Introduction

Local and Smooth Optimization

Example:

- Portfolio Optimization, part 1
- Expected Shortfall
- GARCH

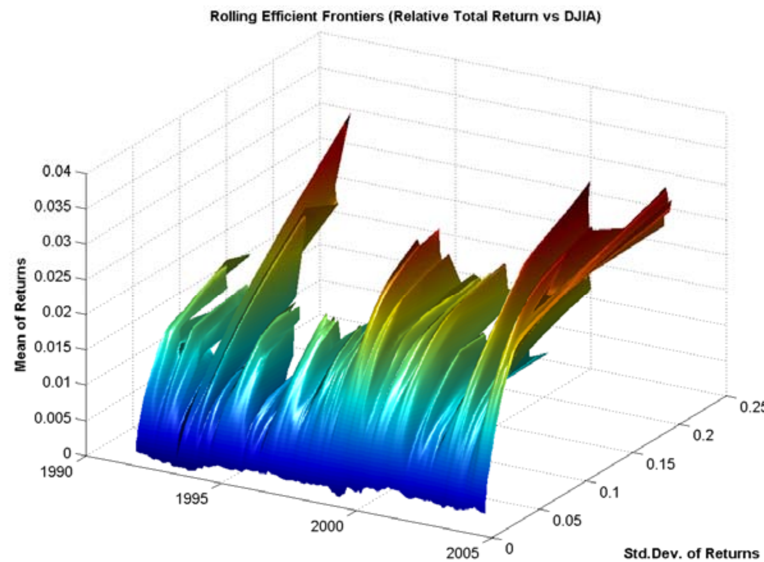
Global or Non-Smooth Optimization

Example:

- Portfolio Optimization, part 2

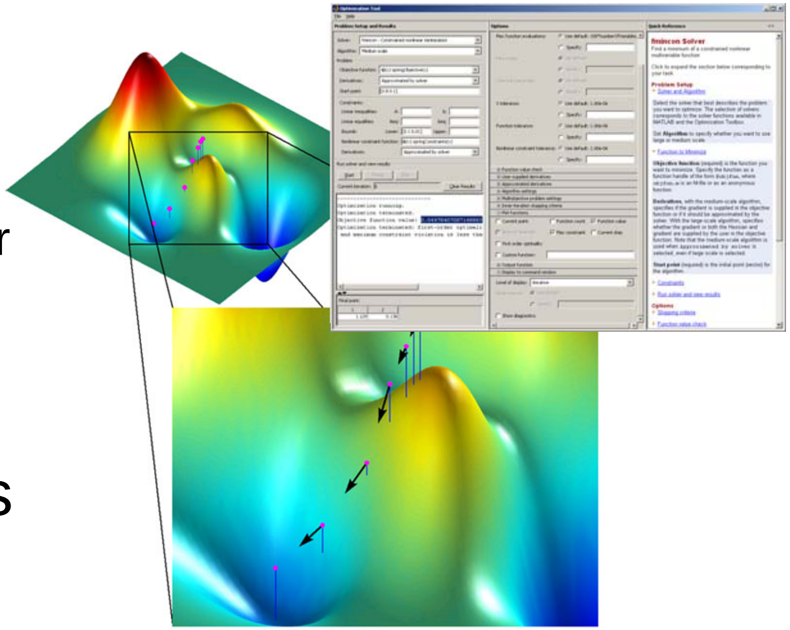
Parallel Computing

Summary



Optimization Toolbox

- Graphical user interface and command line functions for:
  - Linear and nonlinear programming
  - Quadratic programming
  - Nonlinear least squares and nonlinear equations
  - Multi-objective optimization
  - Binary integer programming
- Customizable algorithm options
- Standard and large-scale algorithms
- Output diagnostics



Example:
Minimum of a smooth function

Agenda

Introduction

Local and Smooth Optimization

Example:

- Portfolio Optimization, part 1
- Expected Shortfall
- GARCH

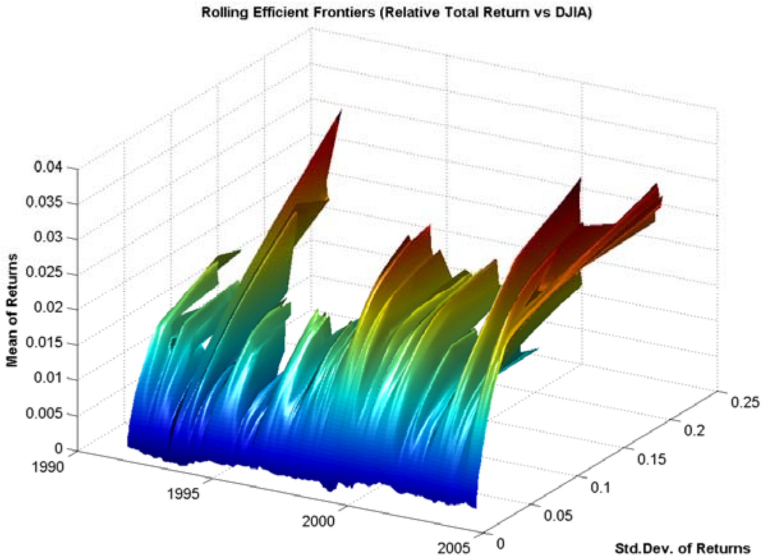
Global or Non-Smooth Optimization

Example:

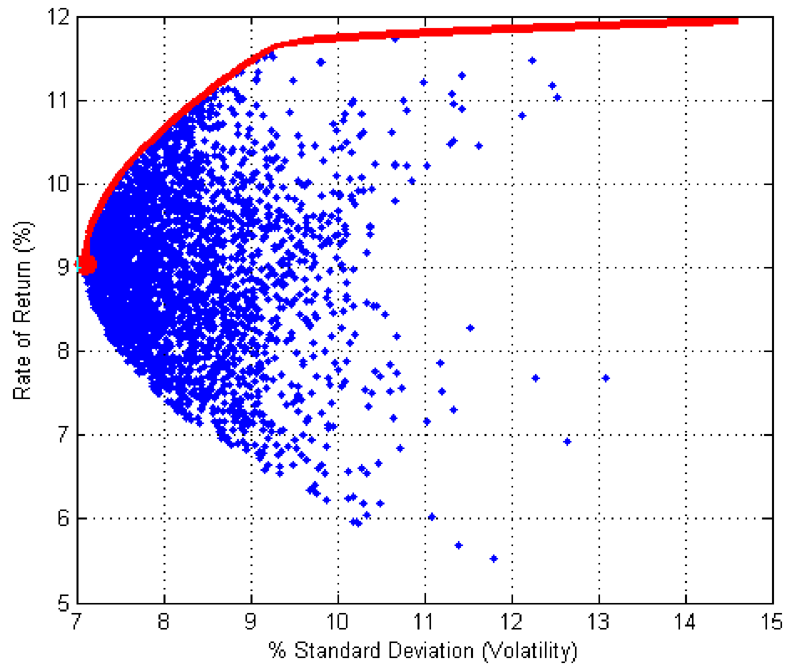
- Portfolio Optimization, part 2

Parallel Computing

Summary



Example:
Portfolio Optimization



Classic Mean-Variance

A portfolio is MV optimal if it has least risk for a given level of returns:

minimize 5 @ z W F z   z 5 U q >F 5 U q § q

subject to ẽ W @ z W ẽ   ẽ 5 U q

[

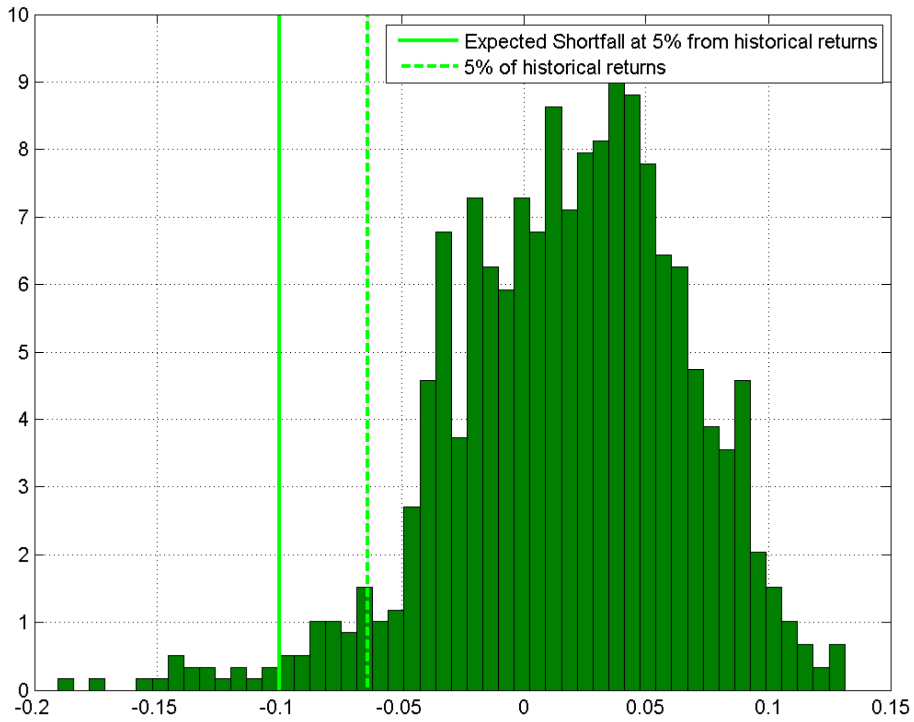
4 @ z 1

1

and possibly 3 z 1 4

Du d> Eu@ e===

Example:
Expected Shortfall



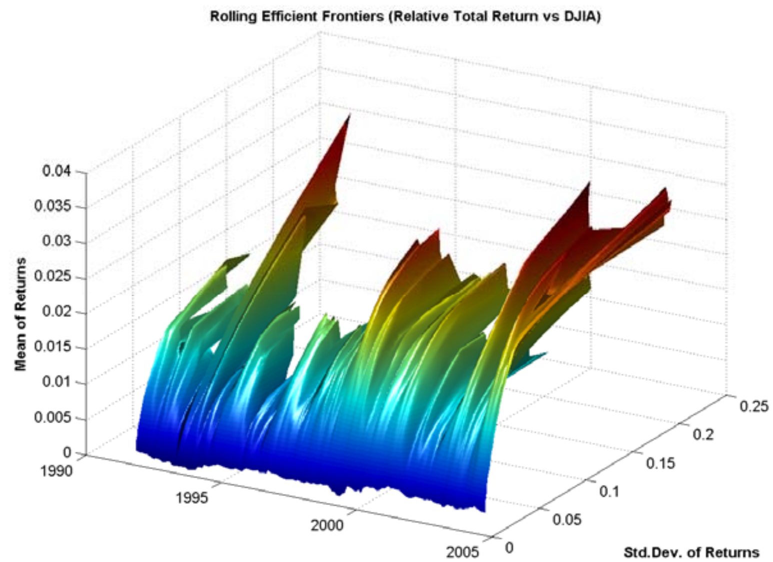
Agenda

Introduction
Local and Smooth Optimization
Example:
- Portfolio Optimization, part 1
- Expected Shortfall
- GARCH

Global or Non-Smooth Optimization

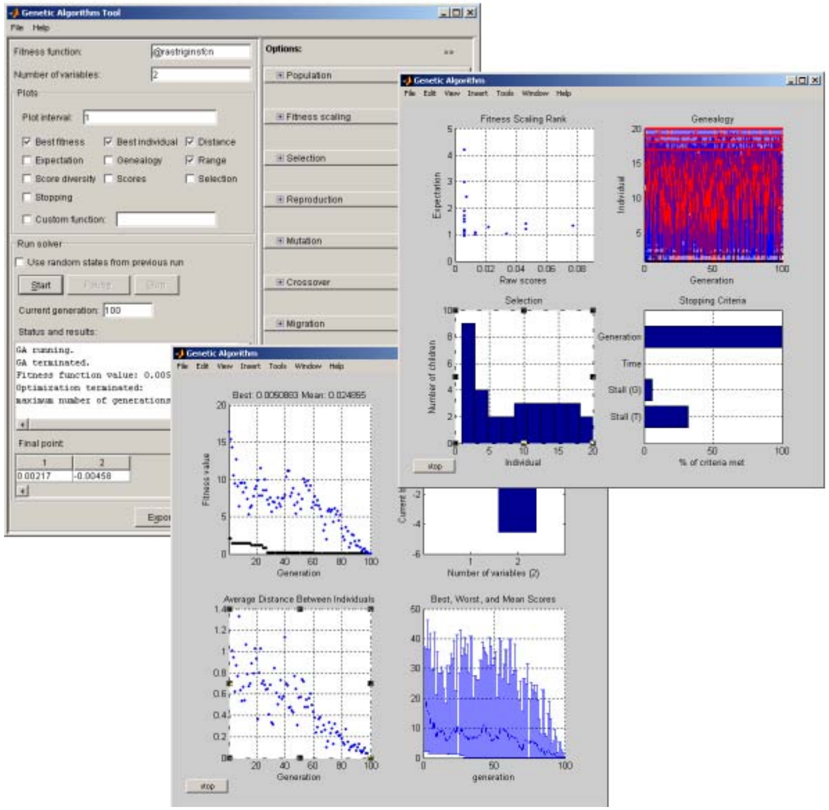
Example:
- Portfolio Optimization, part 2

Parallel Computing
Summary



Genetic Algorithm and Direct Search Toolbox

- Graphical user interface and command line functions for:
  - Genetic algorithm solver
  - Direct search solver
  - Simulated annealing solver
- Useful for problems not easily addressed with Optimization Toolbox:
  - Discontinuous
  - Highly nonlinear
  - Stochastic
  - Discrete or custom data types



What is a Genetic Algorithm?

- Genetic Algorithms use concepts from *evolutionary biology* to find exact or approximate solutions to optimization problems
- Start with an initial generation of candidate solutions that are tested against the objective function
- Subsequent generations evolve from the 1st through *selection*, *crossover* and *mutation*
- The *individual* that best minimizes the given objective is returned as the ideal solution

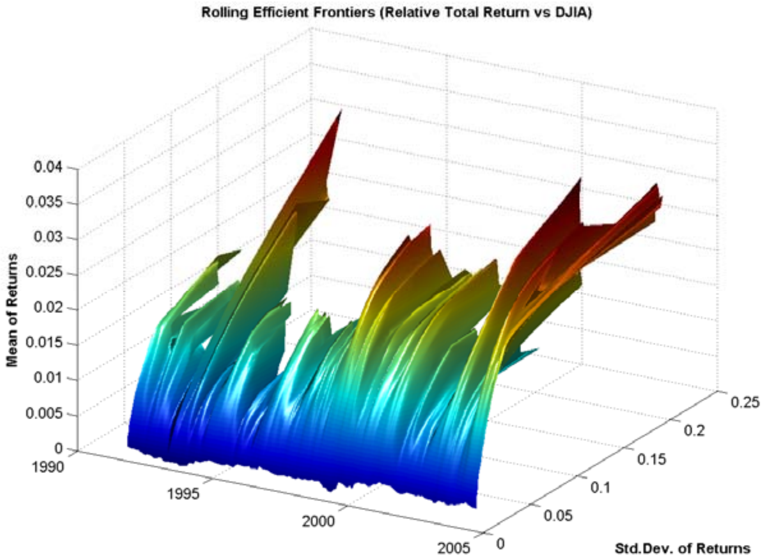
Example:
Minimum of a stochastic function

Agenda

Introduction
Local and Smooth Optimization
Example:
- Portfolio Optimization, part 1
- Expected Shortfall
- GARCH

Global or Non-Smooth Optimization
Example:
- Portfolio Optimization, part 2

Parallel Computing
Summary



Example: Custom Evolution Algorithms

Ticker	Return (Annual)	Std Dev (Annual)	Portfolio Weight	
AA	-1.04%	26.63%	0.00%	0
AIG	2.98%	13.27%	0.00%	0
AXP	14.11%	15.51%	0.00%	0
BA	22.14%	22.62%	0.00%	0
C	15.29%	14.27%	0.00%	0
CAT	3.41%	28.65%	0.00%	0
DD	14.47%	16.22%	0.00%	0
DIS	33.04%	19.02%	9.30%	1
GE	7.26%	12.43%	0.00%	0
GM	43.72%	41.03%	0.00%	0
HD	-2.88%	20.13%	0.00%	0
HON	19.28%	18.77%	0.00%	0
HPQ	33.47%	25.77%	0.00%	0
IBM	17.21%	14.13%	0.00%	0
INTC	-24.89%	26.84%	0.00%	0
JNJ	8.57%	11.33%	31.33%	1
JPM	20.11%	16.99%	0.00%	0
KO	18.77%	10.64%	28.64%	1
MCD	28.79%	17.68%	0.00%	0
MMM	-0.78%	18.13%	0.00%	0
MO	16.41%	16.52%	8.66%	1
MRK	30.82%	18.83%	0.00%	0
MSFT	9.92%	20.87%	0.00%	0
PFE	10.05%	21.01%	0.00%	0
PG	10.12%	13.31%	0.00%	0
T	40.30%	16.80%	11.02%	1
UTX	10.09%	17.91%	0.00%	0
VZ	27.54%	16.10%	0.00%	0
WMT	-0.05%	16.87%	0.00%	0
XOM	27.20%	18.73%	11.05%	1

- *Encode which subset of equities to test using bit strings of 1's and 0's*
- *In this case, there are 30 bits with exactly 6 bits equal to 1*
- *In general, each bit string will have N bits with exactly K bits equal to 1*
- *It is inefficient to test every possible bit string. For N = 30, K = 6, there are 593,775 possible combinations!*

Example: Custom Evolution Algorithms

- Selection
  - *Retain* the best performing bit strings from one generation to the next. *Favor these for reproduction*

- Crossover
  - parent1 = [1 0 1 0 0 1 1 0 0 0]
  - parent2 = [1 0 0 1 0 0 1 0 1 0]
  - child = [1 0 0 0 0 1 1 0 1 0]

- Mutation
  - parent = [1 0 1 0 0 1 1 0 0 0]
  - child = [0 1 0 1 0 1 0 0 0 1]

Toolbox Comparison

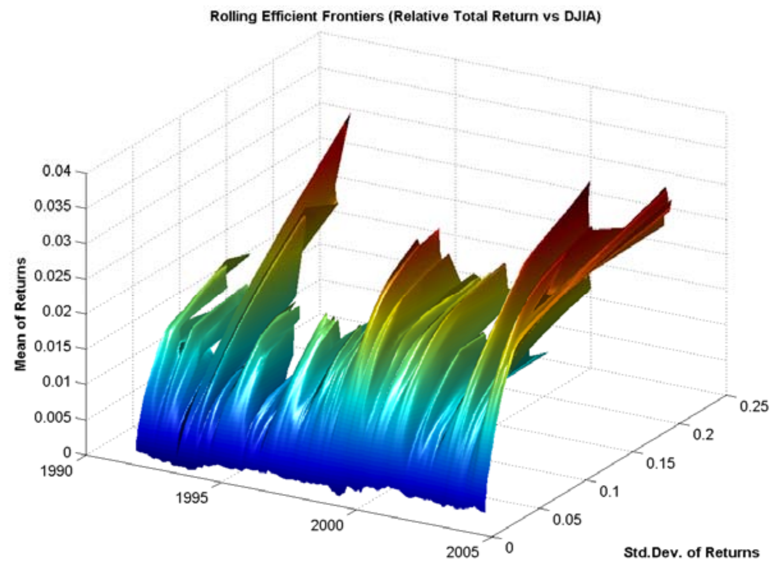
	Optimization Toolbox	Genetic Algorithm and Direct Search Toolbox
Faster	✓	
Can handle larger problems	✓	
Better on non smooth, noisy, stochastic problems		✓
More likely to find global solution		✓
Can solve problems that involve custom data types		✓

Agenda

Introduction
Local and Smooth Optimization
Example:
- Portfolio Optimization, part 1
- Expected Shortfall
- GARCH

Global or Non-Smooth Optimization
Example:
- Portfolio Optimization, part 2

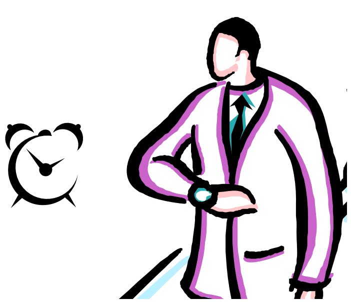
Parallel Computing
Summary



Parallel Computing with MATLAB

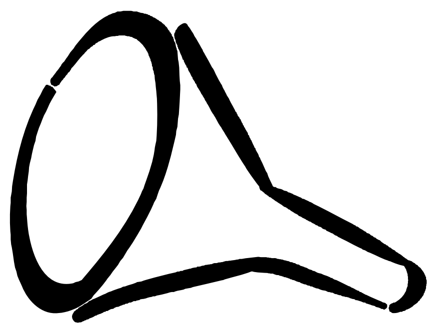
- Why parallel computing?
- How easy is it to use?
- Licensing
- Do I need special hardware?
- Parallel computing and optimization

Large Data Set Handling



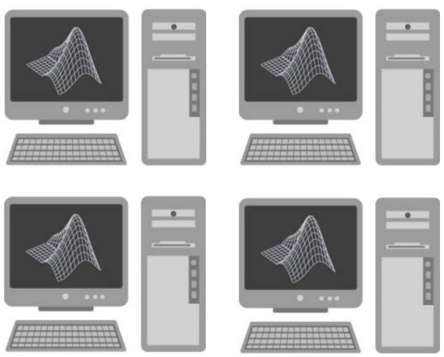
Workarounds

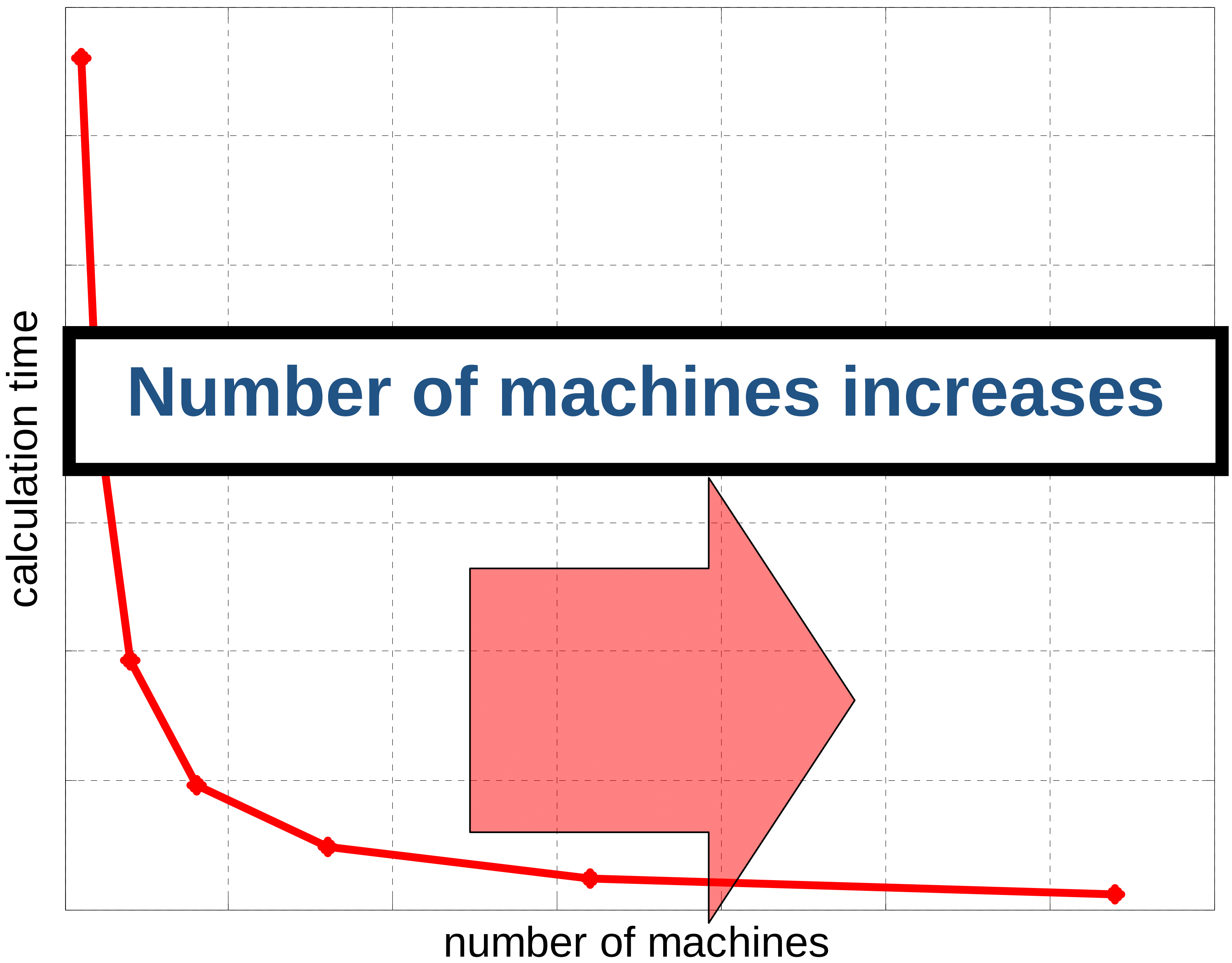
- **Reduce Data**
- **Wait**

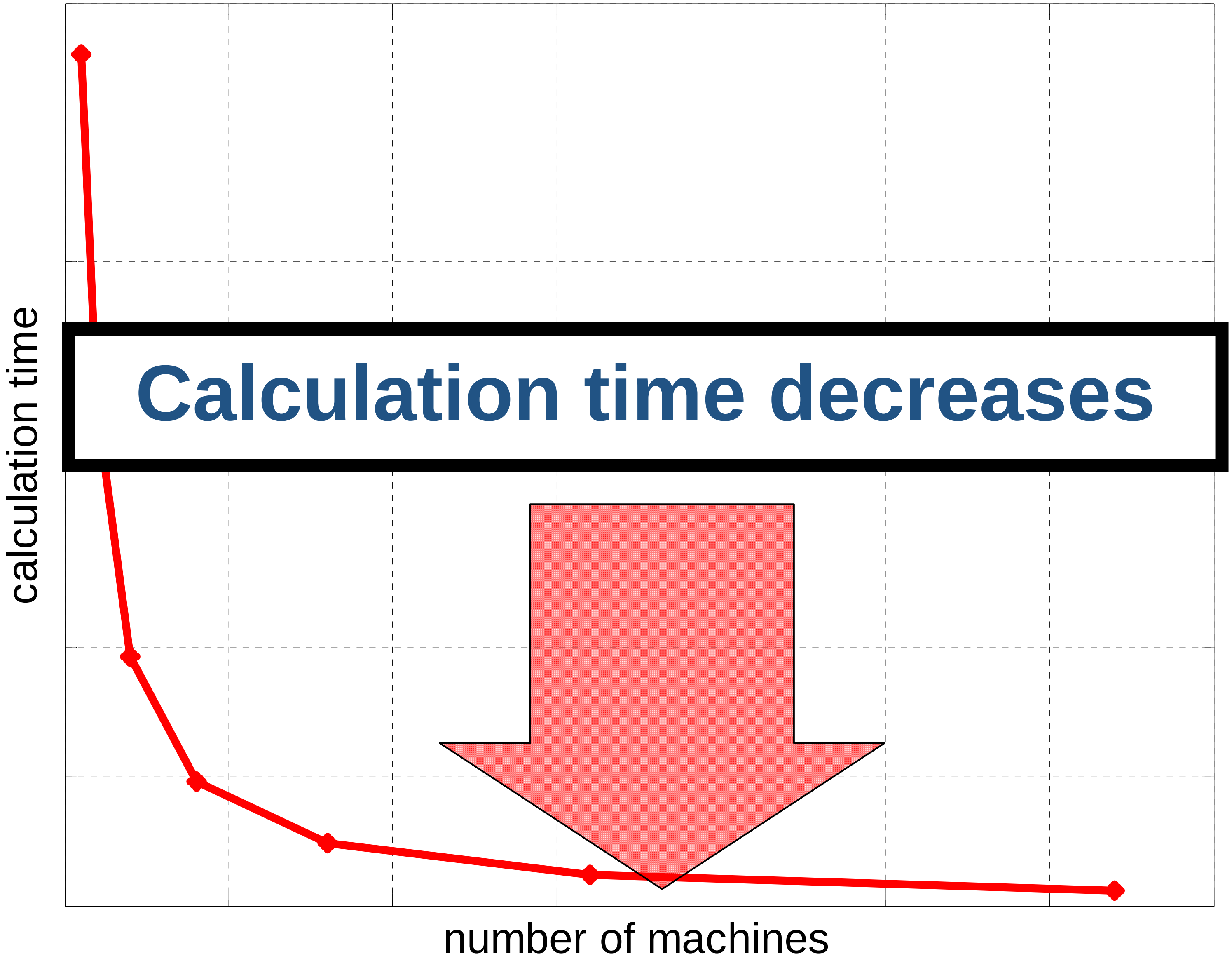


Solutions

- **Increase Addressable Memory**
- **Add Processors**



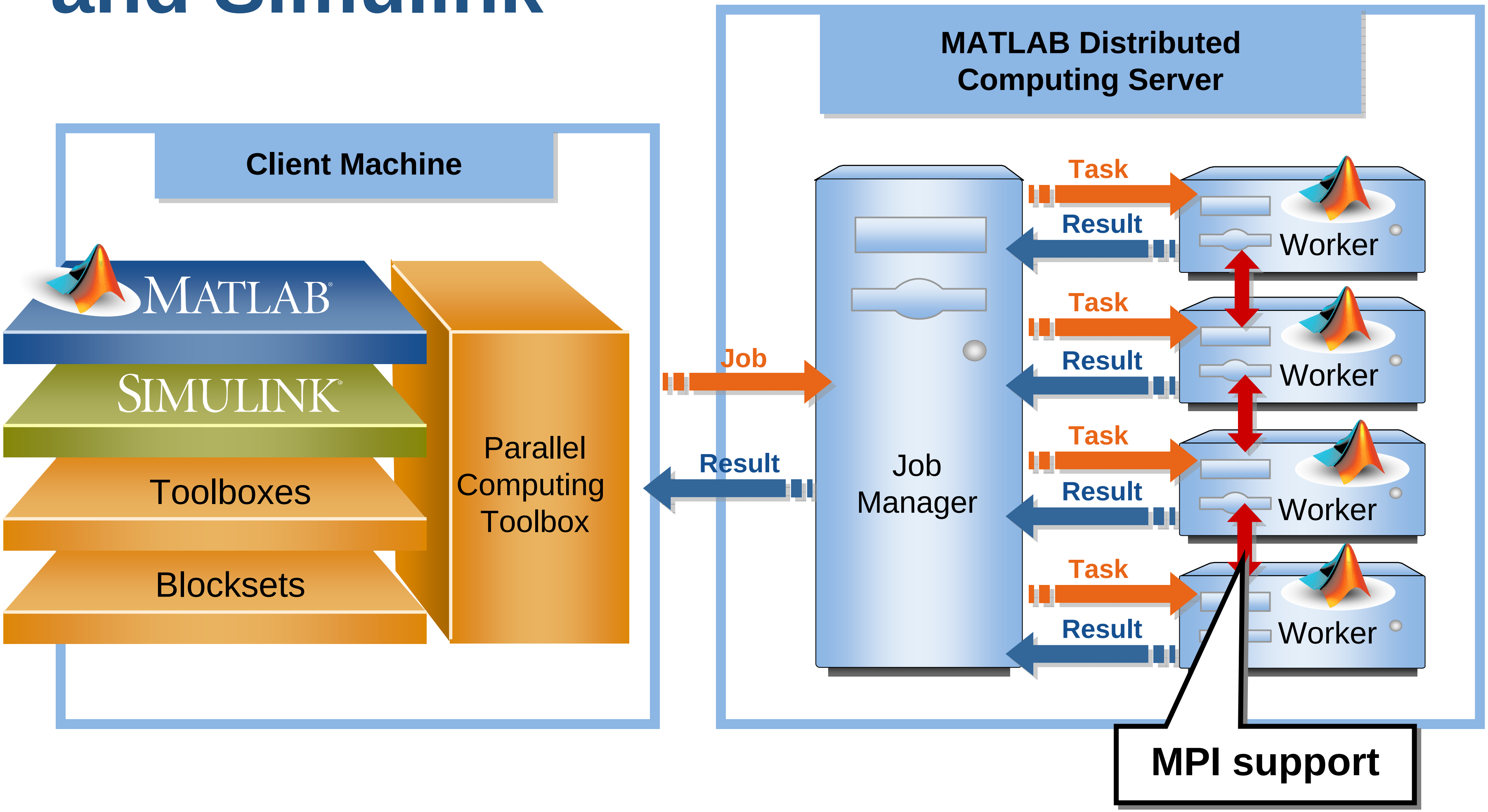

Number of machines increases
calculation time
number of machines


Calculation time decreases
calculation time
number of machines

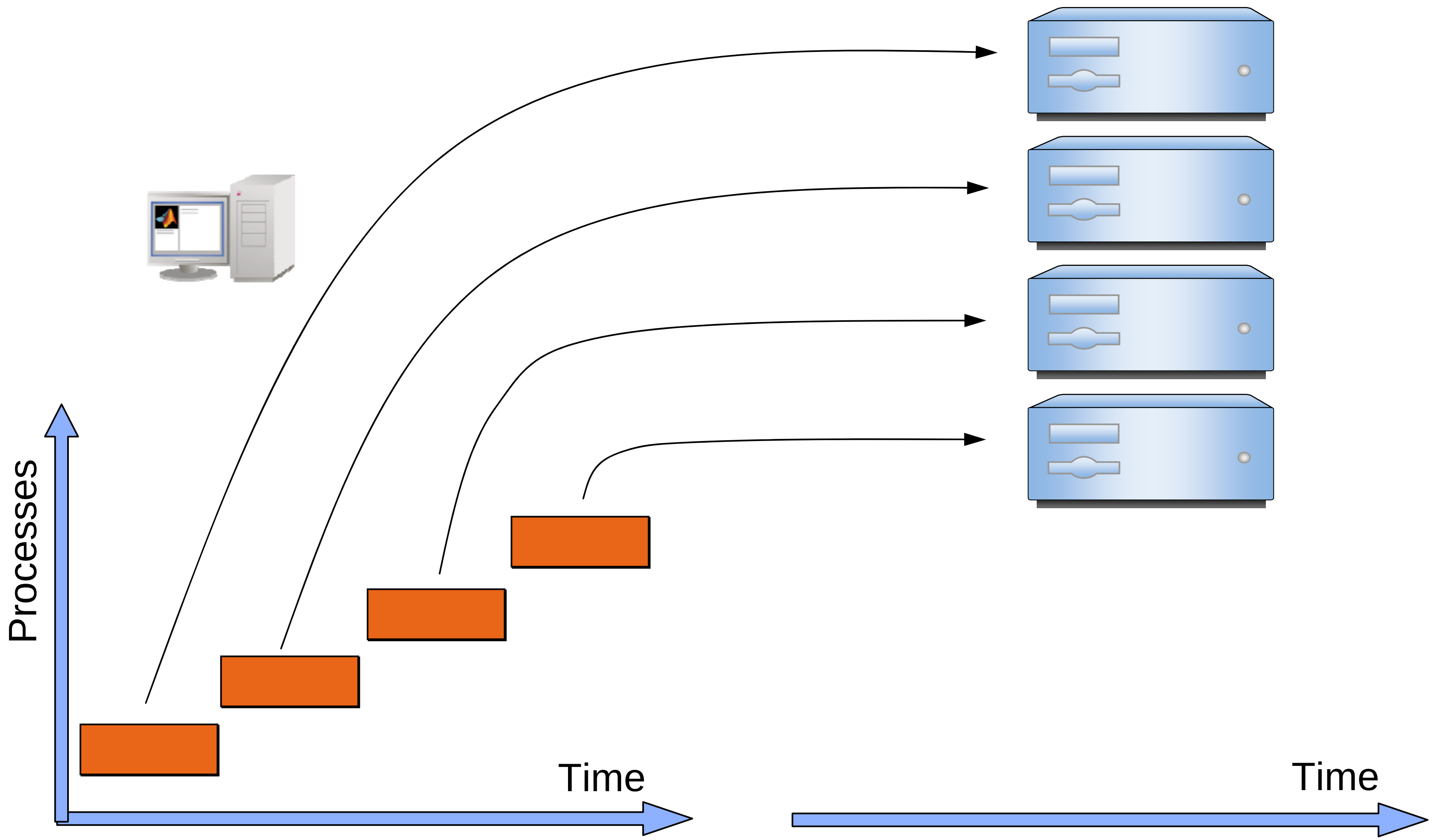
Parallel Computing with MATLAB

- Why parallel computing?
- How easy is it to use?
- Licensing
- Do I need special hardware?
- Parallel computing and optimization

Parallel Computing with MATLAB and Simulink



Distributing Tasks (Task Parallel)

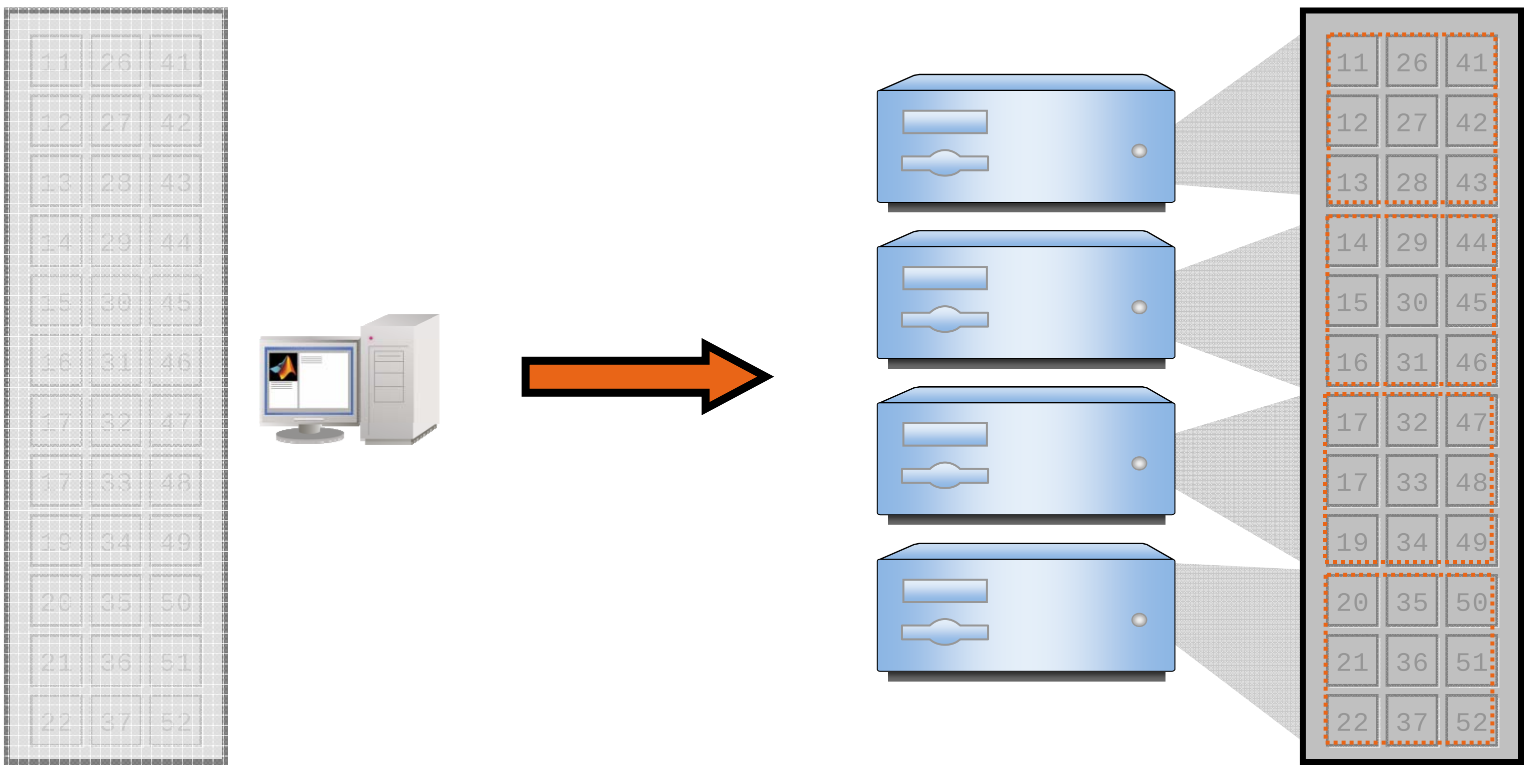


Parallel for loops

```
parfor (i = 1 : n)
   % do something with i
end
```

- Mix task parallel and serial code in the same function
- Run loops on a pool of MATLAB resources
- Iterations must be order-independent
- M-Lint analysis helps to identify if existing for loops can be changed to parfor

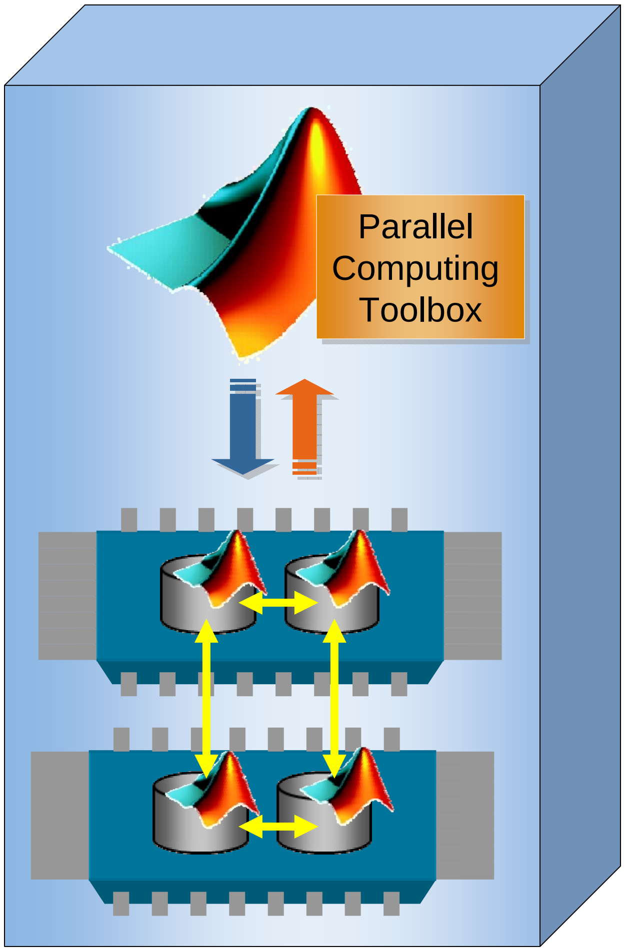
Large Data Sets (Data Parallel)



Parallel Computing with MATLAB

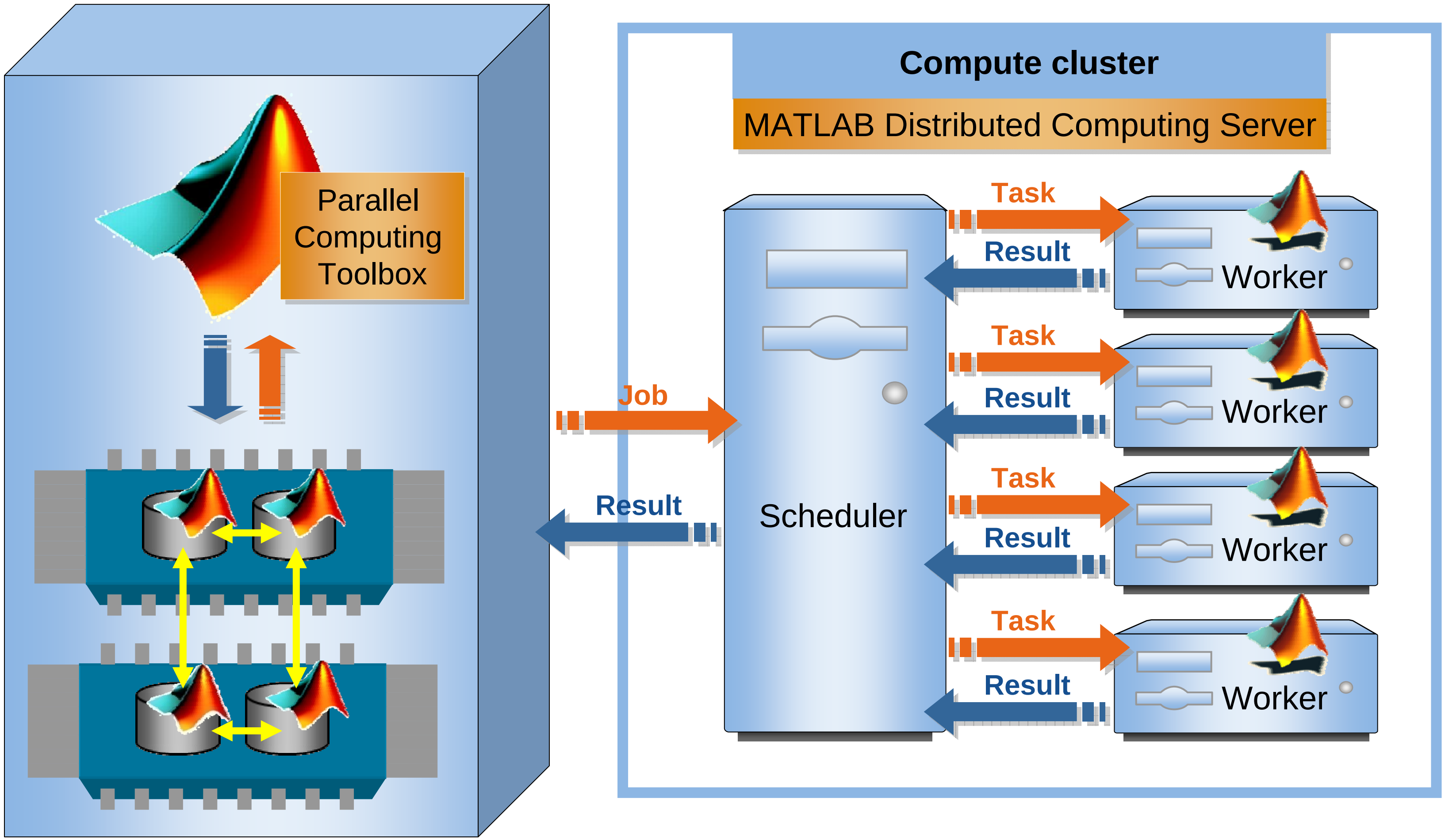
- Why parallel computing?
- How easy is it to use?
- Licensing
- Do I need special hardware?
- Parallel computing and optimization

Run *eight local* workers with a PCT license

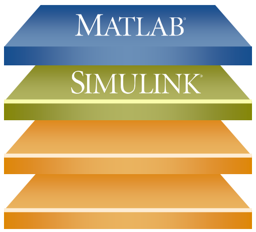


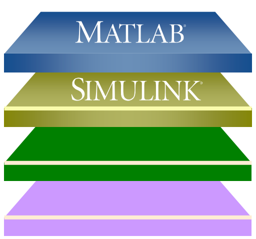
- Easy to experiment with explicit parallelism on multi-core machines
- Rapidly develop distributed and parallel applications on local computer
- Take full advantage of desktop power
- No separate compute cluster required

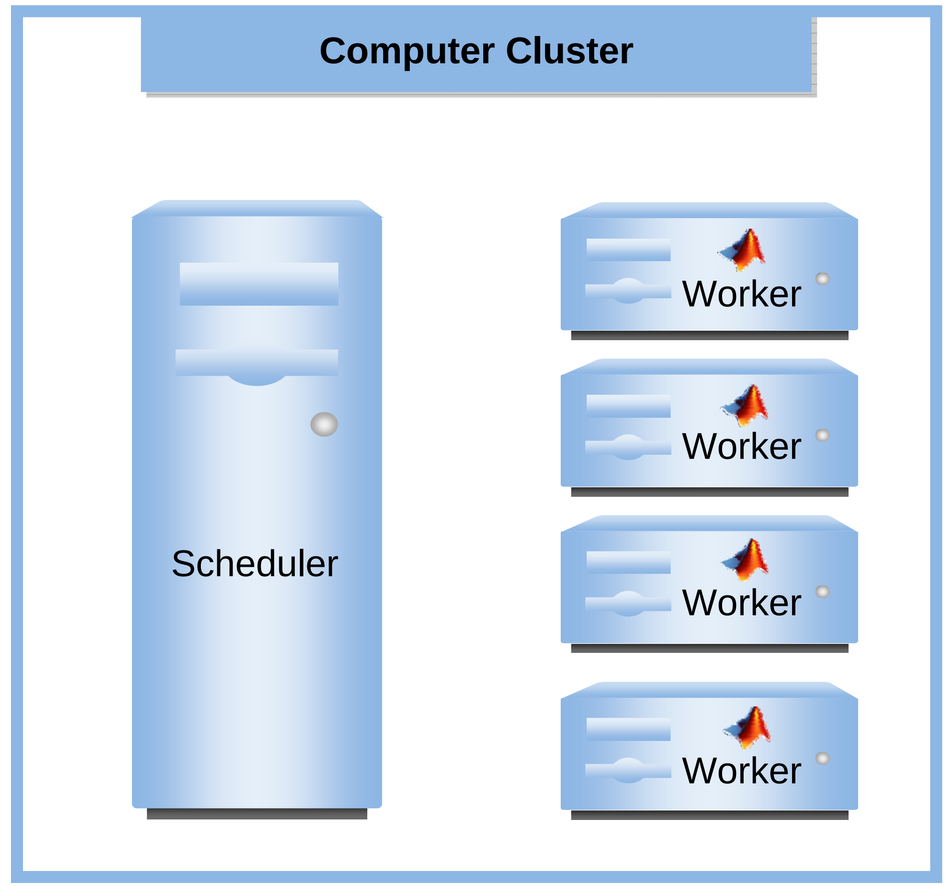
Scale up to cluster configuration with no code changes



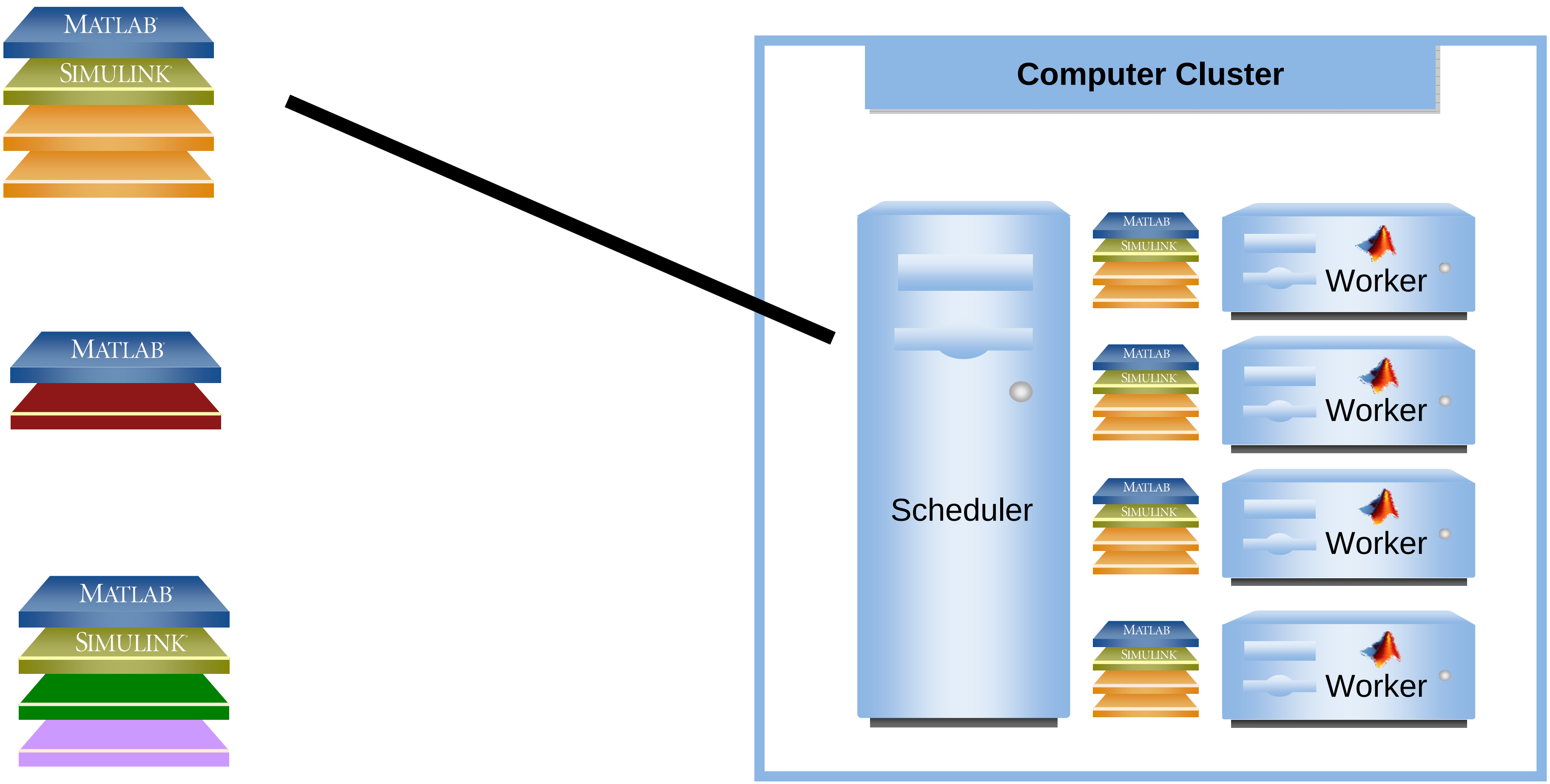
Dynamic Licensing



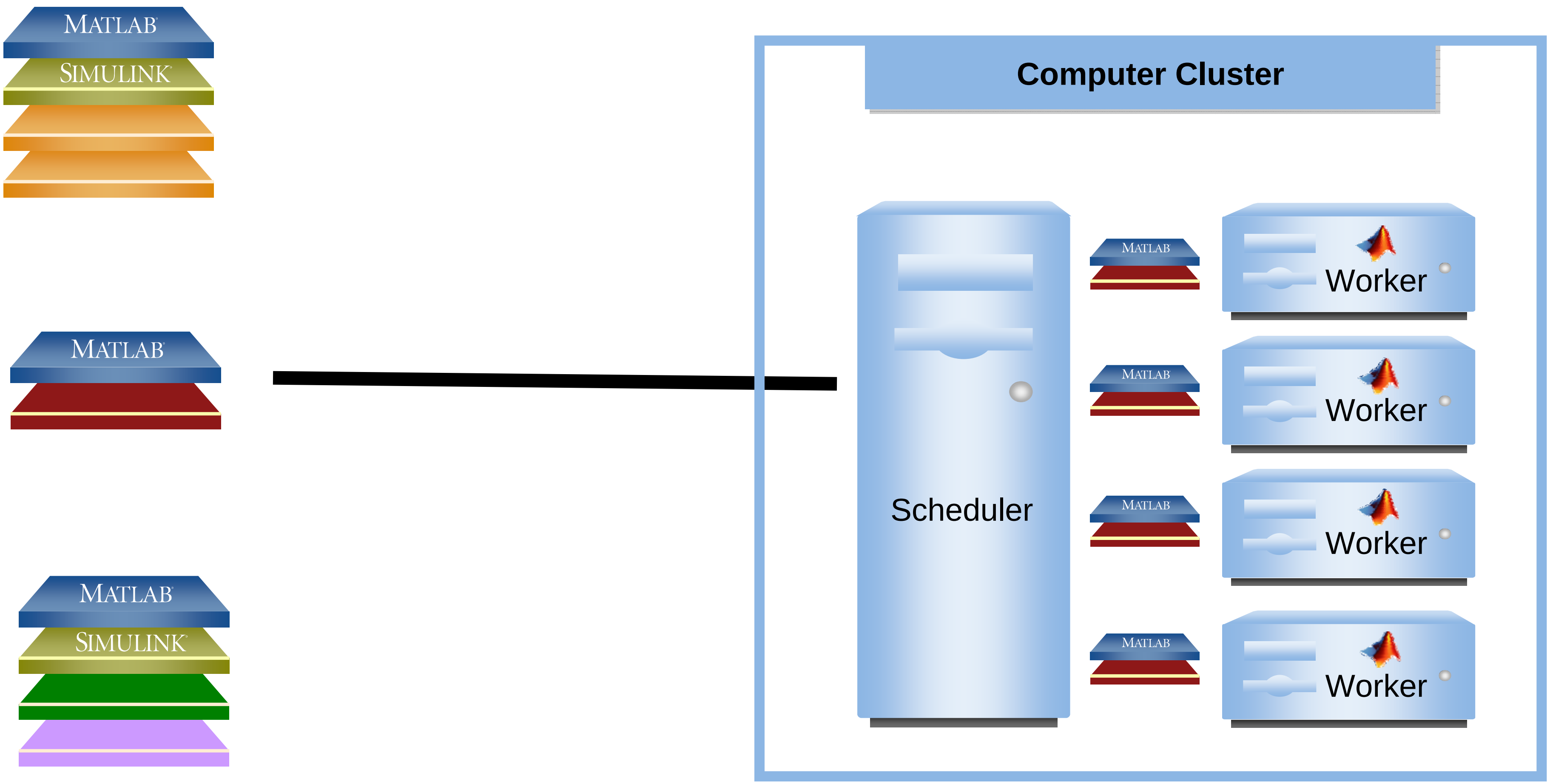




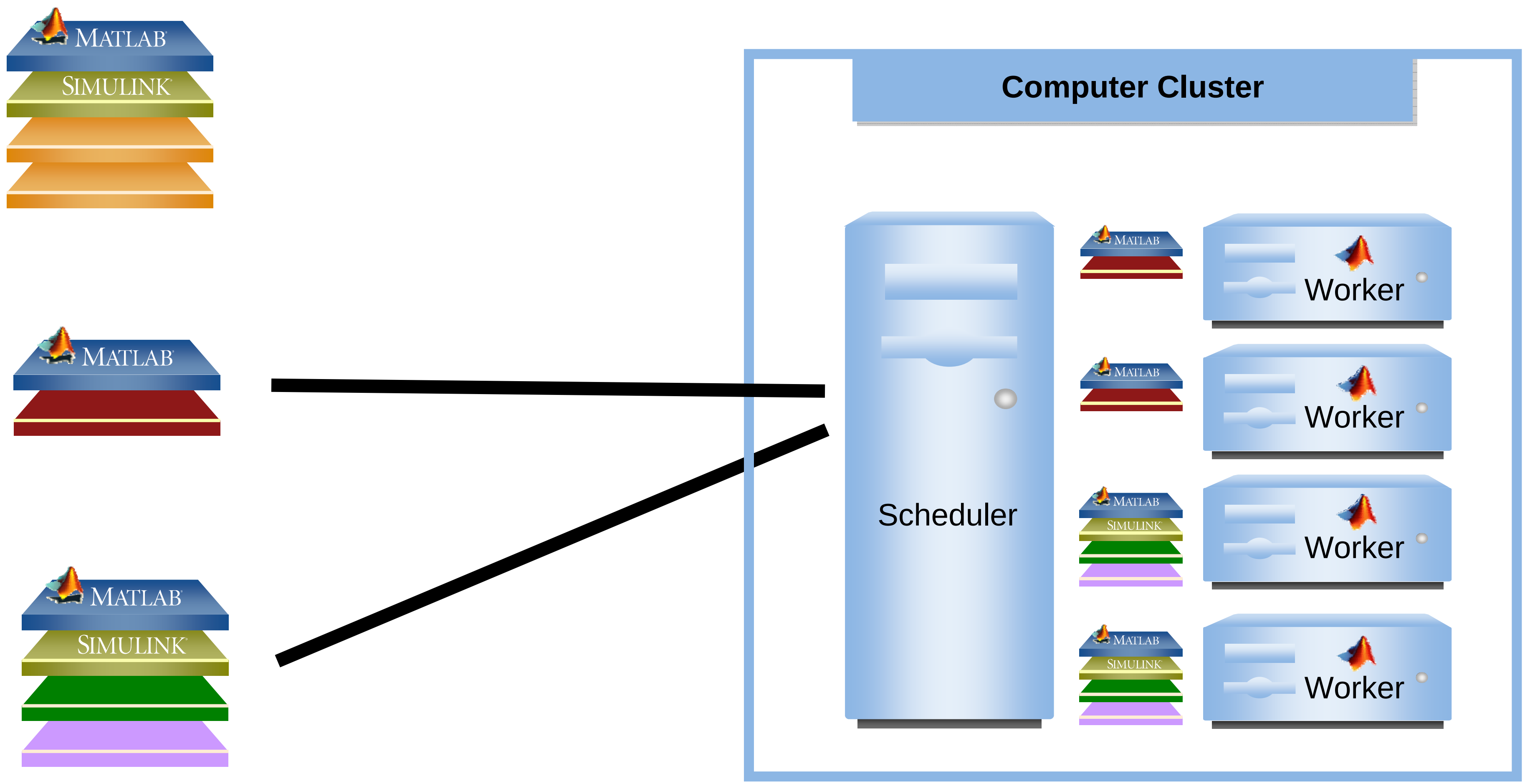
Dynamic Licensing



Dynamic Licensing



Dynamic Licensing – Multiple Users



Parallel Computing with MATLAB

- Why parallel computing?
- How easy is it to use?
- Licensing
- Do I need special hardware?
- Parallel computing and optimization

Supported on all MATLAB platforms

Parallel Computing with MATLAB

- Why parallel computing?
- How easy is it to use?
- Licensing
- Do I need special hardware?
- Parallel computing and optimization

Parallel Computing and Optimization

Both,

- the Optimization Toolbox and
- the Genetic Algorithms and Direct Search Toolbox

contain solvers which incorporate

- parallel optimization functionality,
- parallel estimation of gradients,
- nested parallel functions

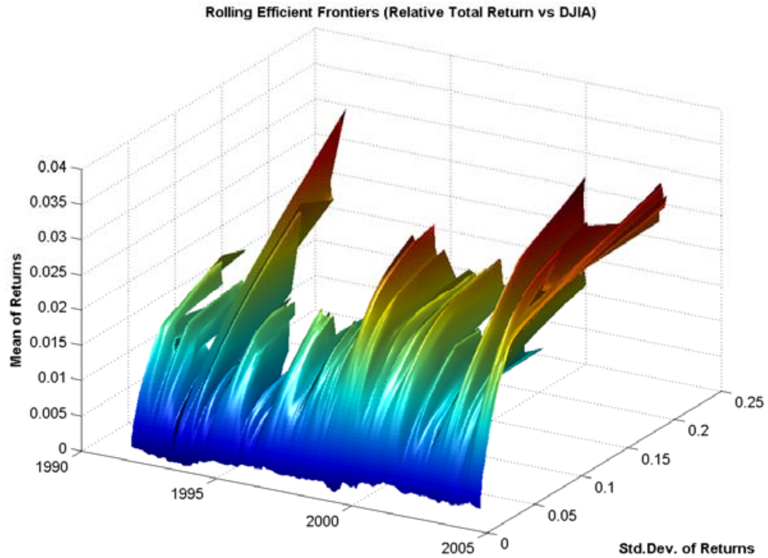
provided the Parallel Computing Toolbox is available.

Parallel Computing with MATLAB

- Why parallel computing?
  - Process large amounts of data faster
- How easy is it to use?
  - Small modifications to your existing MATLAB programs
- Licensing
  - Dynamic!
- Do I need special hardware?
  - Windows, Linux, Solaris, or Mac are fine
- Parallel computing and optimization
  - Nicely integrated

Agenda

- Introduction
- Local and Smooth Optimization
- Example:
  - Portfolio Optimization, part 1
  - Expected Shortfall
  - GARCH
- Global or Non-Smooth Optimization
- Example:
  - Portfolio Optimization, part 2
- Parallel Computing
- Summary



Benefits

- Solvers for a wide range of optimization problems
- Easy to use
- Optimization supported by parallel computing