University of Stavanger

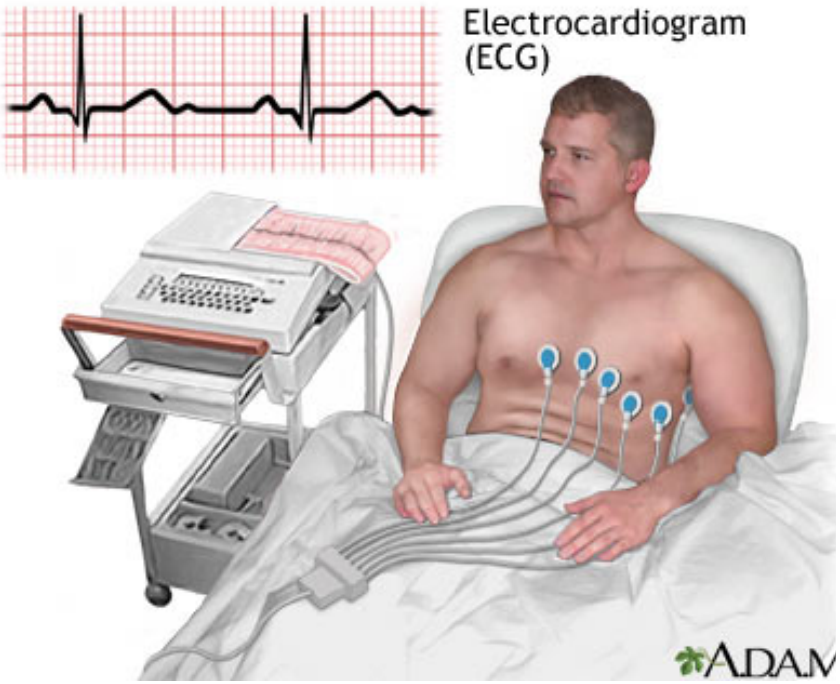
Parallel-PSM
Parallelizing the Population Stochastic Modeling R Package with UNICORE

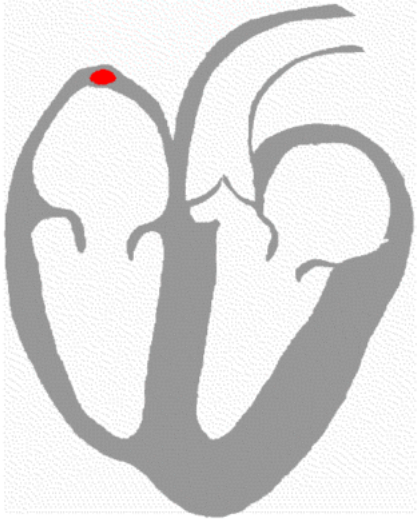
Abdulrahman Azab

abdulrahman.azab@uis.no

University of Stavanger

ECG - Electrocardiogram





University of Stavanger

CPR - cardiopulmonary resuscitation

University of Stavanger

Start of the story…

University of Stavanger

Modelling the relationship between ECG characteristics and CPR quality during cardiac arrest

Kenneth Gundersen

Jan Terje Kvaløy

University of Stavanger

Introduction

- Measurements easily available during out-of-hospital resuscitation include:
  - Electrocardiogram (ECG)
  - End-Tidal CO_2 (ETCO2)
  - **Compression depth** and **force**
  - Thoracic impedance

  } Response

  } Intervention

- Overall aim: a mathematical model for the interaction between intervention and response during VF/VT.
- This work: develop minimal model for CPR quality variables and ECG coarseness (response).

University of Stavanger

Introduction

- This type of model can possibly be used to:
  - Identify the CPR quality variables that indicate its effectiveness.
  - Test different compression techniques against each other.
  - Predict the patiens' response to further CPR and thereby to **chose between immediate defibrillation or further CPR.**
  - General framework for testing different hypothesis.

ECG coarseness – median slope

- The ROSC-predictor median-slope (MS) has high prediction accuracy and is easily computed:

$$MS = \text{median}\left(\left|ecg(n) - ecg(n-1)\right|\right), n = 1, \ldots, N$$

- MS combines amplitude and frequency information -> high ECG amplitude and frequency (coarse ECG) indicates a good state of patient.

University of Stavanger

ECG coarseness – median slope

Normal

Heart block

University of Stavanger

Data

- Observational data from out-of hospital cardiac arrest episodes.
- 85 patients in baseline data, 27 patients contributed with a total of 118 VF/VT sequences.

University of Stavanger

Challenges

- Model fitting extremely computationally intensive.
- Hetereogenous patients -> random effects on many parameters -> many model parameters to fit.
- Limited size of available dataset.
- Noise in ECG recordings (must be manually identified and censored).

University of Stavanger

Future research

- **Speed up model fitting (modify software to enable parallell processing).**
- **Obtain (larger) dataset where ECG noise is minimized.**
- Include $ETCO_2$ in an extended model.
- Consider excluding ”hopeless” patients from analysis.
- Take a step back and develop (idealized) model from animal data?

University of Stavanger

In brief…

University of Stavanger

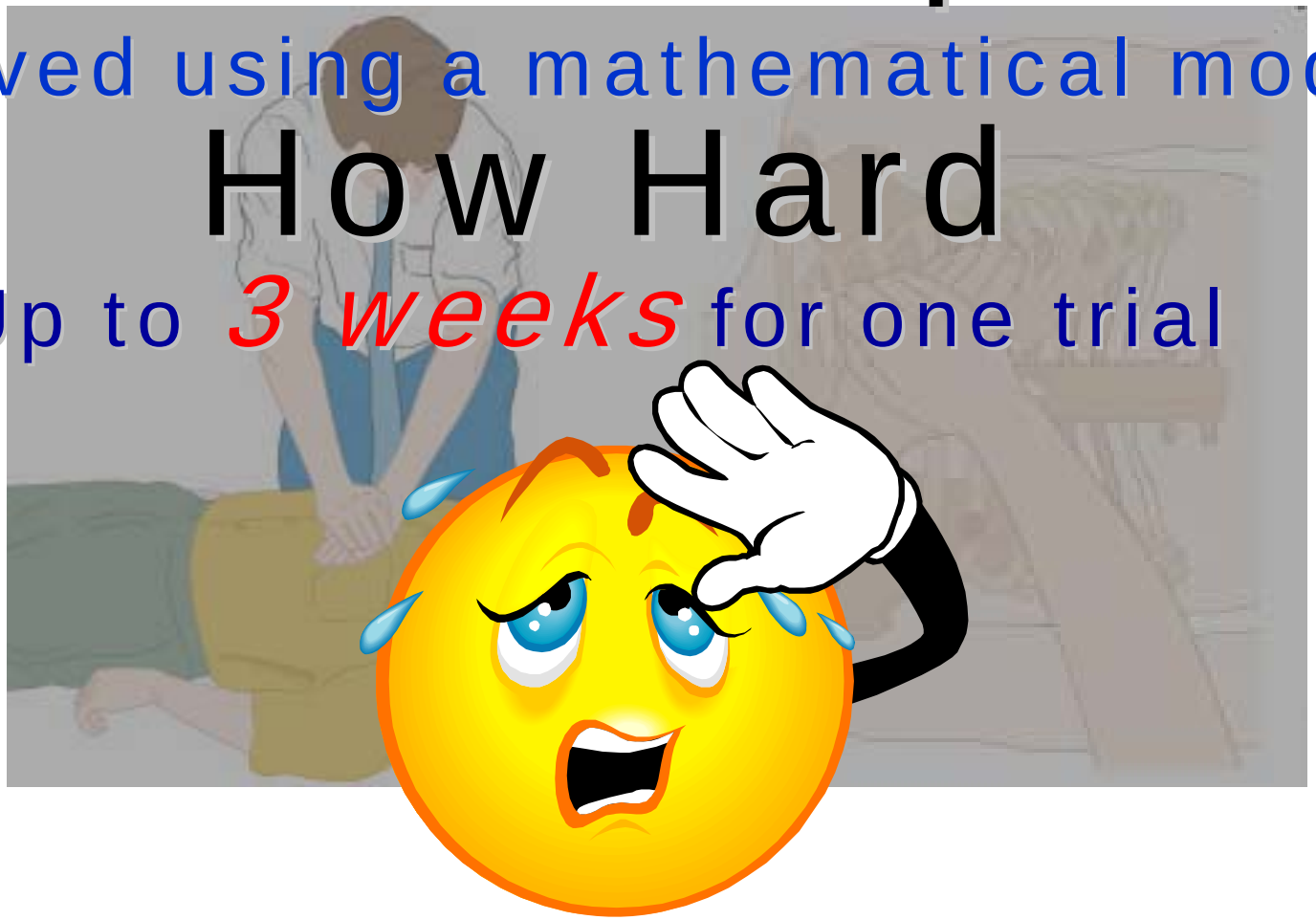
In brief…

How Deep

Solved using a mathematical model

How Hard

Up to *3 weeks* for one trial



University of Stavanger

And then…

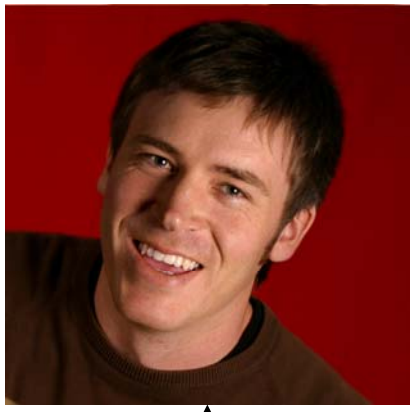
University of Stavanger

Scenario…

Hein,
can you try to do it?

University of Stavanger

Scenario…



Hein,
can you try to do it?

CC:

Azab,
See what you can do?

University of Stavanger

Scenario…

bla bla bla…

Hmmmm…

University of Stavanger

Team

Kenneth Gundersen
Post Doc
UiS

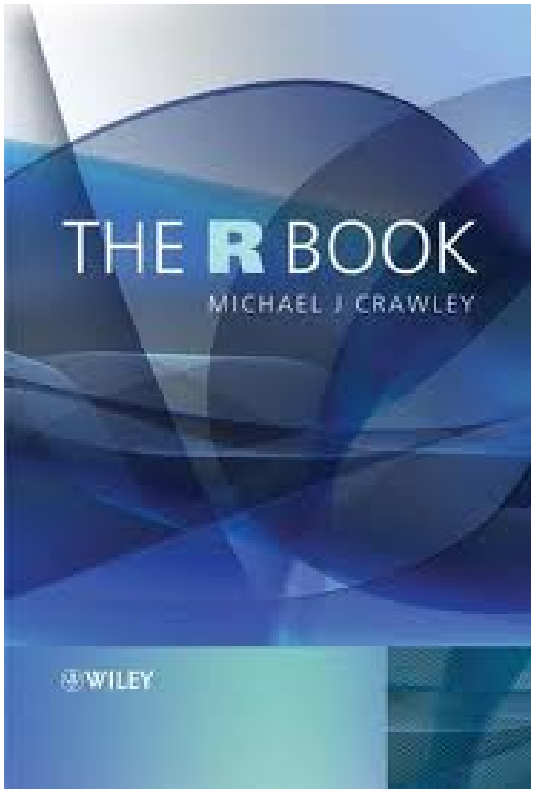
Hein Meling
Asc. Professor
UiS

Abdulrahman Azab
Research Fellow
UiS

University of Stavanger

Scenario…

Learning R



University of Stavanger

The *R* language

What is R?

- A 'language and environment for statistical computing and graphics' (implements a dialect of the language 'S')
- Syntax is C-like, but philosophy is functional
- Focus on matrices and vectors
- Free, open-source (GPL)
- Well documented
- Command line based, but there are GUIs
- Latest version: 2.13.0 (April 2011)
- URL: `http://www.r-project.org/`

University of Stavanger

The *R* language

R Features

- Very many analysis methods available
- Scriptable and extensible
- Can be used interactively, process batch jobs or run as a script
- Bindings to many other systems/languages, e.g., Python, Perl, Matlab, *SQL, Excel
- Active user community with thousands of contributed packages

University of Stavanger

The *R* language

Elementary Data Types

Basic elements: *numbers*, *strings*, *logicals*

```
> 42
[1] 42
> "a string"
[1] "a string"
> TRUE
[1] TRUE

> 42 + 13.5
[1] 55.5
> 42 > 13.5
[1] TRUE
> substr("a string", 3, 5)
[1] "str"
> ! TRUE
[1] FALSE
```

University of Stavanger

The *R* language

Compound Data Types: Vectors

- Basic elements collected in *vectors*, *lists*, *matrices* and *data frames*
- Collection of *equal* types of elements: *vector*:

```
> 1:5
[1] 1 2 3 4 5
> c(42, 1:5)   # "c" for "concatenate"
[1] 42  1  2  3  4  5
> c("three", "small", "things")
[1] "three"  "small"  "things"
```

- Indexing:

```
> nums <- c(42, 33, 58, 1, 3.2)
> nums[2]
[1] 33
> nums[2:3]
[1] 33 58
> nums[c(1,3)]
[1] 42 58
```

University of Stavanger

The *R* language

Compound Data Types: Lists

- Collection of *different* types of elements: *list*:

```
> c(42, "Mary")       # Probably not what you want
[1] "42"   "Mary"
> list(42, "Mary")    # That's more like it!
[[1]]
[1] 42

[[2]]
[1] "Mary"
```

- Indexing: [] and [[]]:

```
> lag <- list(c(3,5), "string", rep(TRUE, 5))
> lag
> lag[2:3]    # sub list
> lag[3]      # still sub list
> lag[[3]]    # element
```

University of Stavanger

The *R* language

Compound Data Types: Matrices

- Two-dimensional collections: *matrices* and *data frames*
- Same element type (usually number): *matrix*:

```
> A <- matrix(1:9, ncol = 3, nrow = 3)
> A
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> B <- matrix(6:1, ncol = 2, nrow = 3)
> A %*% B
     [,1] [,2]
[1,]   54   18
[2,]   69   24
[3,]   84   30
```

- Indexing: `A[2,1]`

University of Stavanger

The *R* language

Variables

- Values can be stored in *variables*:

```
> mynum <- 42
> mynum + 13.5
[1] 55.5
> adj <- "interesting"
> sentence <- paste("Very", adj)
> sentence
[1] "Very interesting"
```

- Tip: use descriptive names; avoid single-character names.
- List all variables: `ls()`
- Show value of variable: `mynum` or `print(mynum)`
- Remove a variable: `rm()`, e.g., `rm(adj)`

University of Stavanger

The *R* language

Help!

This is probably the most important slide!

- `?mean` - help for a function
- `help.search("regression")` or simply `??regression` - search in your installed R
- `RSiteSearch("logistic")` - search the R web site
- `demo()` - list/run demos
- `vignette()` - list package vignettes
- `help.start()` - start help centre

University of Stavanger

The *R* language

http://www.notur.no/notur2011/material/R_course_notur2011_slides.pdf

University of Stavanger

Back to the problem…

University of Stavanger

ECG – CPR modeling code (by Kenneth)

1. Create the model

   NLmod.MS ← ***Model***

2. Generate the simulated heart block reading

   simdatar <- PSM.simulate(NLmod.MS,...)

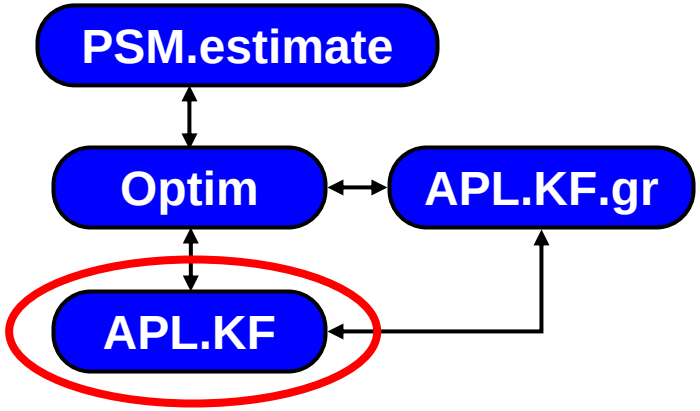
3. Estimate the optimum compression force and depth

   fit_simmod <-
   PSM.estimate(NLmod.MS,simdatar,...,maxit=1000,...)

30 minutes per iteration on a single machine

University of Stavanger

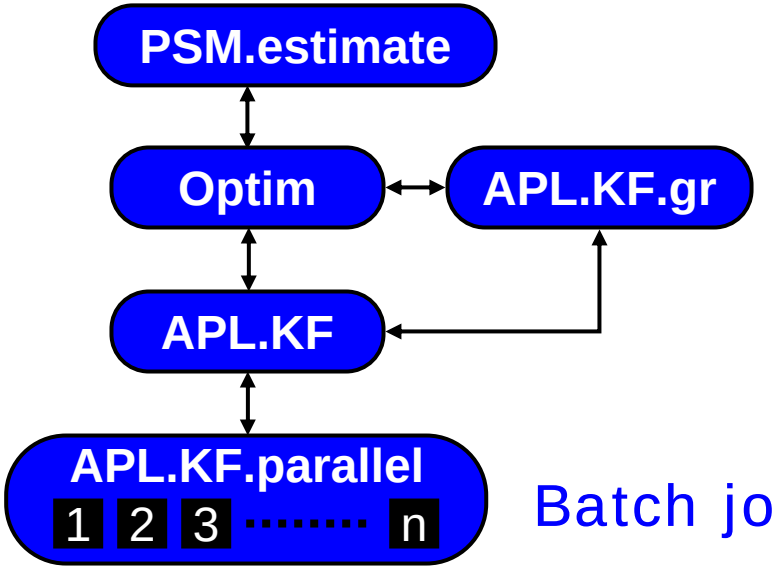
ECG – CPR modeling code (by Kenneth)



Very computational intensive, but parallelizable

University of Stavanger

ECG – CPR modeling code (by Kenneth)



University of Stavanger

Challenges

- Parallel packages for R, e.g. snow and GridR, are either not dependable or faulty.
- R doesn't directly support java. Cannot use HiLA.
- The computation is iterative, cannot be shaped as a workflow.
- It is time consuming to create a new parallel R package, *and I want to finish my PhD*.

University of Stavanger

Thinking…

Azab,
Why don't you implement a file-system interface

University of Stavanger

Thinking…

Azab,
Why don't you implement a file-system interface

Let's try and see..

Do you think it is a good solution?

University of Stavanger

Stroll: Job submission by file-system **write()**

```
#Create the job directory
taskDir<-dir.create("./Parallel.APL.KF")

# Set the job configuration
write("exec=job.R;in=in$(i).RData;target=unicore", file
    = "./config")
```

...Generate input files...

```
# Submit the job
write("SUBMIT -W", file = "./control")
```

...Collect output files...

University of Stavanger

It works

University of Stavanger

Stroll: Job submission by file-system **write("SUBMIT")** using Condor

```
> source("simmod_Grid.R")
* Non-linear model (Grid)*
Using logit transformation of parameters
Grid: Starting switch statement
Using Optimizer:        optim
optim: Start
optim: functions defined
optim: 1
APL.KF: Start  dimS =  40 , Time: 0.19
Grid: input copied to 40 Workers: elapsed time =  0.91
1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 ,11 ,12 ,13 ,14 ,15 ,16 ,17 ,18 ,19 ,20 ,21 ,22 ,23 ,24 ,25 ,26 ,27 ,28 ,29 ,30
     ,31 ,32 ,33 ,34 ,35 ,36 ,37 ,38 ,39 ,40 ,
Submitting job(s).....................................
Logging submit event(s).....................................
40 job(s) submitted to cluster 624.
Waiting for results from workers....
3 ,9 ,5 ,7 ,6 ,11 ,12 ,4 ,8 ,13 ,17 ,10 ,14 ,15 ,2 ,16 ,18 ,19 ,20 ,21 ,22 ,23 ,24 ,25 ,1 ,26 ,27 ,28 ,29 ,30
     ,31 ,32 ,33 ,34 ,36 ,37 ,38 ,39 ,40 ,35 ,
APL.KF: Ends  , Time: 150.5
                a
  -logL  = -45.2597115   (2:30.31)
optim: 2
Grid: switch statement ended
[1] Runtime:  2:30.56
Collecting average data colums..
```

University of Stavanger

Stroll: Job submission by file-system **write("SUBMIT -W")** using UNICORE

```
> source("simmod_Grid.R")
* Non-linear model (Grid)*
Using logit transformation of parameters
Grid: Starting switch statement
Using Optimizer:        optim
optim: Start
optim: functions defined
optim: 1
APL.KF: Start  dimS =  40 , Time: 0.07
Grid: input copied to 40 Workers: elapsed time =  4.21
1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 ,11 ,12 ,13 ,14 ,15 ,16 ,17 ,18 ,19 ,20 ,21 ,22 ,23 ,24 ,25 ,26 ,27 ,28 ,29 ,30
     ,31 ,32 ,33 ,34 ,35 ,36 ,37 ,38 ,39 ,40 ,
Waiting for results from workers....
1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 ,11 ,12 ,13 ,14 ,15 ,16 ,17 ,18 ,19 ,20 ,21 ,22 ,23 ,24 ,25 ,26 ,27 ,28 ,29 ,30
     ,31 ,32 ,33 ,34 ,35 ,36 ,37 ,38 ,39 ,40 ,
APL.KF: Ends  , Time: 294.01
                a
  -logL  = -45.2597115   (4:53.97)
optim: 2
Grid: switch statement ended
[1] Runtime:  4:54.04
Collecting average data colums..
```

University of Stavanger

Execution Scenario

Me

Start

End

Stroll

My machine

Lab E-462 UiS

Grid

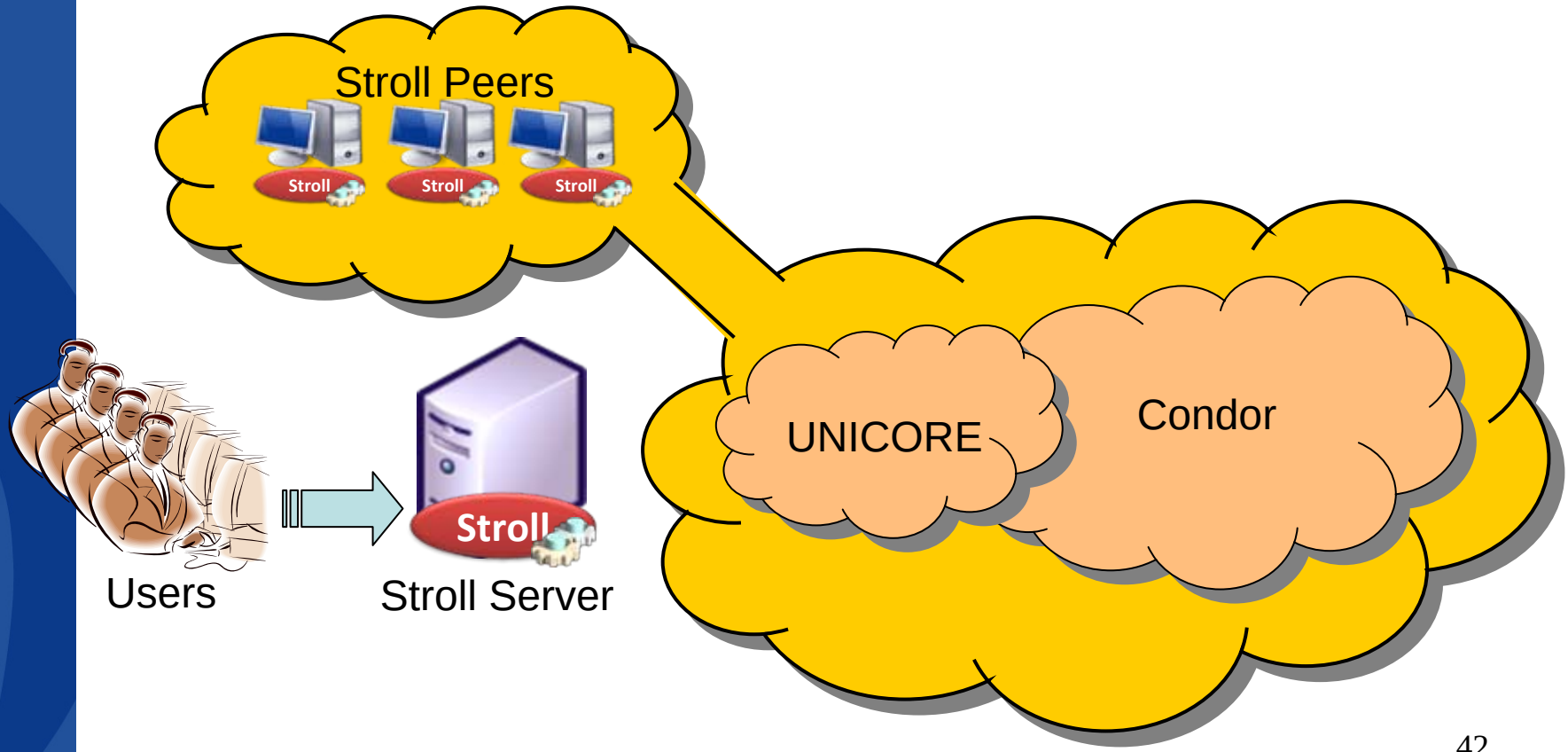
20 PCs

From 3 weeks
to
60 hours
Speedup = 8.5

University of Stavanger

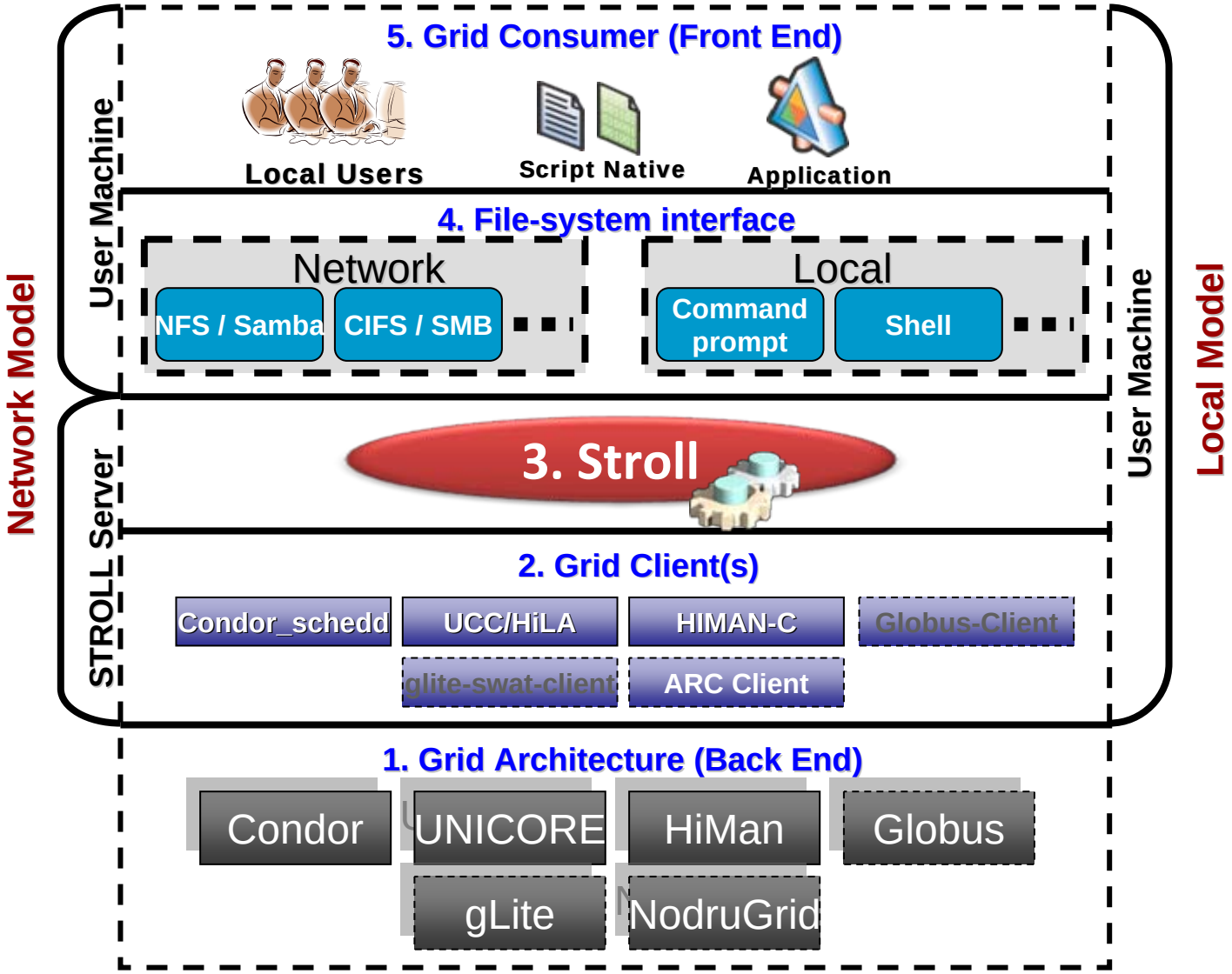
Setup

- 20 single node UNICORE target systems.
- 40 Condor workers, in 3 labs.



University of Stavanger

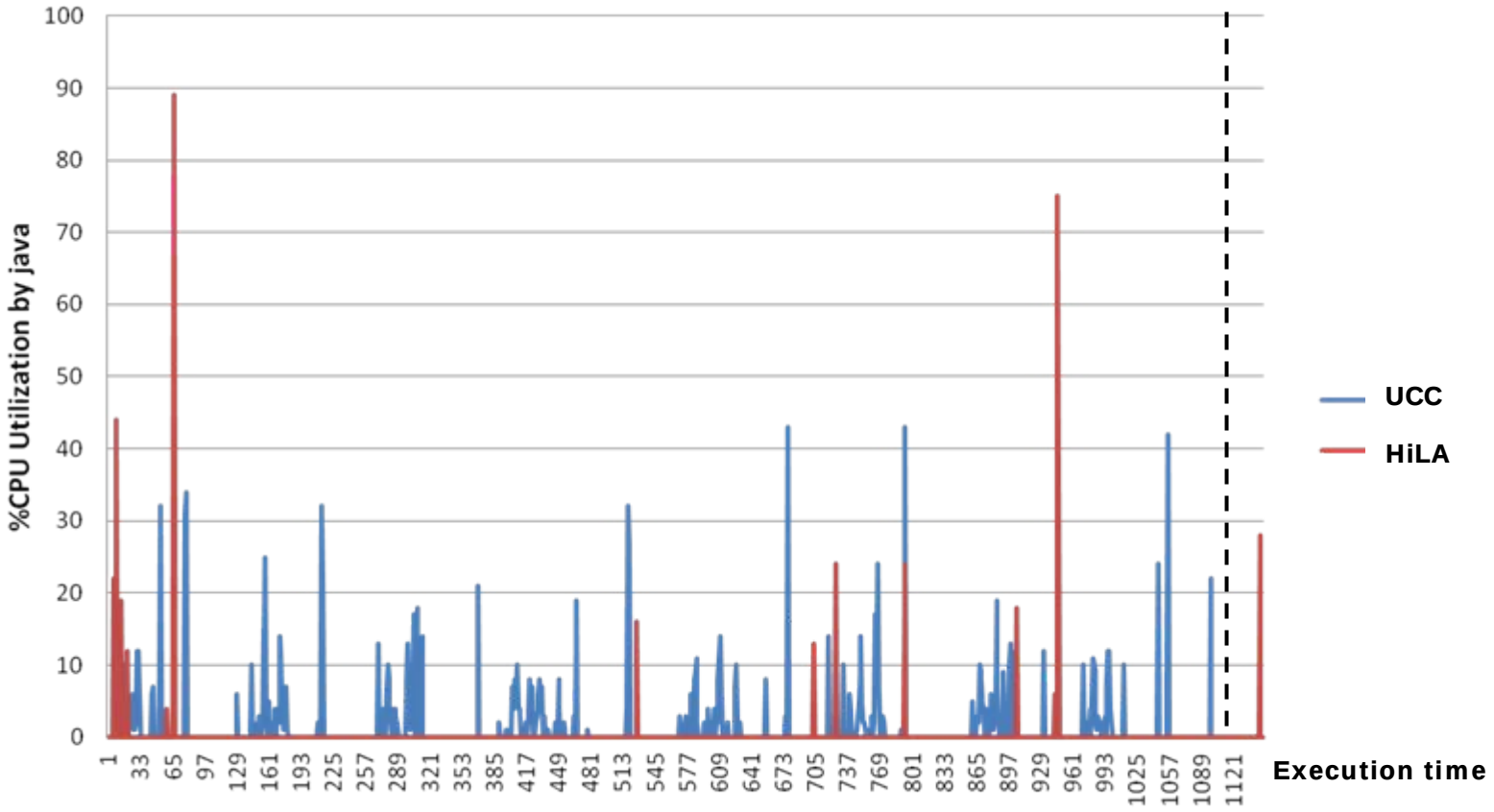
Stroll Architecture



University of Stavanger

Results – UNICORE

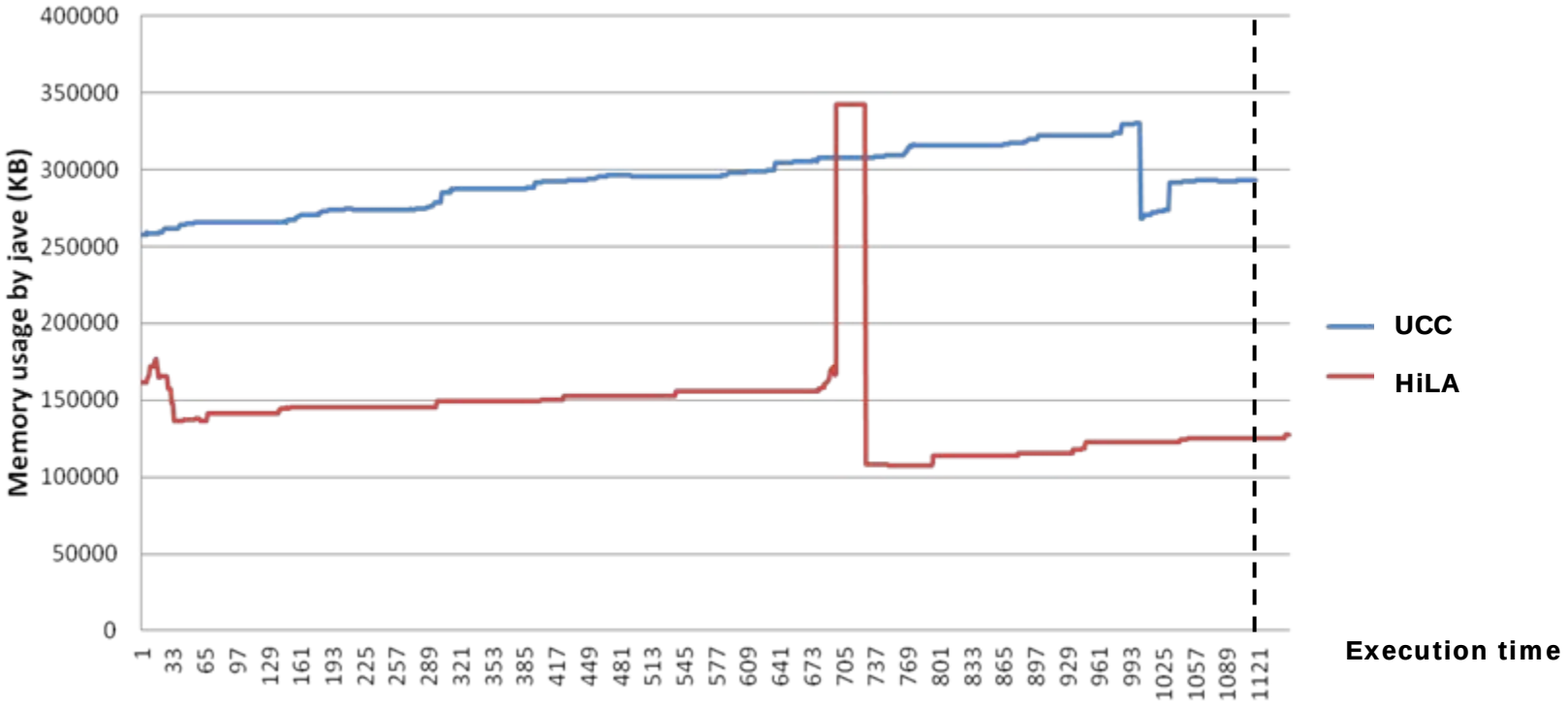
- PSM.estimate(...,maxit=1,...)



University of Stavanger

Results – UNICORE

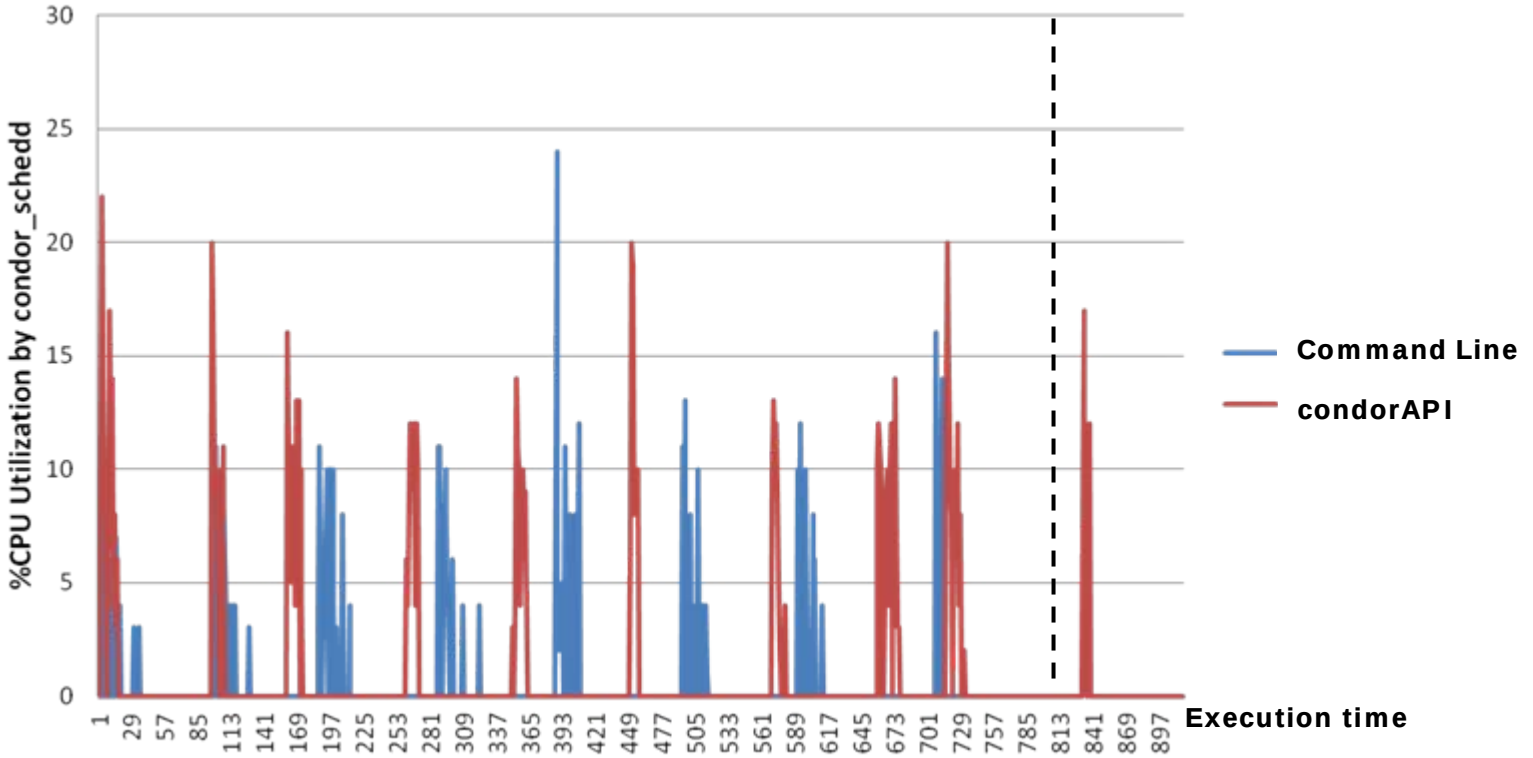
- PSM.estimate(...,maxit=1,...)



University of Stavanger

Results – Condor

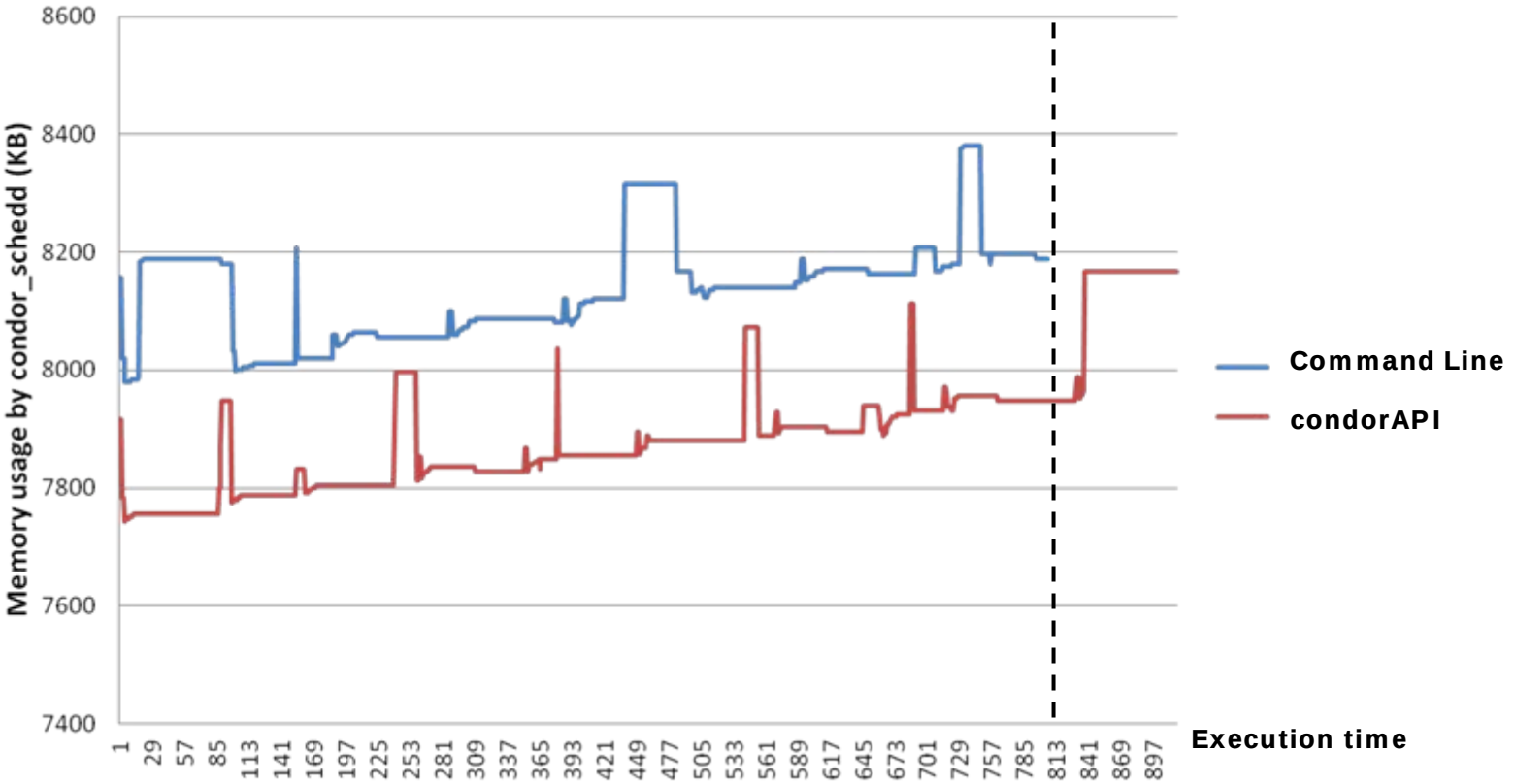
- PSM.estimate(...,maxit=1,...)



University of Stavanger

Results – Condor

- PSM.estimate(...,maxit=1,...)



University of Stavanger

Questions

?

Universitetet i Stavanger