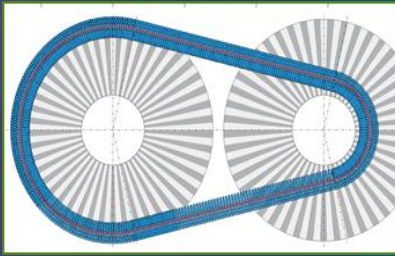


PROVIDING WORLD-WIDE INTRANET ACCESS TO PRODUCT LIFETIME CALCULATIONS USING MATLAB PRODUCTION SERVER



Dirk Twisk

Bosch Transmission Technology



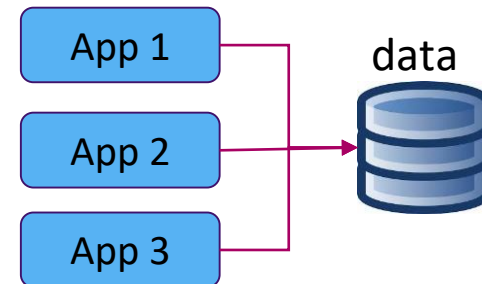
Using MATLAB Production Server for product lifetime calculations

Contents

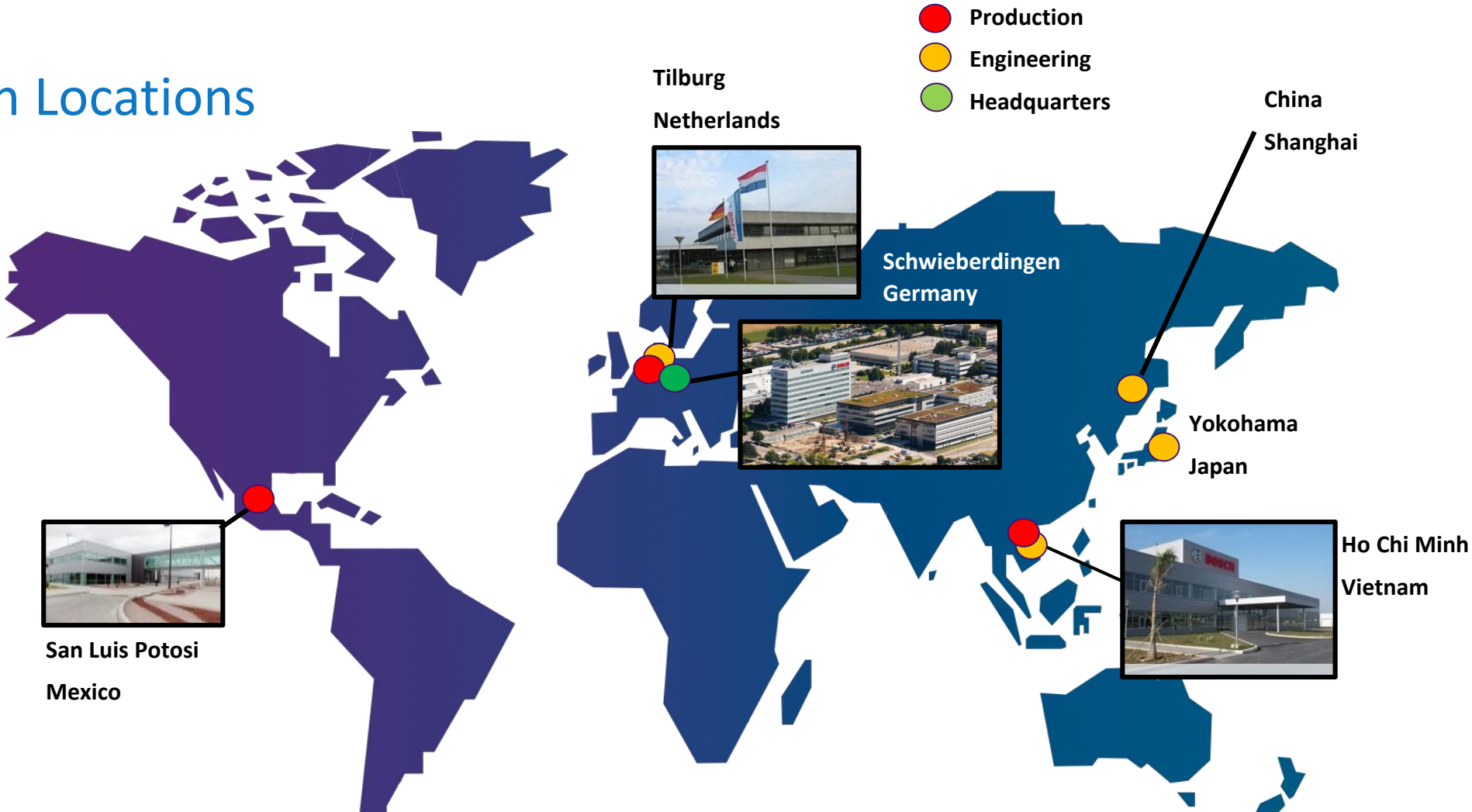
1. Bosch Transmission Technology
2. The CVT
3. Modelling
4. Implementation with MATLAB
5. Server based calculations
6. MATLAB Production Server solution
7. Next steps

Engineering calculations at Bosch Transmission Technology

- ▶ All engineers should use the same tools.
 - ▶ Many tools are developed in house – what is the latest version?
- ▶ Quality procedures are used to develop and verify software tools.
 - ▶ Use versioning systems and (unit) tests.
- ▶ Everybody uses the same data
 - ▶ Multiple channels of communication internally and externally.
- ▶ Data can be shared between software applications.
- ▶ Effective use of computational resources.



Bosch Locations



World-Wide Engineering is the key to success!

Introduction to the CVT



Model calculations for Push Belt CVT

- ▶ Designs are made by engineers, based on user requirements:
 - ▶ Maximum torque and power, expected lifetime (kms), ratio coverage, package size.
- ▶ Detailed calculation results are shared with the customer.
 - ▶ Bosch designs and produces the push-belt only.
 - ▶ All other transmissions components are made by the customer.
 - ▶ Integration engineering.
- ▶ Calculations are made during all stages of the development process.
 - ▶ Initial offering to customer
 - ▶ Product Development.
 - ▶ Verification in Laboratory.

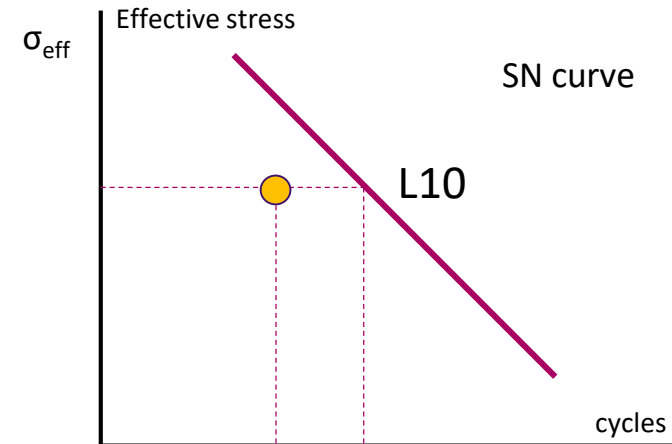
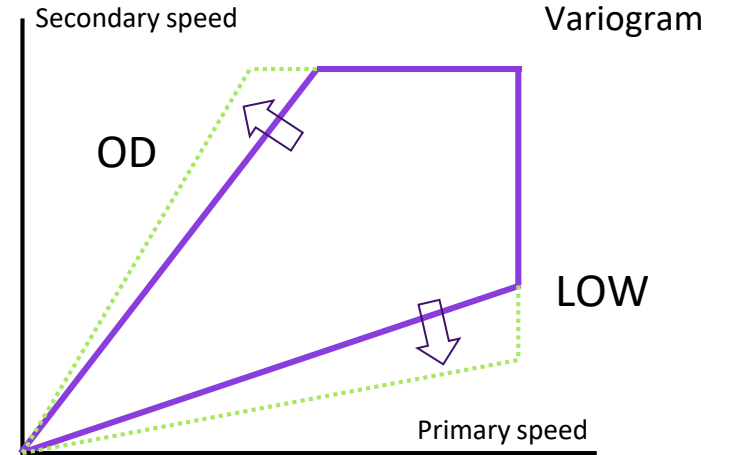
Engineering calculations

► Geometrical calculations

- Achieve a large Ratio Coverage.
 - Deep LOW is for take-off performance
 - OD is required for fuel efficiency
- Determine build size. Smaller and less weight is better.

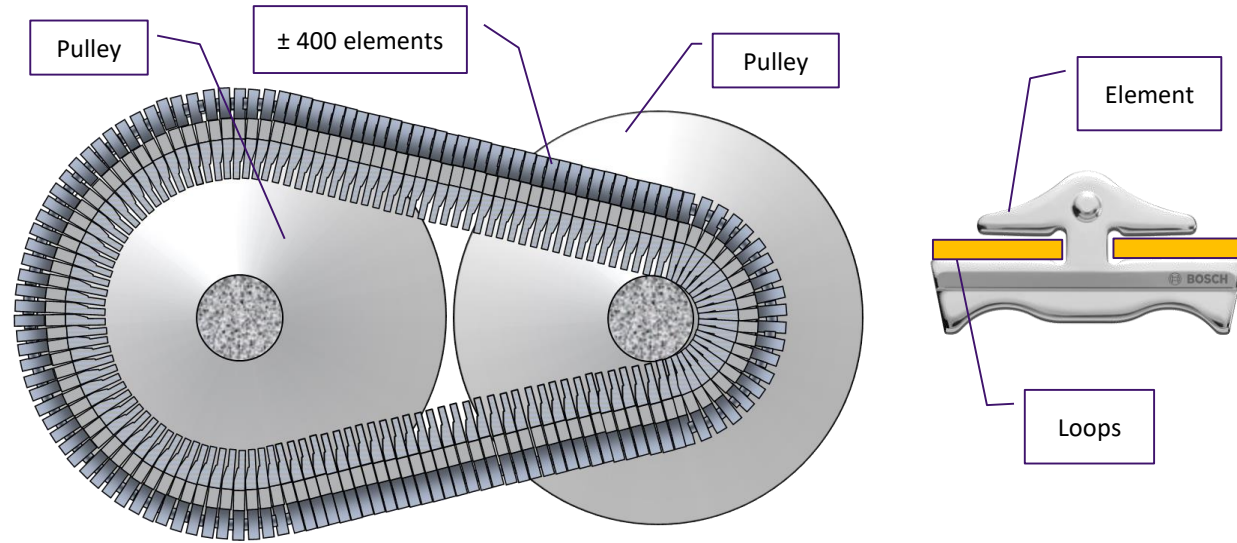
► Forces and Stresses

- Calculate forces and stresses acting on the system.
- Calculate expected lifetime of the push belt
 - Number of stress cycles before most critical part fails.

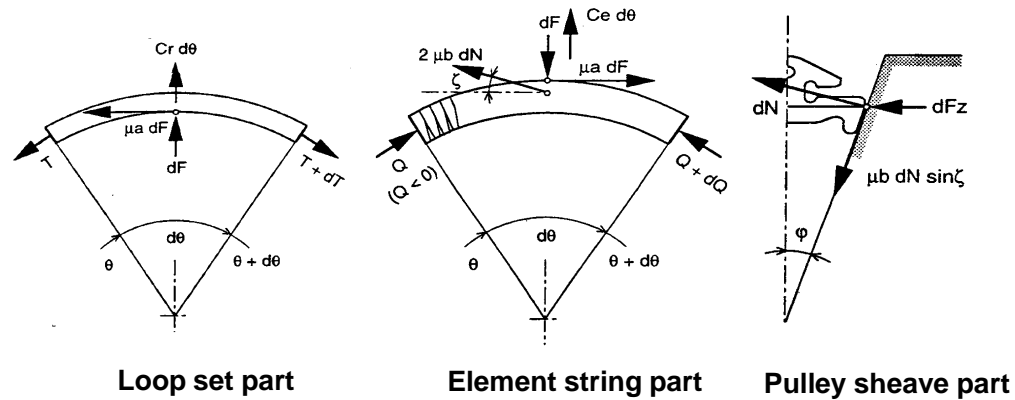


Belt Calculation Model

Physical Model



BCP model



Differential Equations

Solving the BCP problem in MATLAB

```
[R1,R2,exitflag] = fsolve(@(SolvVec) FuncName(SolvVec,ConstVec),InitVec,options);
```



- Call to fsolve
- Target function
- Differential equations

```
% -----
function [F] = Calc_Solution_Above_Transition(x,y)
    fdmax = x(1);
    fstrek = x(2);
    beta_s = x(3);
    faxis = y(1);

    F = [Fdsec(alphas,fdmax,fstrek,beta_s)
        FaxisA(fdmax,fstrek,beta_s)+FaxisB(fdmax,fstrek,beta_s)-faxis
        Msecf(fdmax,fstrek)-Ms];
end

% -----
function [F] = Calc_Solution_Below_Transition(x,y)
    fdmax = x(1);
    fstrek = x(2);
    beta_s = x(3);
    faxis = y(1);

    F = [Fdsec(alphas,0.0,fstrek,beta_s)-fdmax
        FaxisA(0.0,fstrek,beta_s)+FaxisB(0.0,fstrek,beta_s)-faxis
        Msecf(-fdmax,fstrek)-Ms];
end
```

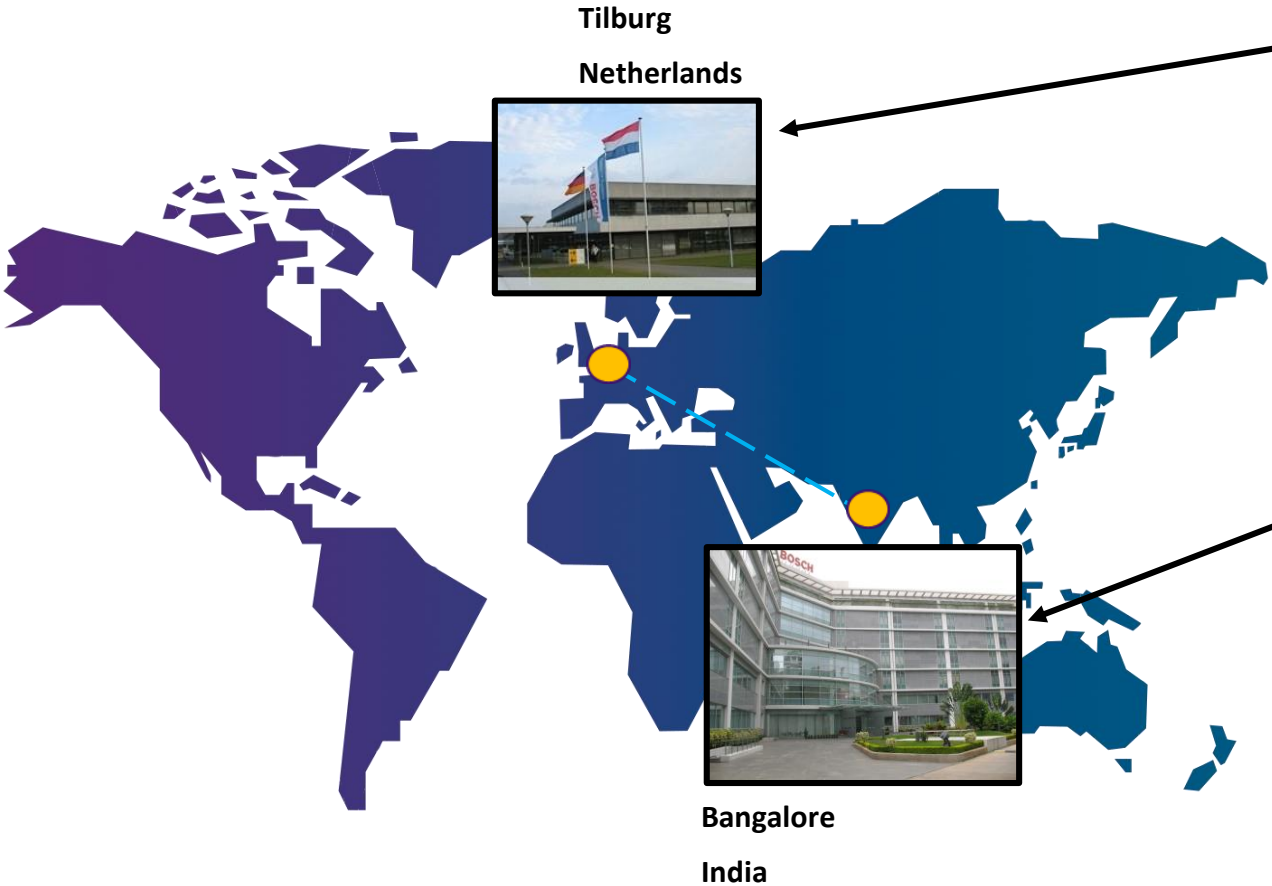


```
% -----
function [F] = FaxisB(fdmax,fstrek,b_s)
    con20 = 0.5*(1-tan(lambda_sec)*mu_lf*sin(gammalfb))./(tan(lambda_sec)+mu_lf*sin(gammalfb));
    if model == BOVEN
        F = con20*((-1/con10b)*(fdmax+Csch-(fstrek-Csn)*exp(-mu_lr*(alphas-b_s))))*(exp(cc
    else
        F = con20*((-1/con10b)*( Csch-(fstrek-Csn)*exp(-mu_lr*(alphas-b_s))))*(exp(con10
    end
end

% -----
function [F] = FaxisC(fdmax,fstrek)
    con20 = 0.5*(1-tan(lambda_pri)*mu_lf*sin(gammalfc))./(tan(lambda_pri)+mu_lf*sin(gammalfc));
    fslos = (fstrek-Csn)*exp(-mu_lr*alphas)+Csn;
    if model == BOVEN
        F = con20*((fslos-Csn-( Csch))*(alphap-alphas));
    else
        F = con20*((fslos-Csn-(fdmax+Csch))*(alphap-alphas));
    end
end

% -----
function [F] = FaxisD(fdmax,fstrek,h_n)
```

Implementation



Tilburg
Netherlands



Bosch PS-CT - Tilburg

- Mechanical Engineering
- Transmission Design (CVT)
- Modelling
- MATLAB

Bosch RBEI India - Bangalore

- Software Systems Design
- Databases
- Java
- MATLAB

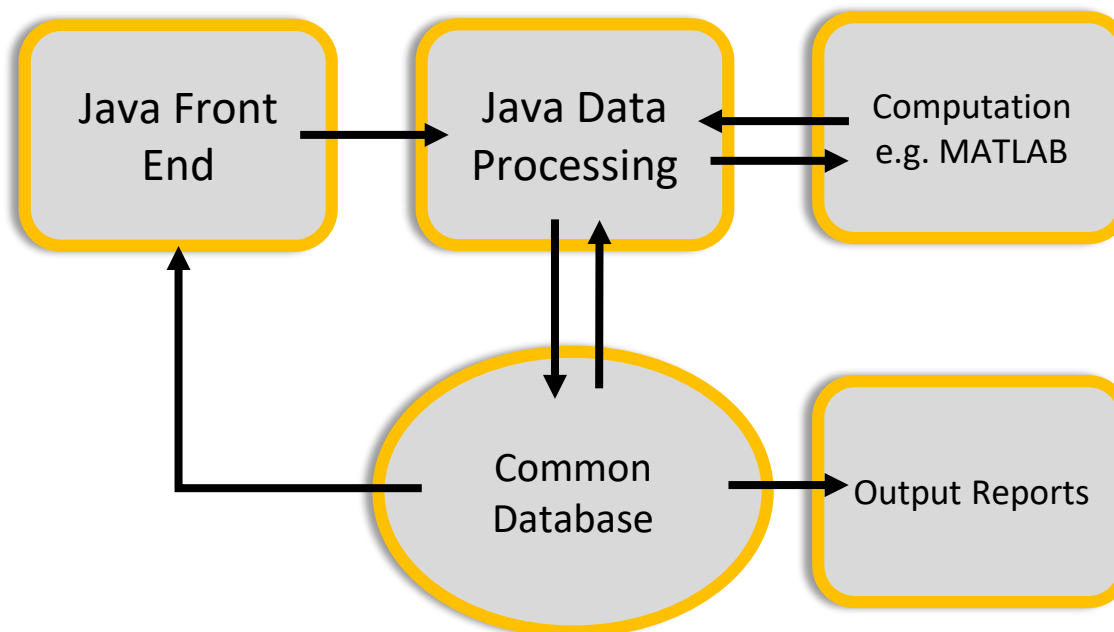
Bangalore
India



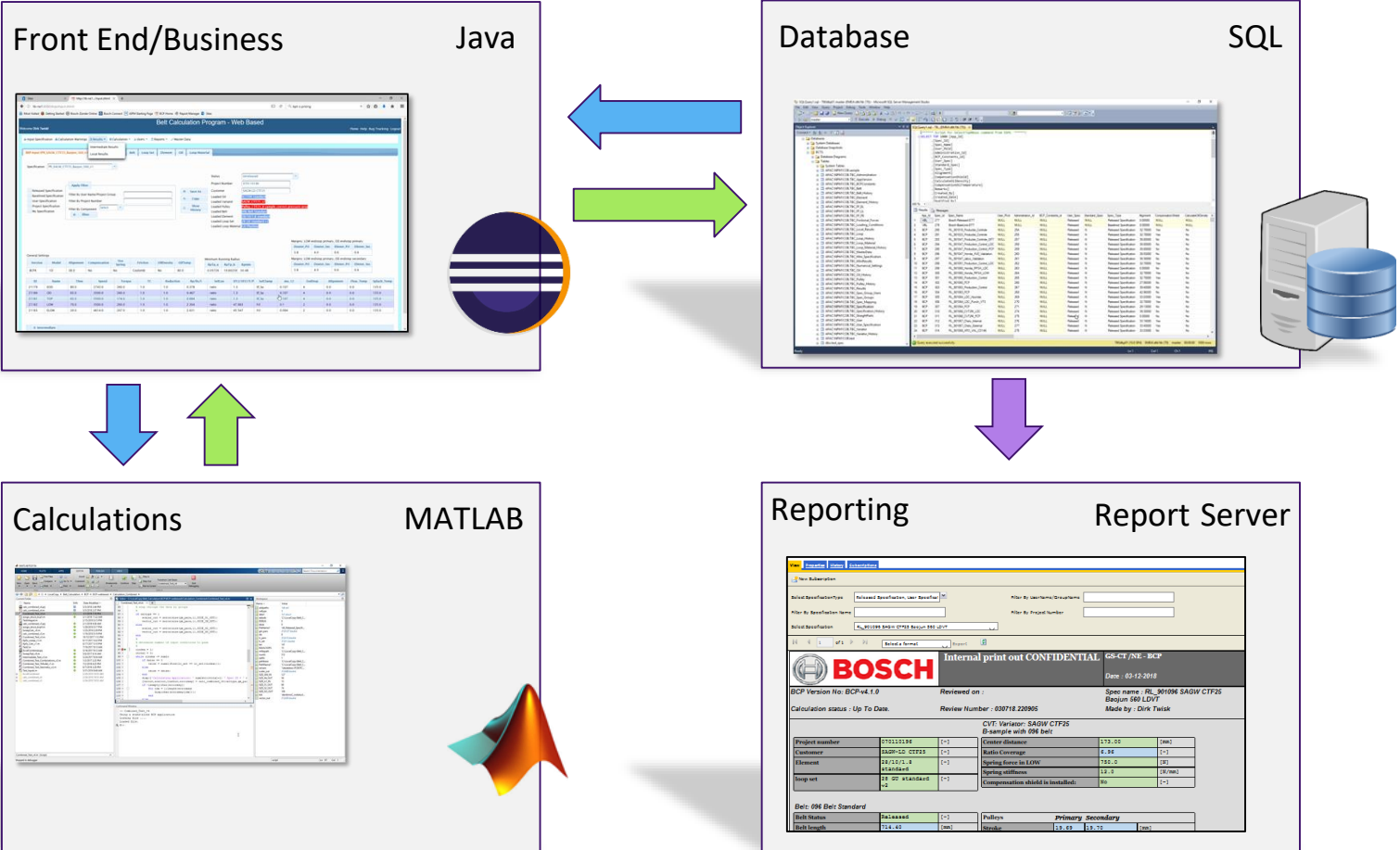
PS-CT = Power Systems – Continuous Transmission
RBEI = Robert Bosch Engineering India

System Design

- Server stores inputs and outputs for several applications
- Start with complete variator models: BCP and VBL
- Data can be accessed by all users (engineers)
- Different interfaces and business logic for different applications.



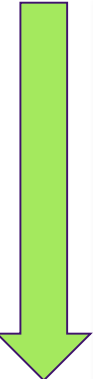

System Implementation



Development – traditional design



Bosch Tilburg

Product Development
Model Development

- 
1. Write Code
 2. Test Code
 3. Compile Code
 4. Jar File(s)
- 

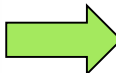
Bosch India

Software Development
Information Systems

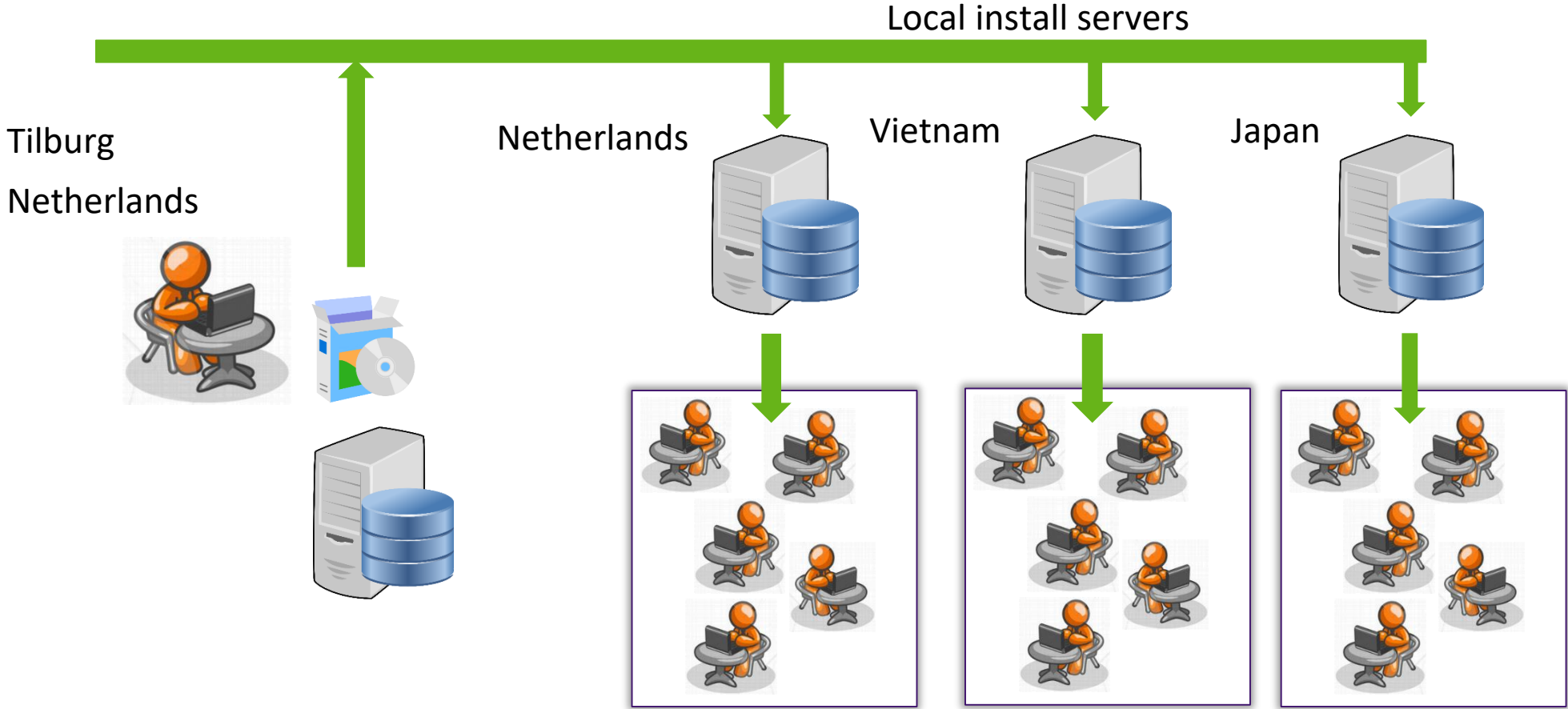
- 
1. Write Code
 2. Test Code
 3. Compile Code
 4. Create Executable
- 

Tilburg IT department

Maintain computer systems
Install Software

- 
1. Upload to Software Server.
 2. Install JAR on users' PC.
 3. [Install MATLAB Runtime].

Local installation



Drawbacks of the traditional design

Complex release process:

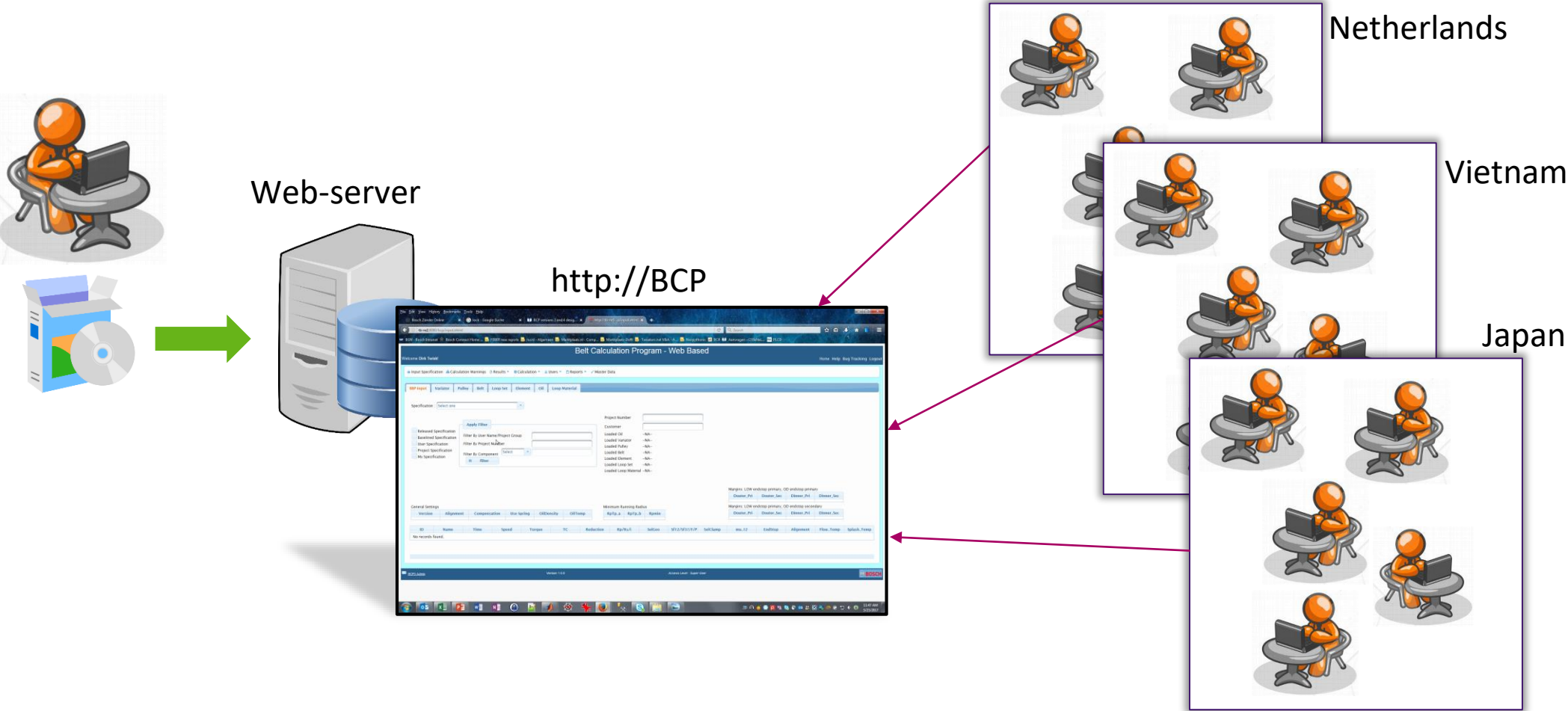
- Updates not synchronized due to the use of different install servers.
- Every user needs to have (correct) MATLAB Runtime installed.
- High maintenance costs (technical support).
- Sending data over the network is slow, so the performance is not acceptable.
- Changes to the MATLAB code requires rebuild of the complete code.

After the second major release, it was decided to change the design



Switch to a web-based approach

Web-based Approach

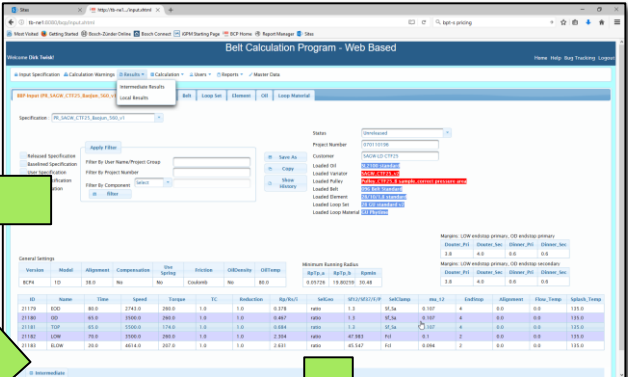
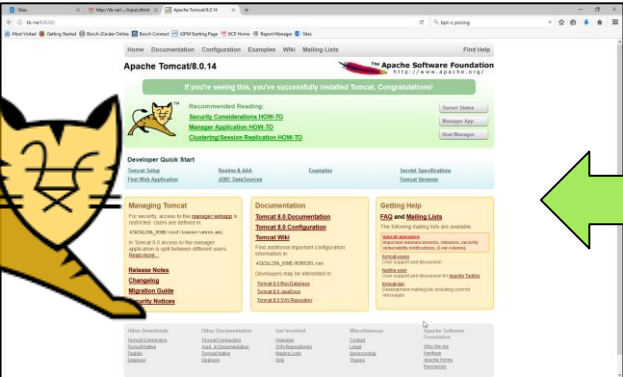


Improved Solution

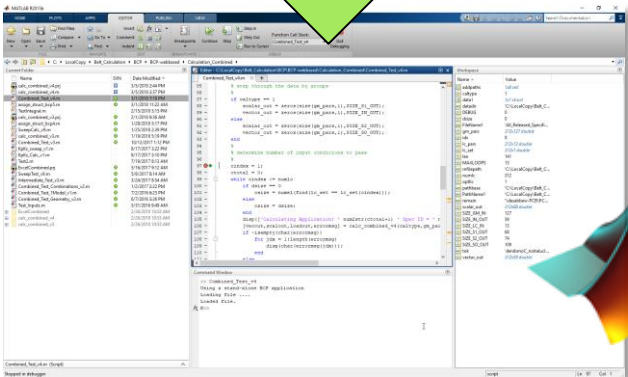
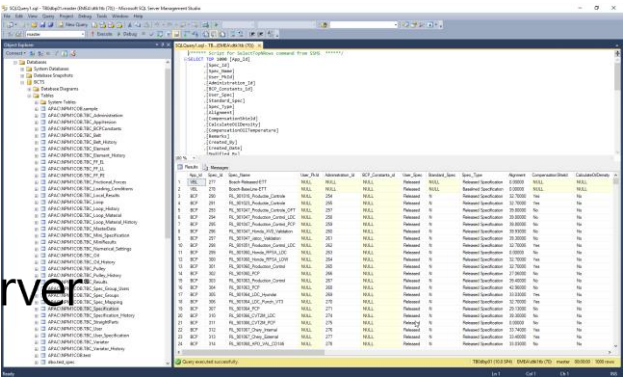
- ▶ Make the application web-based:
 - ▶ All required components run on a single server.
 - ▶ Only inputs and outputs are sent over the network.
- ▶ Users do not have to install any software on local computers
 - ▶ The database itself is used to store user data and grant access to users.
 - ▶ Use NT login credentials
- ▶ Updating software is required at only 1 location
 - ▶ Synchronized updates of software and database.
 - ▶ Minimal disruption of service.
 - ▶ Maximum of 1 hour downtime for major updates.

Architecture

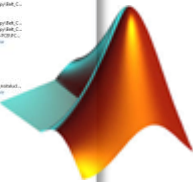
Tomcat



BCP Java applet

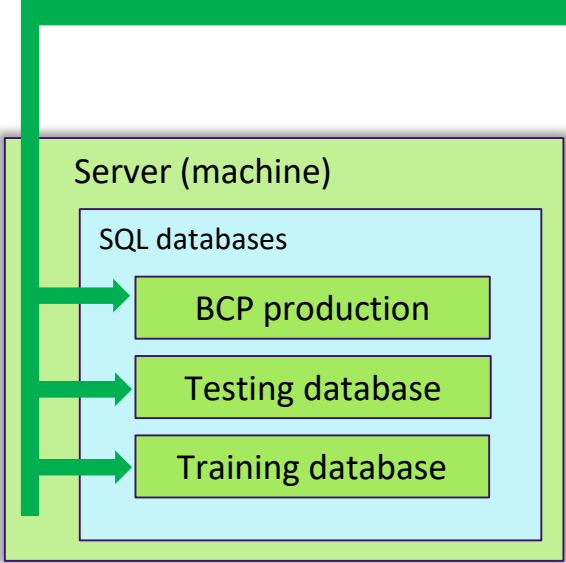


MATLAB
Compiled application

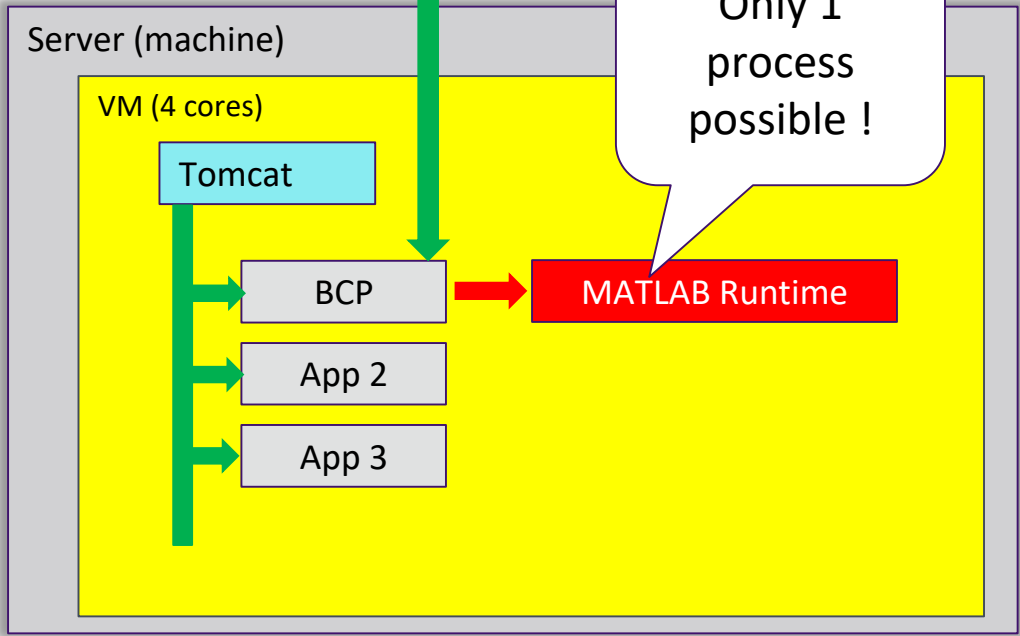


Initial Implementation

network

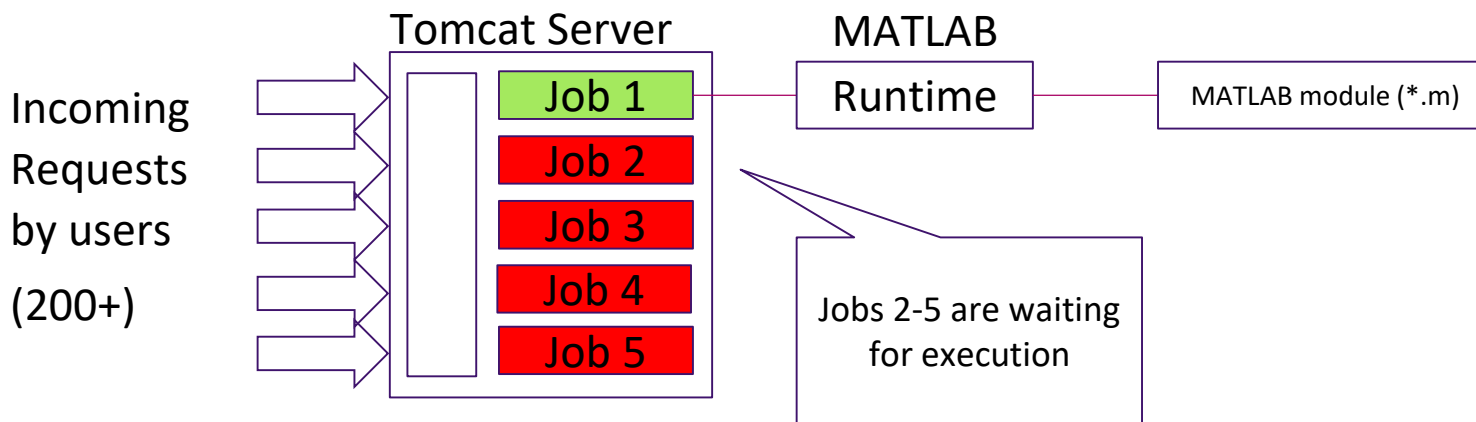


Tilburg - NL



Tilburg - NL

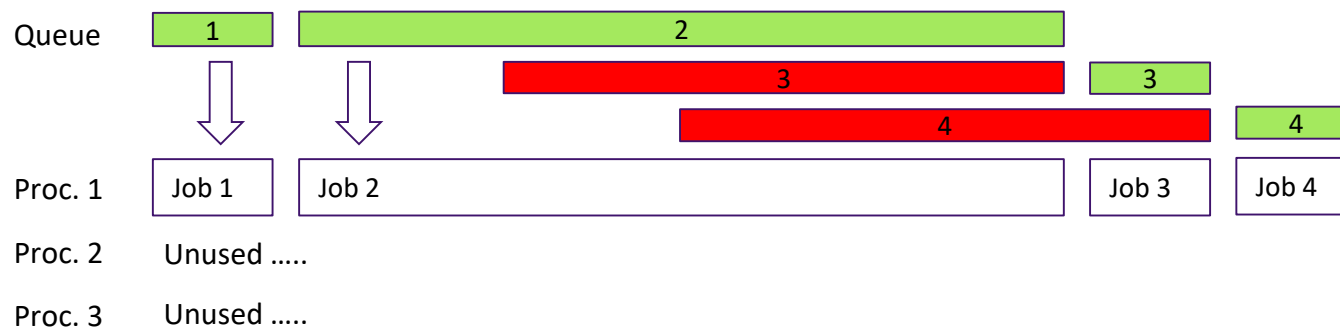
Limitations ...



Calculation Times:

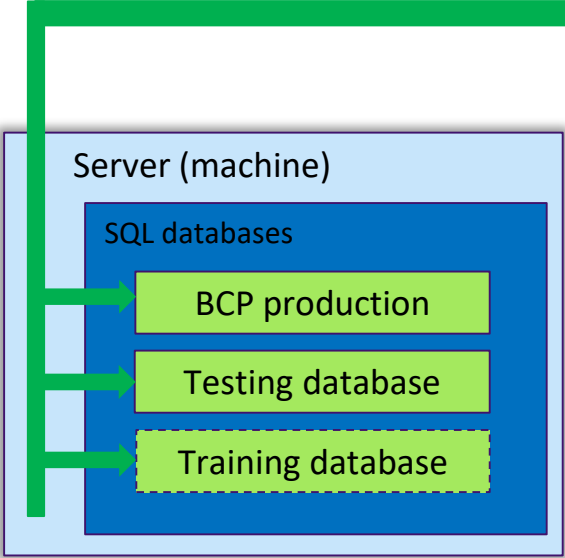
Shortest : 2 seconds

Longest : 24 hours



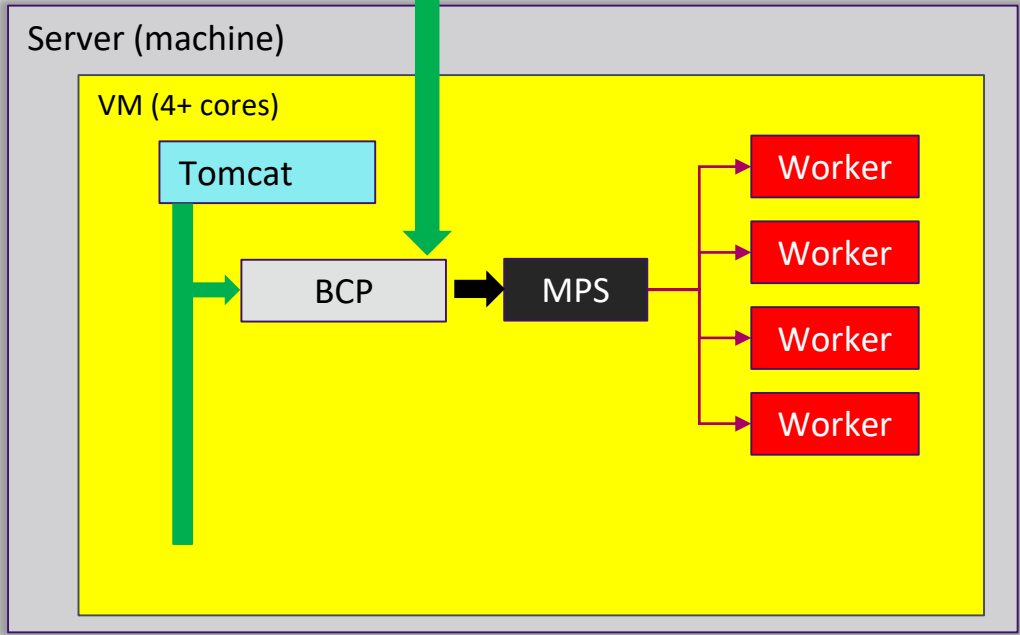
Infrastructure with Production Server – Final Design

network



Tilburg - NL

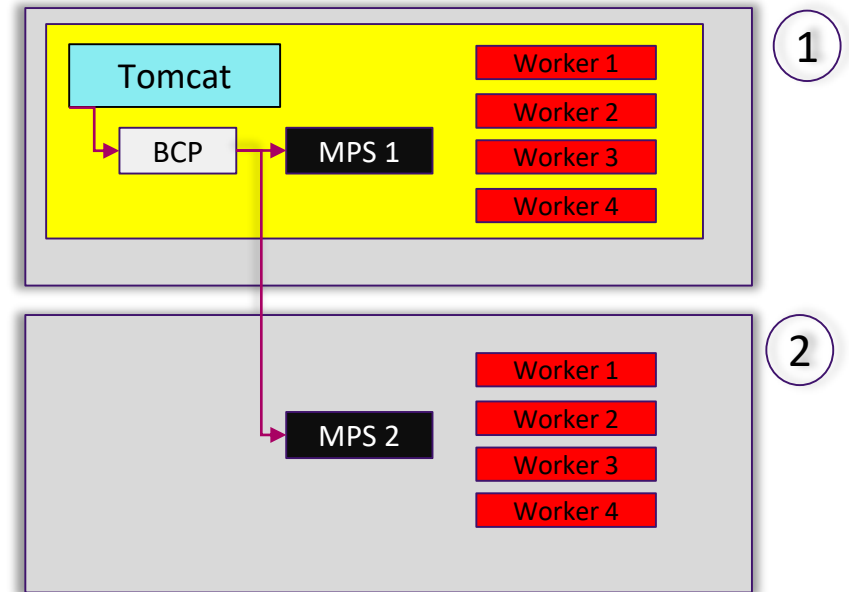
MPS Matlab Production Server



Tilburg - NL

Main Benefits of the Matlab Production Server

- ▶ Multiple MATLAB jobs can be run simultaneously:
 - ▶ Each up to 24 hours per job (vehicle duty cycles).
- ▶ Scalable:
 - ▶ Add more cores to the server.
 - ▶ Assign different machines for different tasks.
 - ▶ Utilize available (unused) computational resources.
- ▶ Decoupling of main BCP code and MATLAB code.
 - ▶ Define inputs/outputs between systems.
 - ▶ Independent updates of Java and MATLAB code.
 - ▶ Log output is used to verify communication between the components.



Next Possible Steps

1. Make existing standard programs web-based:
 - Model-Viewer-Controller type applications.
2. Create a library of common functions, allow calling from:
 - Java, Python, C# (web)
 - Excel
 - MATLAB
3. Provide functionality to non-MATLAB departments
 - Production, Quality, Inspection, ...
4. Consider to allow customers to perform calculations
 - Eliminate iteration loops with customer
 - No access to model and data (IP)

