TECHILA

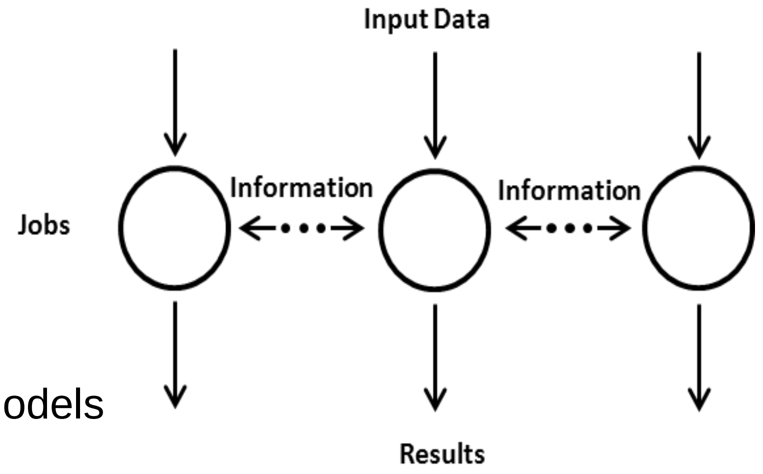
End-User Training

Training day structure

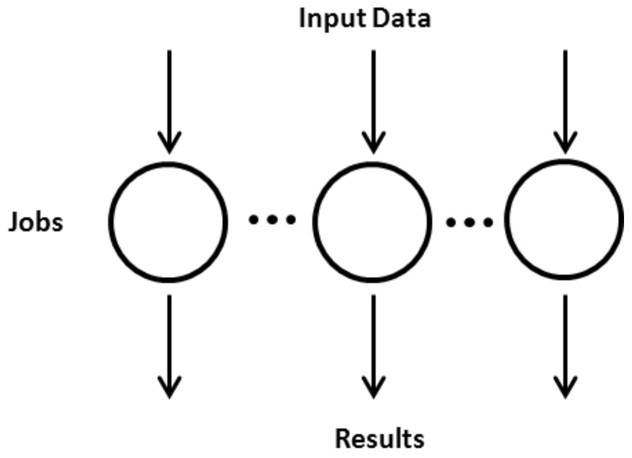
- **General information on the Techila system (10-11)**
  - Operating principle of Techila, available features, terminology, concept of gridification, general things to consider
  - Quick introduction of the Techila Grid Management Kit and Techila Web Interface
- **Techila with Matlab (11-12)**
  - Short introduction on the main interfaces; Peach and GridFor
  - Preparation, testing & hands on labs using MATLAB
- **Lunch (12-13)**
- **Techila with R (13-14)**
  - Short introduction on the R Peach interface and of the required packages
  - Preparation, testing & hands-on labs using R
- **Techila with Python and Java (14-15)**
  - Short intros on the Python and Java Peach syntaxes and required preparation steps
  - Hands-on labs

Distributed Computing Problems

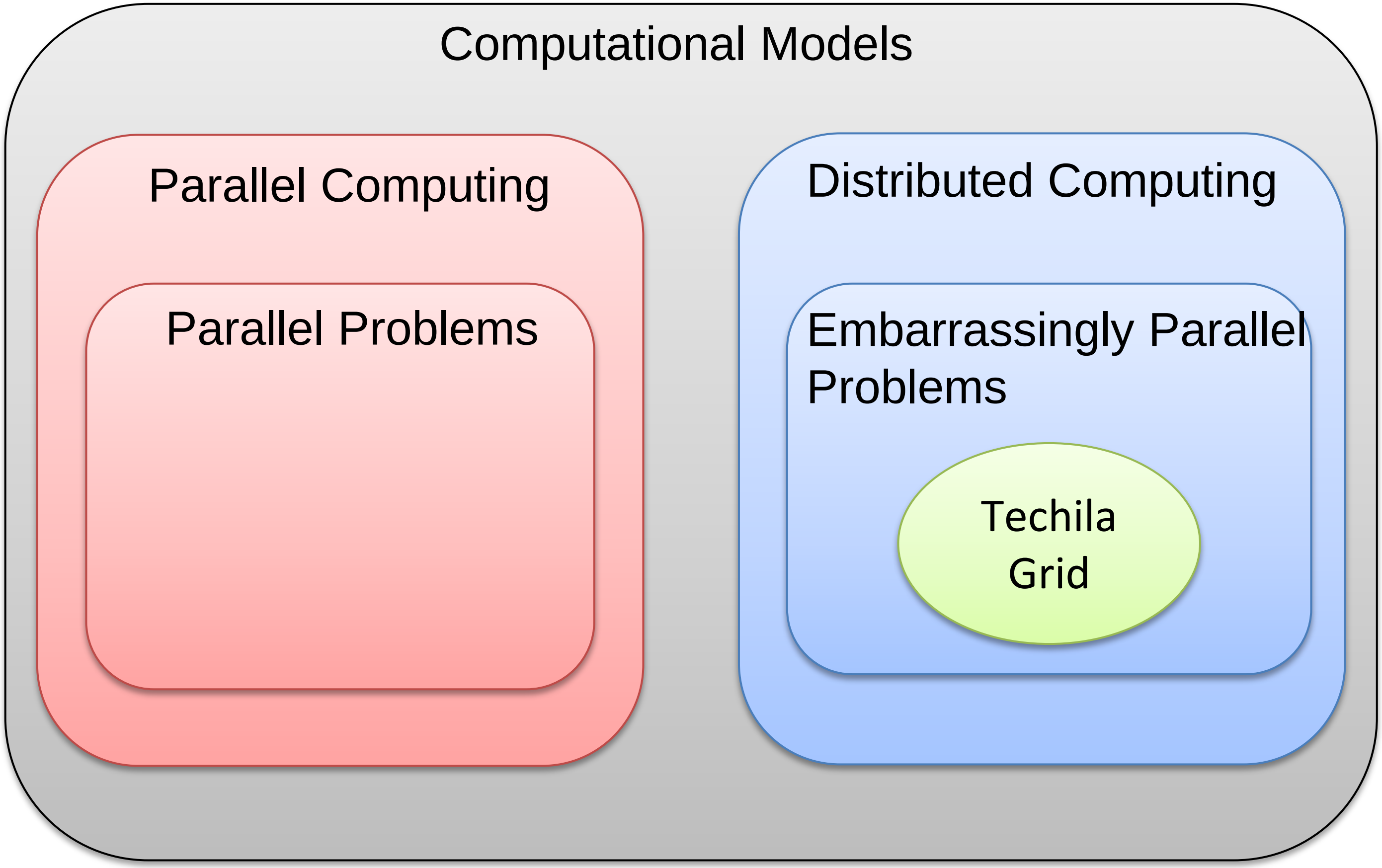
- **Parallel Problems**
  - Jobs depend on each other’s states
  - Communication between jobs
  - For example fluid dynamics or finite element models
  - Usually not suitable for Grid-type environment



- **Embarrassingly Parallel Problems**
  - Jobs are totally independent
  - No communication between jobs
  - For example Monte Carlo, MCMC
    - Suitable for distributed computing → Techila Grid

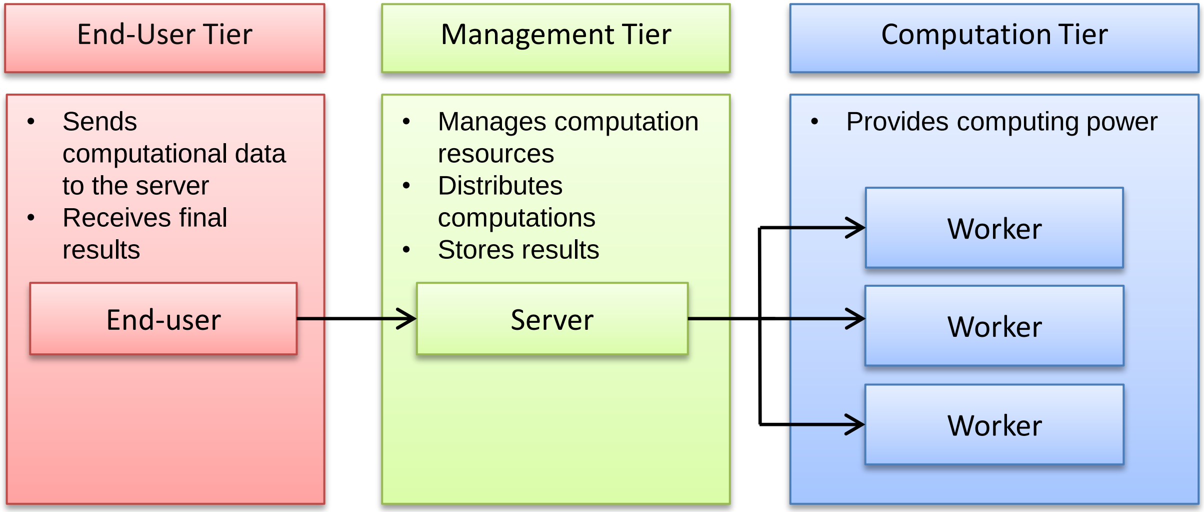


Computational Models



Roles

- **Worker**
  - Workstations, laptops, clusters
  - Windows, Linux, Mac OS X
- **Server**
  - Management
  - Security
- **End-User**
  - Researcher
  - Using his/her own workstation



End-User Tier
Management Tier
Computation Tier
• Sends computational data to the server
• Receives final results
End-user
• Manages computation resources
• Distributes computations
• Stores results
Server
• Provides computing power
Worker
Worker
Worker

Authentication

- **Authentication with End-User specific Keys (typically username.jks)**
  - All the connections are secured using the End-User Key
  - All the bundles are signed with the End-User Key
  - Stored in a password protected keystore
  - **Handle with care → do not store in an insecure location**

Techila Grid Management Kit

- **Enables access to the Techila environment**
  - Latest version available at the Techila Extranet and http://www.techila.fi/TechilaGMK.zip
  - Techila Extranet located at: www.techila.fi/extranet/
    - Requires reqistration
- **Contains:**
  - Techila Grid Getting Started document
  - Examples for various programming languages, including MATLAB and R
  - Examples on how to distribute binaries by using the Command Line Interface (CLI)
  - End-User Guides for MATLAB, R and the CLI with walkthroughs of the examples
- **Easy to update:**
  - Simply download the new version and extract over the old installation
  - Configuration settings in the grid_settings.ini file will not be overwritten

Configuring the grid_settings.ini file

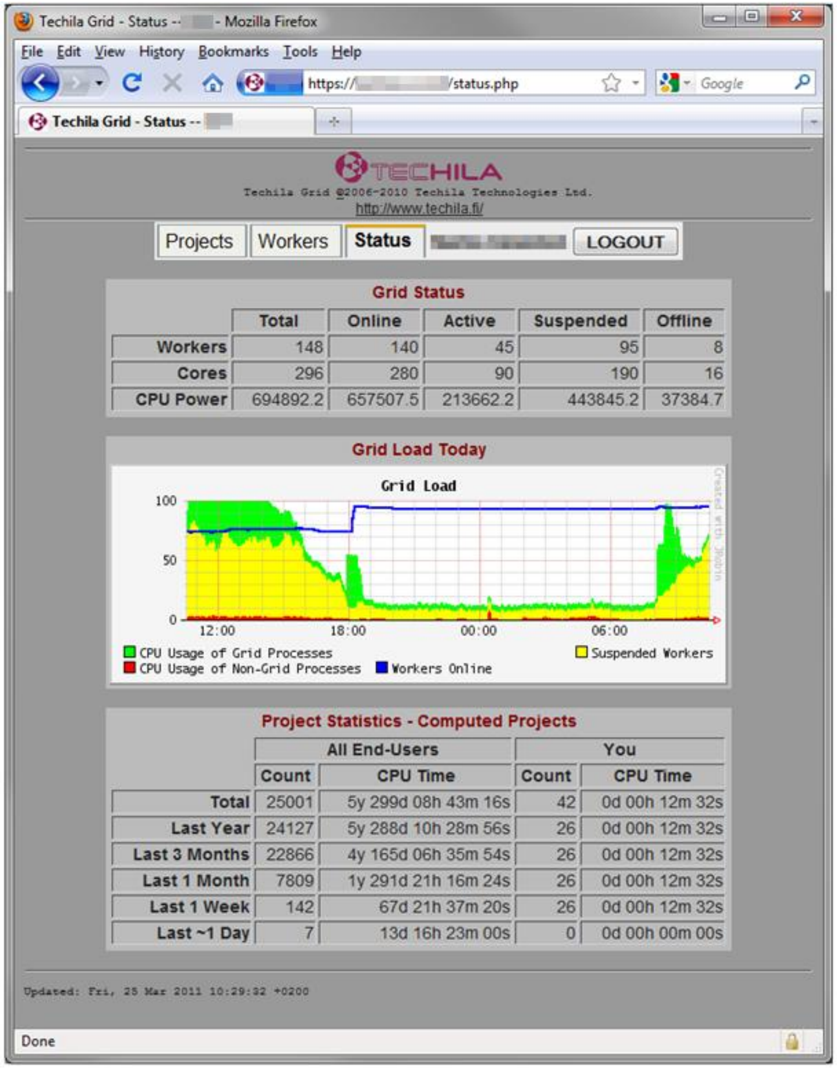
- **Configuration needed to define the address of the Techila Server and location of the keystore file**
- **Steps:**

1. Download and extract the TechilaGMK.zip on your computer
2. Navigate to the 'gmk' directory in the Techila Grid Management Kit
3. Rename the file 'grid_settings.ini.template' to 'grid_settings.ini'
4. Open the 'grid_settings.ini' with a text editor
5. Modify the following parameters
    - hostname = techila.mathstat.helsinki.fi
    - alias = <The alias of your End-User Key>
    - keystore = <Location of the keystore (.jks) file>
6. Save changes

Techila Web Interface

- Information on the Techila environment
- Contains information on your Projects
- Can be used for stopping, removing and restarting Projects
- Provides easy access to error messages
- Login required
- Status page located at:

  https://techila.mathstat.helsinki.fi/status.php


Techila Grid - Status -- [redacted] - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://[redacted]/status.php
Google
Techila Grid - Status -- [redacted]
TECHILA
Projects
Workers
Status
[redacted]
LOGOUT
Grid Load
100
50
0
12:00
18:00
00:00
06:00
🟩 CPU Usage of Grid Processes
🟥 CPU Usage of Non-Grid Processes
🟦 Workers Online
🟨 Suspended Workers
Created with JRobin
Done

Techila Grid ©2006-2010 Techila Technologies Ltd.
http://www.techila.fi/

Grid Status

	Total	Online	Active	Suspended	Offline
Workers	148	140	45	95	8
Cores	296	280	90	190	16
CPU Power	694892.2	657507.5	213662.2	443845.2	37384.7

Grid Load Today

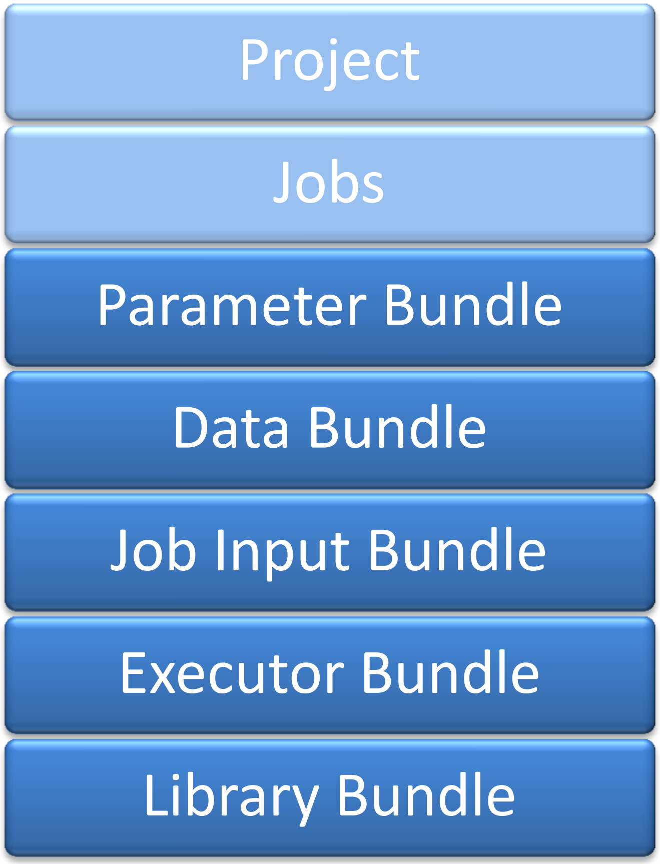
Project Statistics - Computed Projects

	All End-Users		You	
	Count	CPU Time	Count	CPU Time
Total	25001	5y 299d 08h 43m 16s	42	0d 00h 12m 32s
Last Year	24127	5y 288d 10h 28m 56s	26	0d 00h 12m 32s
Last 3 Months	22866	4y 165d 06h 35m 54s	26	0d 00h 12m 32s
Last 1 Month	7809	1y 291d 21h 16m 24s	26	0d 00h 12m 32s
Last 1 Week	142	67d 21h 37m 20s	26	0d 00h 12m 32s
Last ~1 Day	7	13d 16h 23m 00s	0	0d 00h 00m 00s

Updated: Fri, 25 Mar 2011 10:29:32 +0200

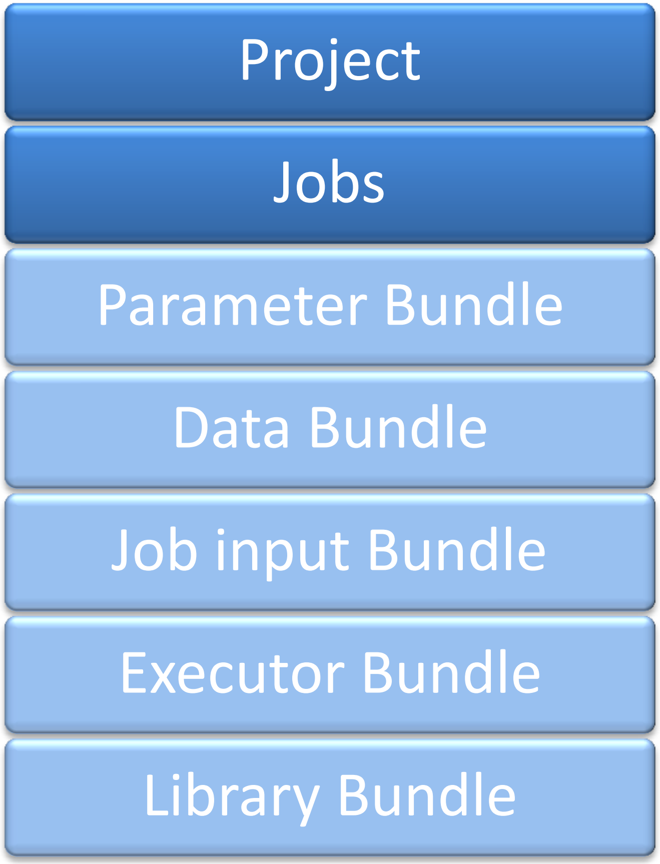
Terminology

- **Bundles**
  - Created automatically when creating a Project
  - Containers for data, binaries, libraries
    - Used to transfer all necessary components to Workers
  - Dependencies on other bundles
    - All required Bundles are transferred automatically
  - Security
    - Only signed Bundles allowed in the Techila system
  - Life Cycle Management
    - Expiration times can be defined to automatically remove old, unused Bundles
  - Deployed from Server to Workers on-demand

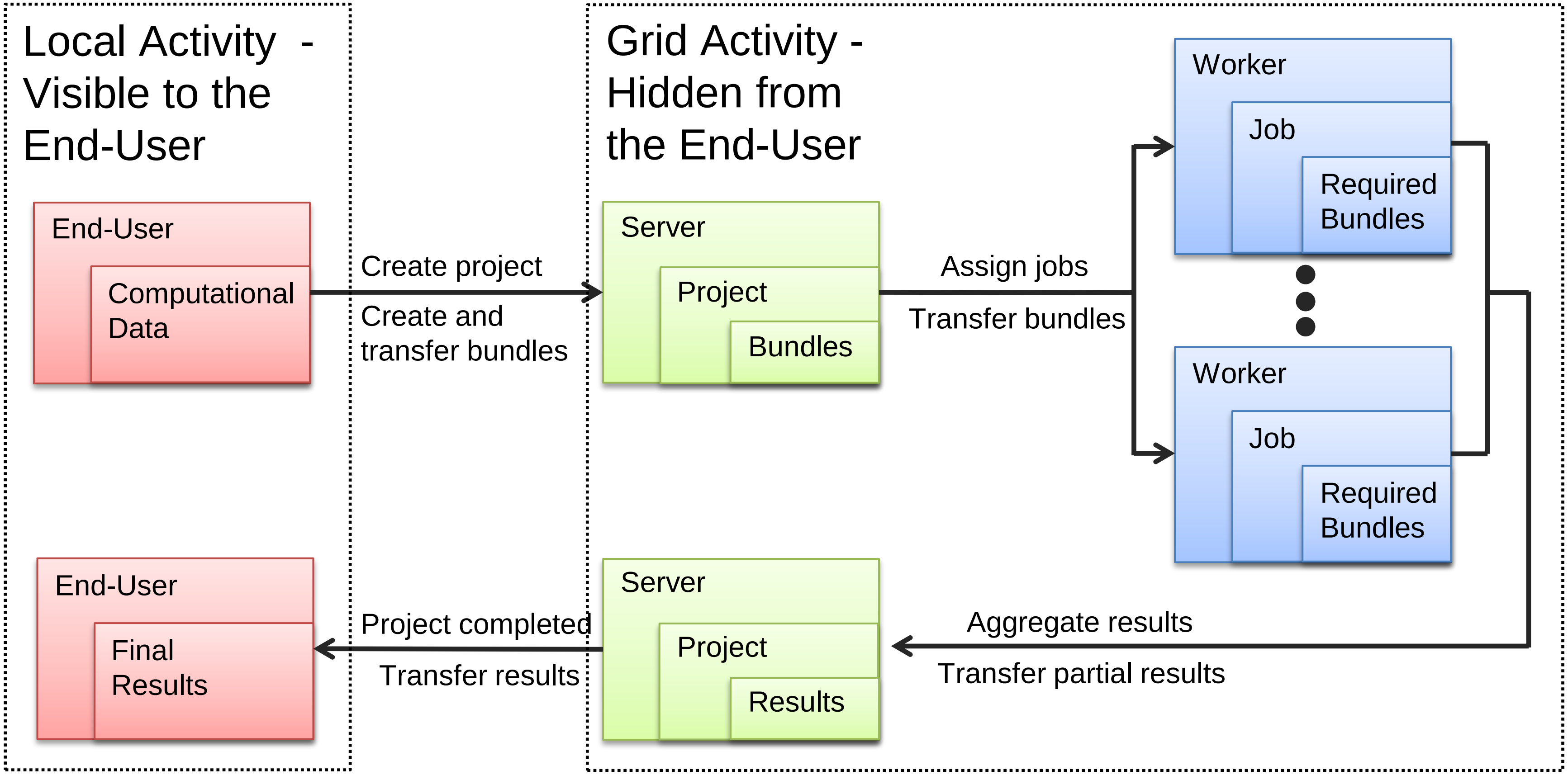


Terminology

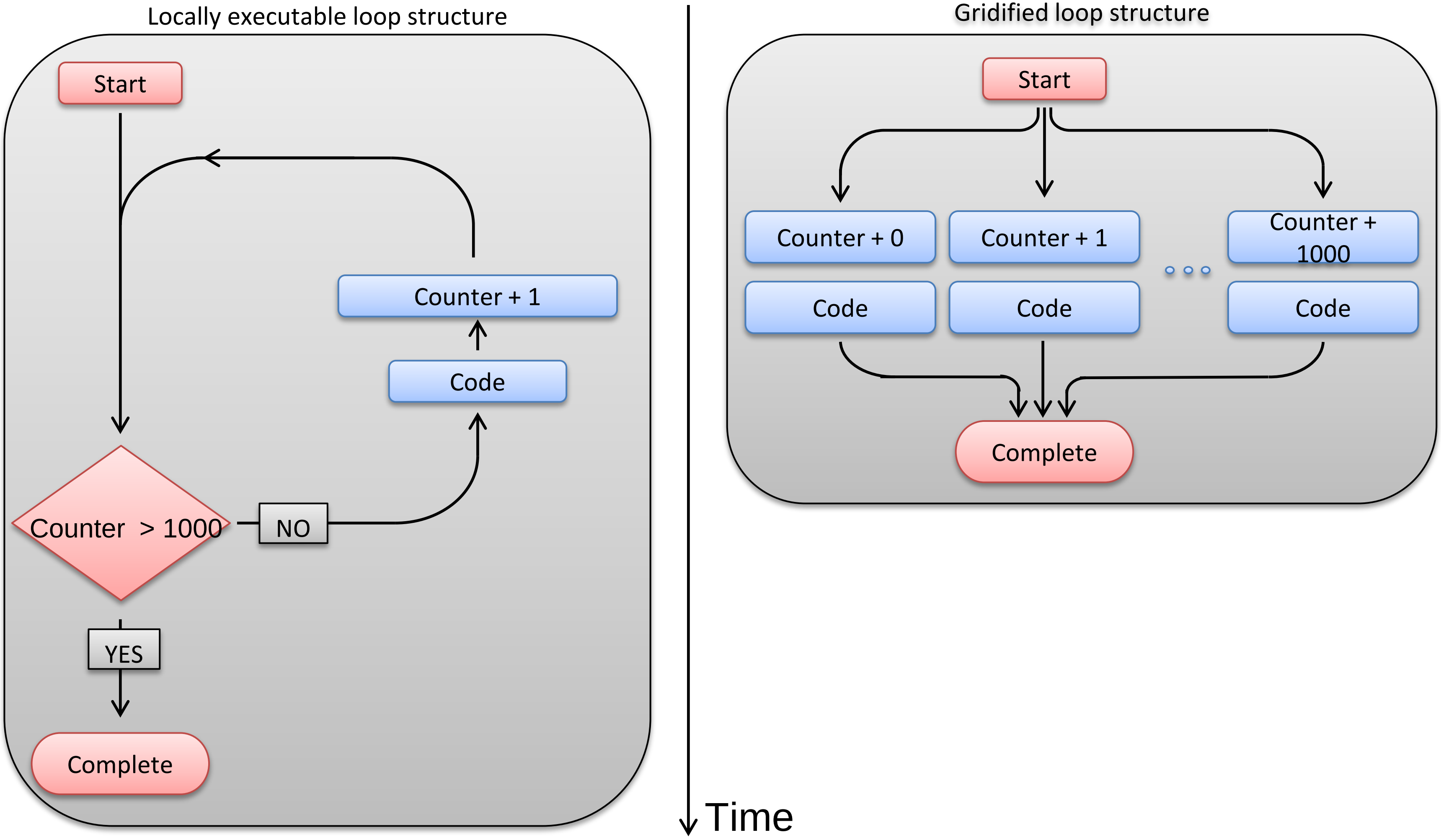
- **Project**
  - Computational problem
  - Container for Jobs
  - Control parameters for the execution
- **Jobs**
  - Smallest units in the computational problem
  - Partial problems
  - Deployed and solved on the Workers



Process Flow



Gridification



Gridification

- **Locate the demanding parts of the code**
  - Profiling (MATLAB profiler)
    - Information with ”doc profiler” in MATLAB
  - Timing (debug printouts with timestamps)
- **Divide the code into two parts**
  - Demanding part (individually executable) → Worker Code
  - Other code → Local Control Code
  - Find out all the parameters needed to be delivered to and from demanding part
- **Make sure everything works**
  - Call ”Worker Code” from ”the Local Control Code” with the necessary parameters
- **Gridify**
  - Change the call to Grid call → PEACH()

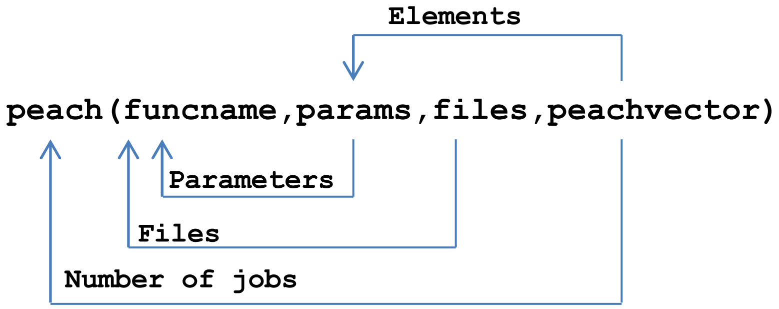
Before Running the code in Grid

- **Make sure the code works locally**
- **Check the memory consumption**
  - The workers are usually workstations without large memory space
- **Check the IO-load vs CPU-load**
  - Computation should be more CPU-intensive than data-intensive
- **Check the size of the input data and output data**
  - The network latency to transfer the data may reduce performance
- **Check the length of a single job**
  - Extremely short jobs (less than few seconds) are not effective because of network latency
  - Long jobs may get interrupted (and restarted) because of unstable environment (reboots)

    → Snapshot support!!!

Peach

- **Peach**
  - Finds prerequisites for function
  - If necessary, compiles the function with prerequisites
  - Deploys the executable program to the Grid
  - Executes the program on Workers with given parameters
  - Additionally transfers datafiles to the Workers
- **Peachvector**
  - Tells the number of the jobs to be created into project
  - Gives job-specific input data for the project (”<param>”)



```
peach(funcname,params,files,peachvector)
```

peachvector					
Index	1	2	3	4	5
Elements	1	2	3	4	5

peachvector		
Index	1	2
Elements	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$

Peach

Original Code

```
...
for x=1:length(S0)
 for y=1:length(sigma0)
    price(y,x) = asian(S0(x),sigma0(y)^2,M,nn,r,N,rho,kappa,psi,E,T);
  end
end
...
function [price] = asian(S0,v0,M,nn,r,N,rho,kappa,psi,E,T)
...
```

Peach Control Code

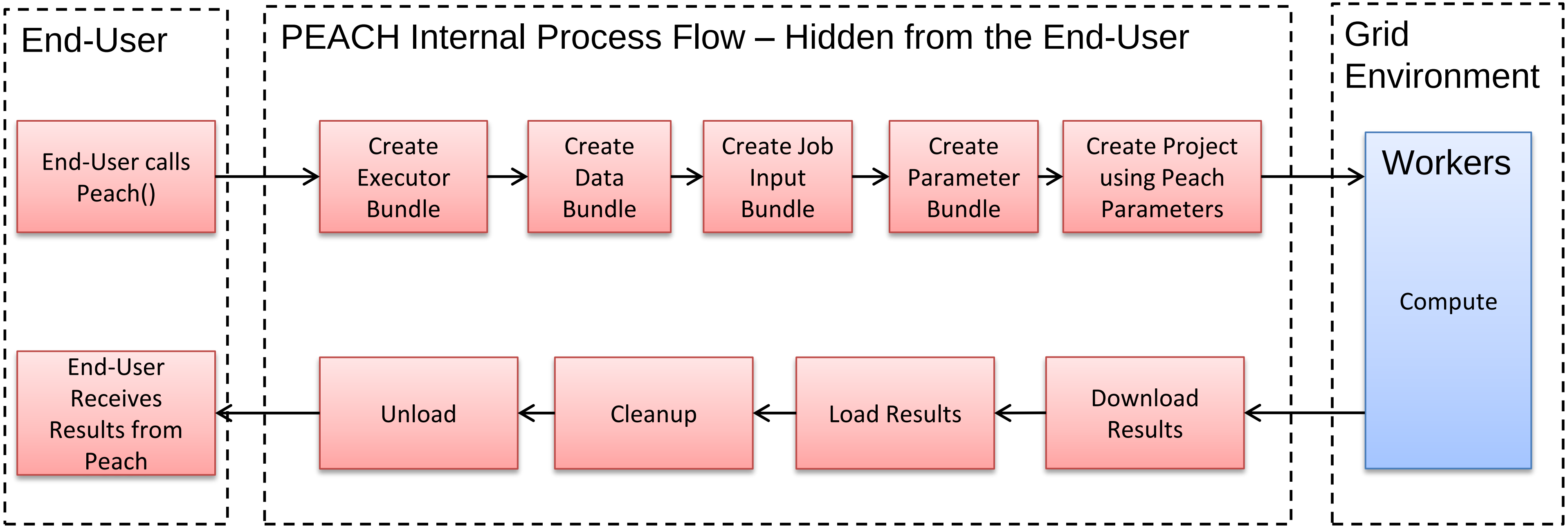
```
...
price = peach('asian', {S0, sigma0.^2, M, nn, r, N, rho, kappa, psi, E, T, '<param>'},
1:length(S0)*length(sigma0));
price = cell2mat(reshape(price,length(sigma0),length(S0)));
...
```

Peach Worker Code

```
function [price] = asian(Sx,vx,M,nn,r,N,rho,kappa,psi,E,T,jobidx)
[j, i] = ind2sub([length(vx), length(Sx)], jobidx);
S0 = Sx(i);
v0 = vx(j);
...
```

Peach

- **Single command interface to Techila Grid**
  - Simplest form: peach(funcname,params,peachvector)



GridFor

- **Currently available for MATLAB**

Original Code

```
...
for x=1:length(S0)
 for y=1:length(sigma0)
  price(y,x) = asian_montecarlo(S0(x),sigma0(y)^2,M,nn,r,N,rho,kappa,psi,E,T);
 end
end
...
function [price] = asian_montecarlo(S0,v0,M,nn,r,N,rho,kappa,psi,E,T)
...
```

GridFor Code

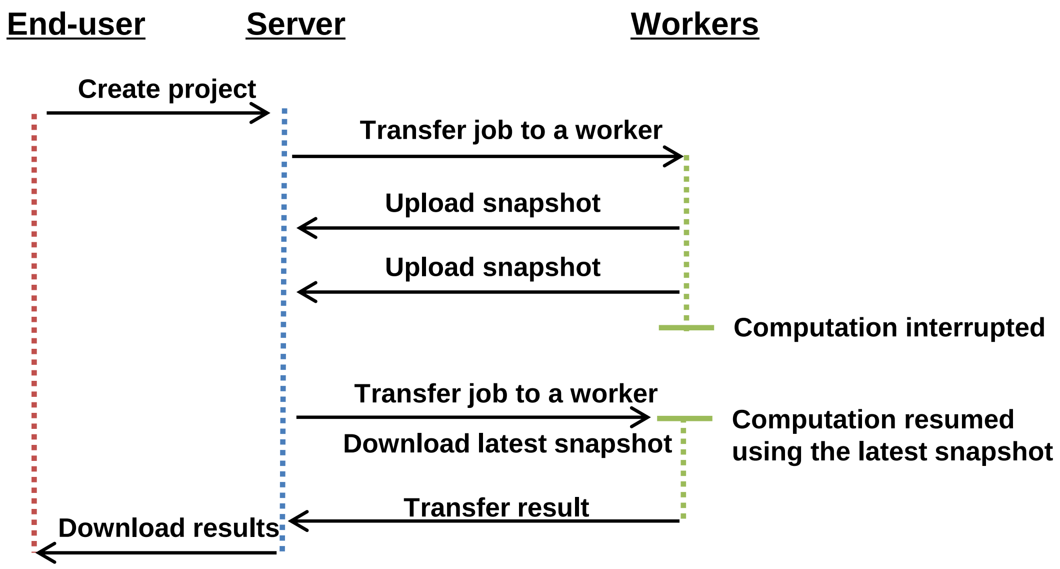
```
...
gridfor x=1:length(S0)
 gridfor y=1:length(sigma0)
  price(y,x) = asian_montecarlo(S0(x),sigma0(y)^2,M,nn,r,N,rho,kappa,psi,E,T);
 gridend
gridend
...
function [price] = asian_montecarlo(S0,v0,M,nn,r,N,rho,kappa,psi,E,T)
...
```

Features:

- **Can be used to improve efficiency and implement more complex distributions**
- **Features include:**
  - Snapshots
  - Streaming
  - Callback functions
  - Job-specific input files
  - Distributing precompiled binaries
  - Detached Projects
  - ...

Features: Snapshots

- **Saving project state**
  - Requires support in computation code
    - To save the state
    - To resume from the saved state
- **Saves time in long runs**
  - Resuming after reboot
  - Resuming on another Worker
  - Optimizing → transfer to faster Worker



Features: Snapshots

- **Saving project state**
  - Requires support in computation code
    - To save the state
    - To resume from the saved state
- **Saves time in long runs**
  - Resuming after reboot
  - Resuming on another Worker
  - Optimizing → transfer to faster Worker

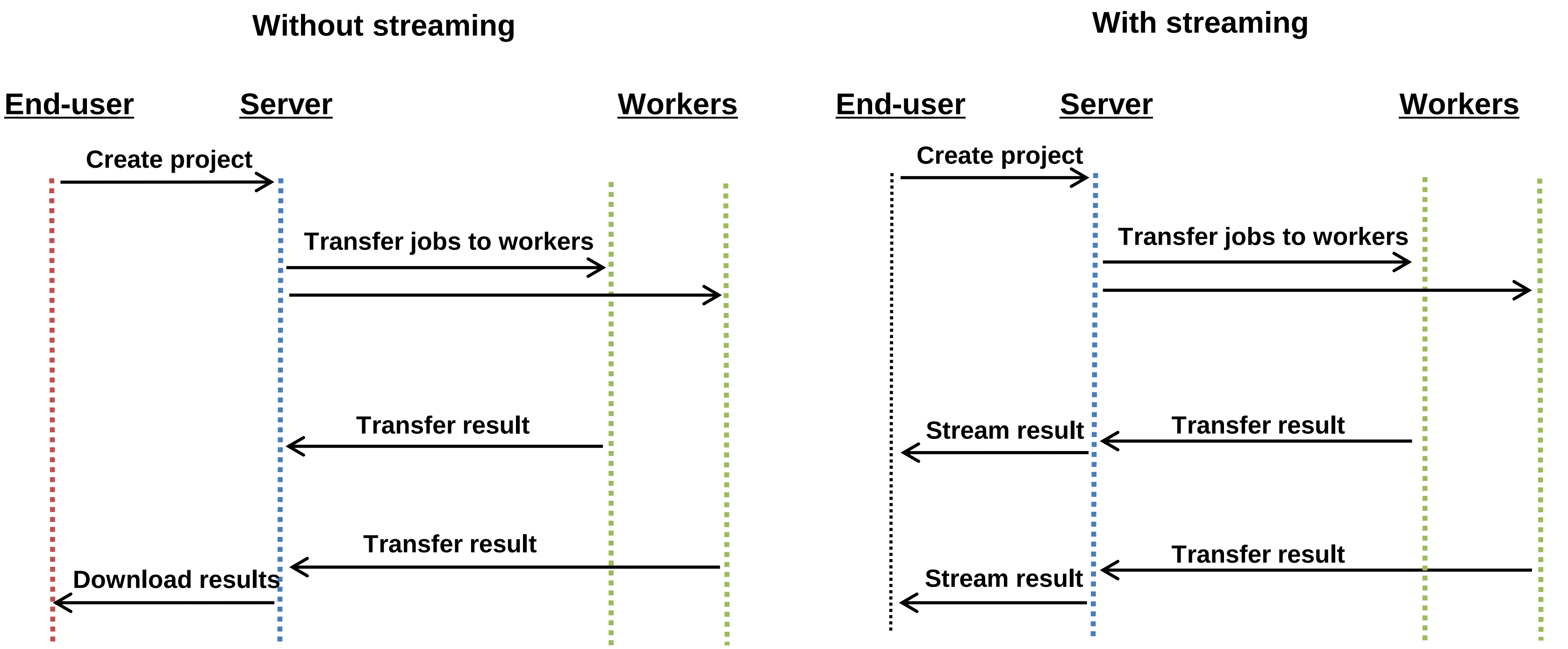
Worker Code

```
% Without snapshot
result=0
for iter=1:10000
    result=comp_intensive_function(result)
end
```

```
% With snapshot
iter=1;result=0;
% Override init values if resuming from a snapshot
loadSnapshot()
for iter=iter:10000
    result=comp_intensive_function(result)
    saveSnapshot('result','iter') % Save intermediate results
end
```

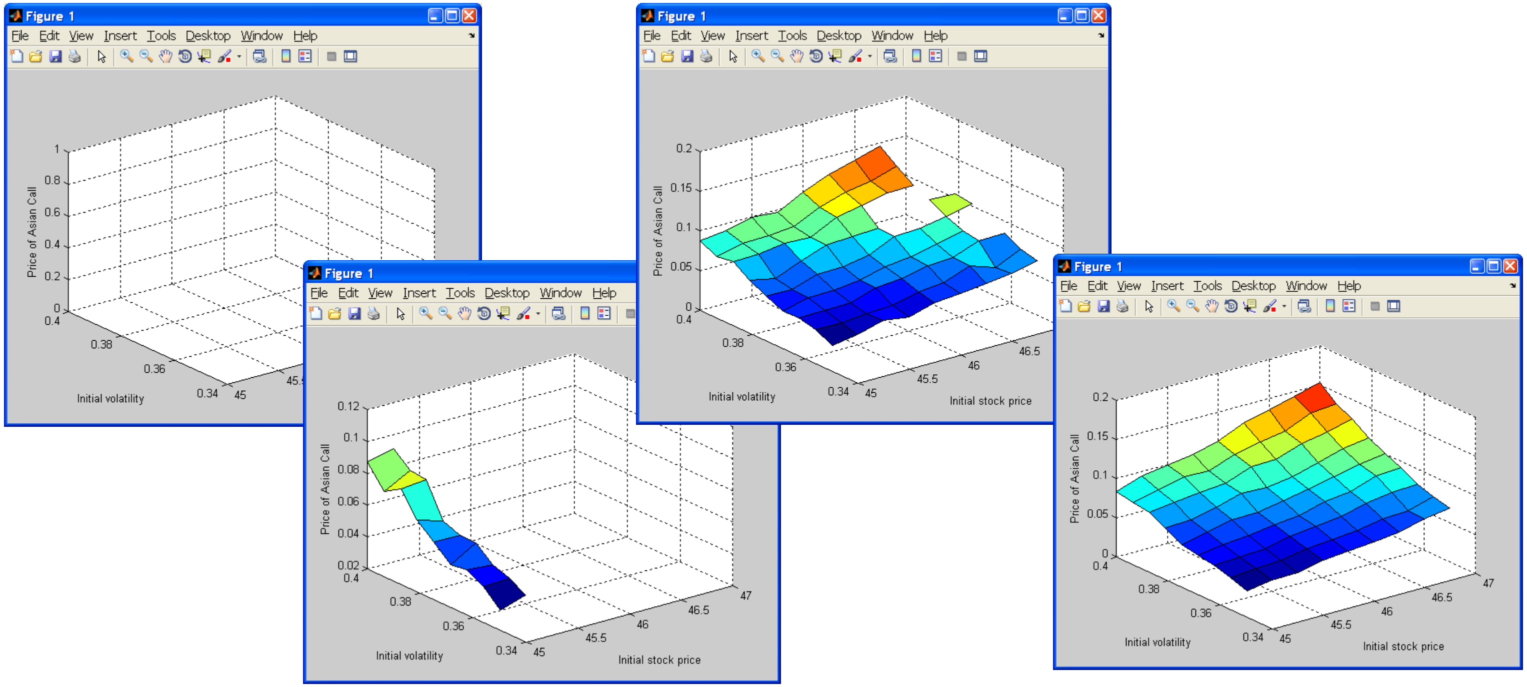
Features: Streaming

- **Transfer results from the Grid as soon as they are available**
  - Enables post-processing job results before the project is completed
  - Saves time when the results are large



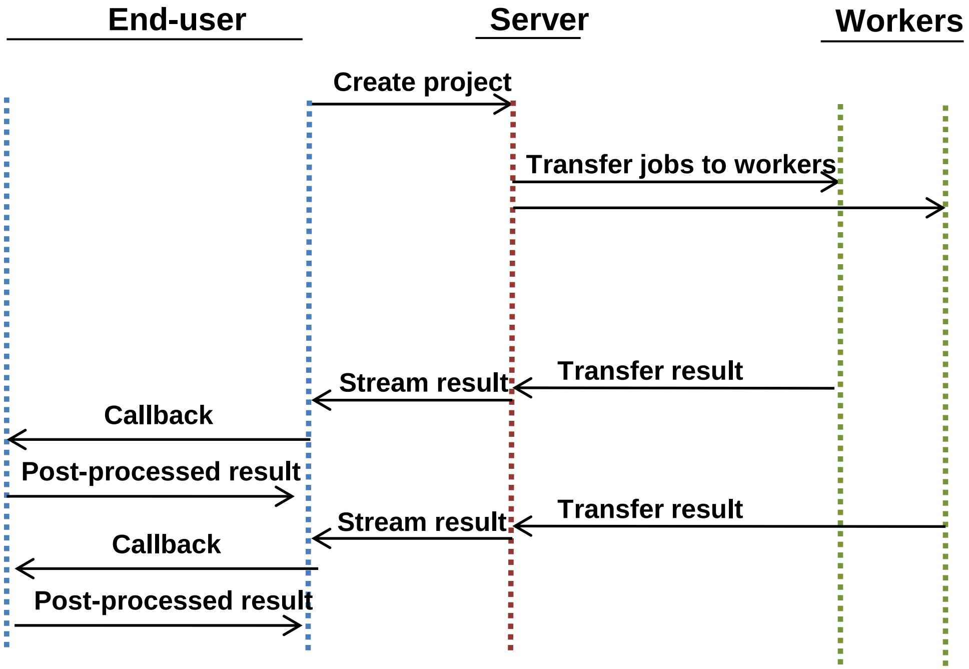
Features: Streaming

- **Transfer results from the Grid as soon as they are available**
  - Enables post-processing job results before the project is completed
  - Saves time when the results are large



Features: Callback

- **Easy way to post-process results**
  - Each job result is delivered to the callback function
  - Results from the callback function are optionally returned from peach()
- **For example:**
  - To plot the results part-by-part
  - To strip parts of the result to files and return other parts from peach()
  - To save memory
  - To manage large result files



With streaming and callback

Features: Callback

- **Easy way to post-process results**
  - Each job result is delivered to the callback function
  - Results from the callback function are optionally returned from peach()
- **For example:**
  - To plot the results part-by-part
  - To strip parts of the result to files and return other parts from peach()
  - To save memory
  - To manage large result files

Local Control Code

```
% Without callback

function result = run_example()
result = peach('funcname',{},1:jobs,)
end
```

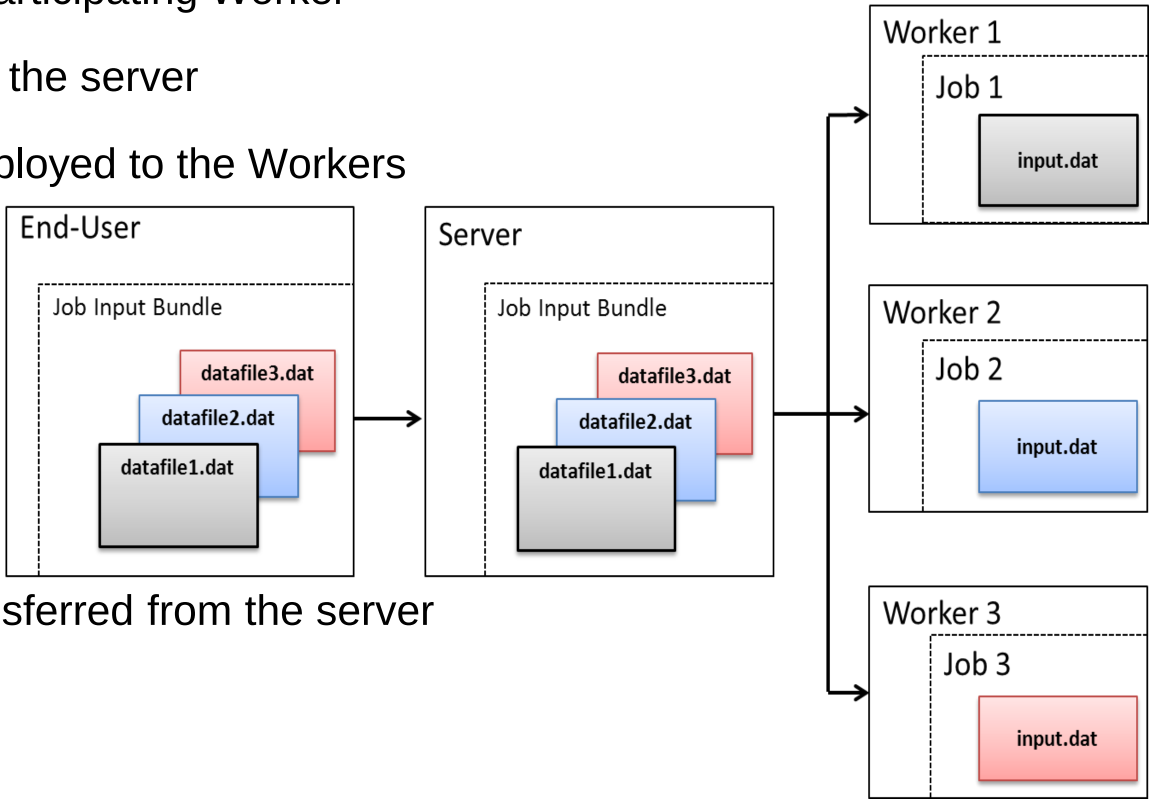
```
% With callback

function result = run_example()
result = peach(‘funcname', {}, 1:jobs, ...
     'CallbackMethod', @callback) % Name of CB function
end

function result = callback(result_file) % CB function
    result=post_process(result_file)
end
```

Features: Job Input Files

- **Enables job-specific data files**
  - Other bundles are deployed to each participating Worker
  - Job Input Bundle is transferred only to the server
  - Individual files from the Bundle are deployed to the Workers
- **Saves memory**
  - Only the part of the data needed by the job is in the inputdata
- **Saves time and network**
  - Only the part needed by the job is transferred from the server

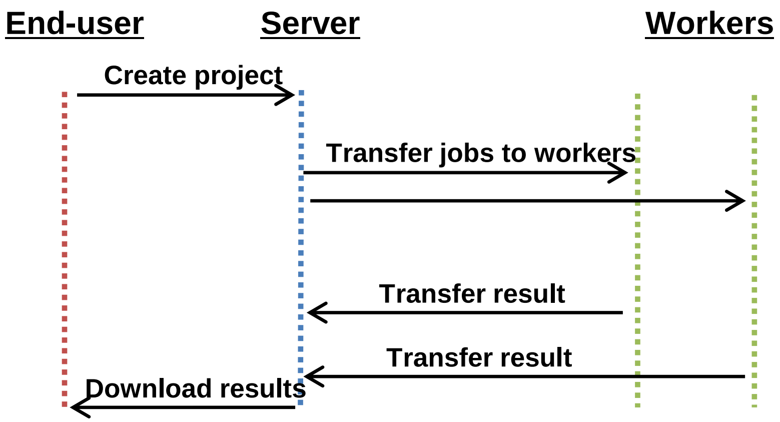


Features: Precompiled binaries

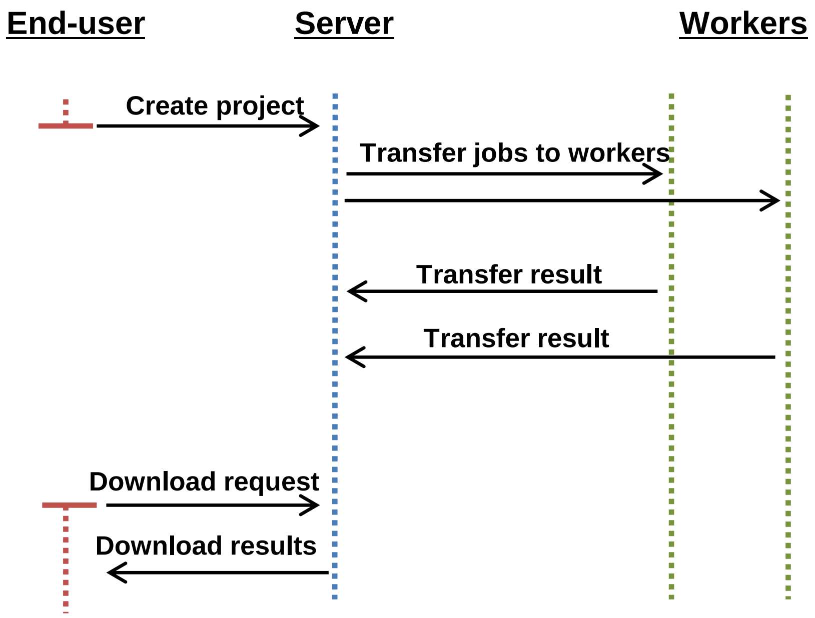
- **Using MATLAB/Perl/Command Line Interface as the front-end**
  - Compute with precompiled binary (Fortran, C/C++, etc.)
  - Optionally post-process with MATLAB/Perl
  - Easy way to execute code written in languages not having peach() (yet)
  - Possible to execute the computation on multiple platforms
    - Requires precompiled binary for each of the platforms

Features: Project Detaching

- **No need to wait for the projects**
  - Start the long project from laptop, turn off the laptop and have a nice weekend
  - Fetch the results on Monday



Normal Convention



Detached Project

Low-Level Interface

- **Peach works in most of the cases**
    - But sometimes it may not be enough
- **Possible to use low level interface**
    - To create bundles
    - To handle bundle requirement trees
    - To handle bundle parameters
    - To create projects
    - To handle project parameters
    - To create individual jobs into projects
    - To monitor the projects
    - To download the results
    - To…

WWW.TECHILA.FI