Bioinformatics with large data-sets

Ari-Matti Sarén

Kimmo Mattila

CSC Environment

Louhi

- Cray XT4/XT5 Massively Parallel Processor (MPP) supercomputer
  - quad-core 2.3-GHz AMD Opteron 64-bit processors
  - 9424 cores
  - 1 GB or 2 GB memory/core
  - SeaStar2 interconnects

- Meant for jobs that parallelize well
  - project resources only after scalability test
  - normally 64-512 cores/job
  - can be increased for Grand Challenge projects

- Louhi user's guide
  - http://www.csc.fi/english/pages/louhi_guide

Vuori

- HP CP4000 BL ProLiant supercluster
  - 2x 6-core 2.6 GHz AMD Opteron 64-bit or 6-core Intel X5650 processors/node
  - 128 nodes with 16 GB memory (1,33 GB/core)
    112 nodes with 32 GB memory (2,66 GB/core)
    24 nodes with 96 GB memory (8 GB/core)
    - These nodes have Intel X5650 processors
  - 8 nodes with GPU (for CUDA programs)
    - 24 GB memory/node
  - InfiniBand 20Gb/s (40GB/s for large memory nodes) interconnects

- Meant for serial and mid-size parallel jobs
  - 1-144 cores/job (more posible after scalability tests)

- Vuori user's guide
  - http://www.csc.fi/english/pages/vuori_guide

Murska

- HP CP4000 BL ProLiant supercluster
  - 2 x dual-core 2.6 GHz AMD Opteron 64-bit processors /node
  - 2176 cores
  - 128 cores 8 GB, 512 cores 4 GB, 512 cores 2 GB, 1024 cores 1 GB
  - InfiniBand (4x DDR) network

- Meant for serial and mid-size parallel jobs
  - 1-256 cores/job (1-32 typical, 128 without scalability tests)

- Murska user's guide:
  - http://www.csc.fi/english/pages/murska_guide

Hippu

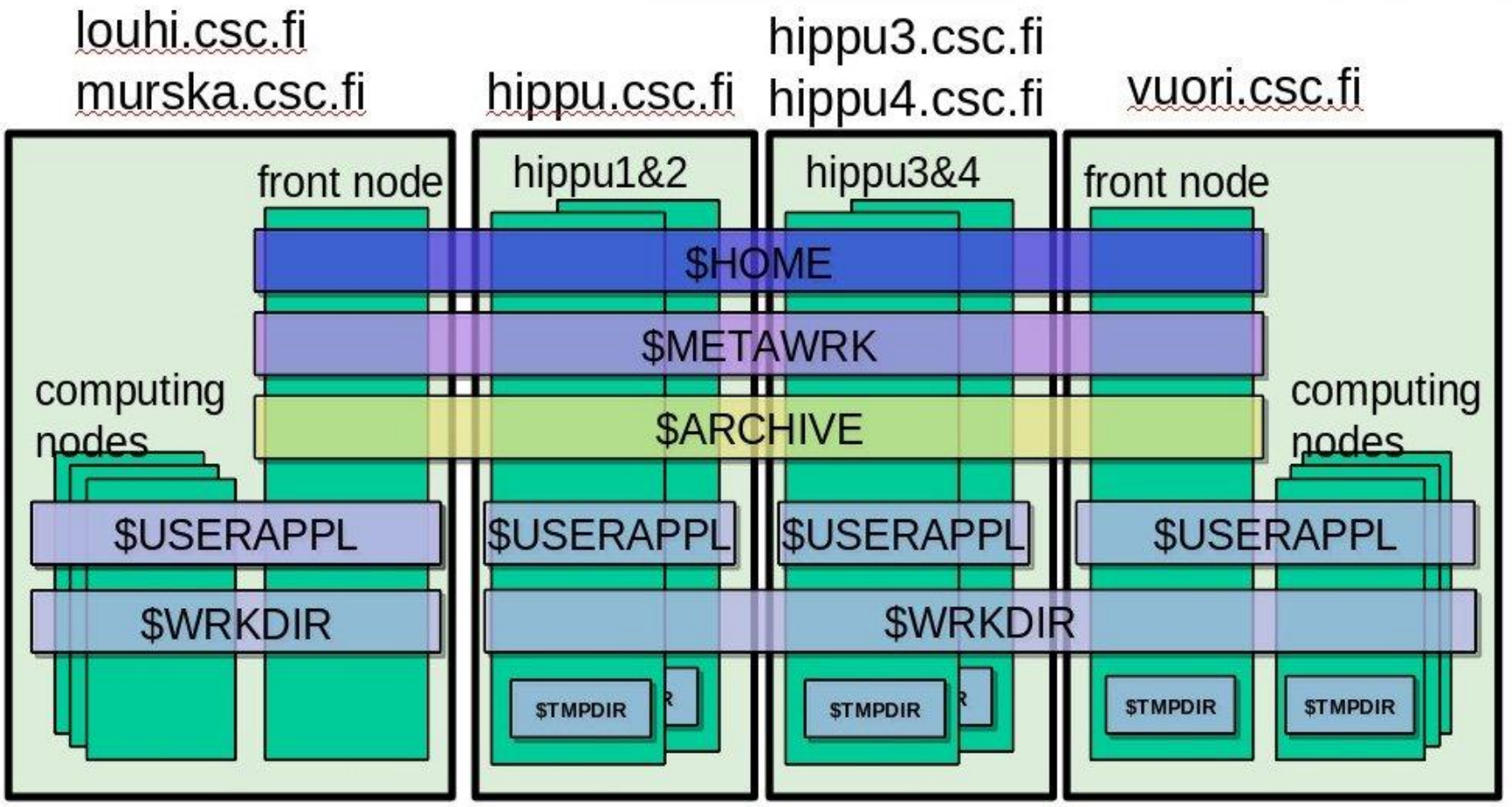
- 2x HP ProLiant DL785 G5 (Hippu1, Hippu2)
  - 8x 4-core 2.5 GHz AMD Opteron 8360 SE 64-bit processors/node
  - 64 cores total
  - 512 GB shared memory/ node
- 2x HP ProLiant DL58 G7 (Hippu3, Hippu4)
  - 4x 8-core Intel Xeon X7560/node
  - 64 cores total
  - 1 TB shared memory/node
- Meant for interactive jobs
  - job length not limited
  - no queue system installed
- Hippu user's guide:
  - http://www.csc.fi/english/pages/hippu_guide

File systems and directories

Directory	Intended use	Default quota/ user	Visibility	Storage time	Backup
$HOME	Initialization scripts, source codes, small data files. Not for running programs or research data.	1 GB	Common, excluding computing nodes	Permanent	yes
$USERAPPL	Users' own application software installations		Server specific	Permanent	Yes
$TMPDIR	Run-time temporary files		Server or computing node specific	~1 day	No
$WRKDIR	Temporary data files	Unlimited	Server specific	At least 7 days	No
$METAWRK	Program development, analysis of results	200 GB	Common, excluding computing nodes	30 days	No
project	Common storage for project members. A project can consist of one or more user accounts	On request	Common, excluding computing nodes	Permanent	Yes
$ARCHIVE	Long-term storage. After user account is closed, its data will be removed from the archive	Default 550 GB or 10 000files. 1 file maximum size 350 GB	Common, excluding computing nodes	permanent	Two tape copies

http://www.csc.fi/english/pages/data-services/Introduction/csc_disks

Visibility of personal directories at CSC



Data handling

Some brief generalizations:

- It’s usually faster to move one large file than many small ones
- On the other hand you should avoid too large files
  - it’s nicer to re-send one 10 GB chunk than the whole 100 GB file
- Consider compression
- Prefer file formats that have checksums or other verification mechanisms
- Data should also be packaged for saving in $ARCHIVE

tar

- concatenates files but does not compress
- preserves directory structure
  - many compression programs don't handle directories well/at all
  - answer: first tar, then compress
- making a tar package:

```
tar cf myfolder.tar myfolde
```

- opening a tar package:

```
tar xf myfolder.tar
```

- checking tar file contents

```
tar tf myfolder.tar
```

http://www.csc.fi/english/pages/data-services/linux_tools/compression#5.2.1

File compression

- File compression/decompression takes time, but saves time on upload/download
  - net gain depends on data size
- Files used in bioinformatics (sequences, pedigree files etc) are often text-based and compress well (to ~30% of original size)
- Compressed file formats typically include checksums
  - if you can uncompress the file without error messages you know your data is intact
- Commonly used compression programs:
  - zip
  - gzip
  - bzip

http://www.csc.fi/english/pages/data-services/linux_tools/compression#5.2.2

- **zip**
  - compressing
    ```
    zip myfiles.zip file1 file2
    ```
  - uncompressing
    ```
    unzip myfiles.zip
    ```
  - leaves origan file intact

- **gzip**
  - compressing
    ```
    gzip myfile
    ```
  - uncompressing
    ```
    gunzip myfile.gz
    ```
  - replaces original file with the compressed file

- **bzip2**
  - slightly better compression ratio than zip/gzip
  - mostly linux spesific
  - compressing
    ```
    bzip2 myfile
    ```
  - uncompressing
    ```
    bunzip2 myfile.bz2
    ```
  - replaces original file with the compressed file

- **Linux**
  - tar, zip, gzip, bzip2 part of most standard distributions

- **Windows**
  - 7-Zip

    free

    makes and opens tar, zip, gzip, bzip2

    http://www.7-zip.org/

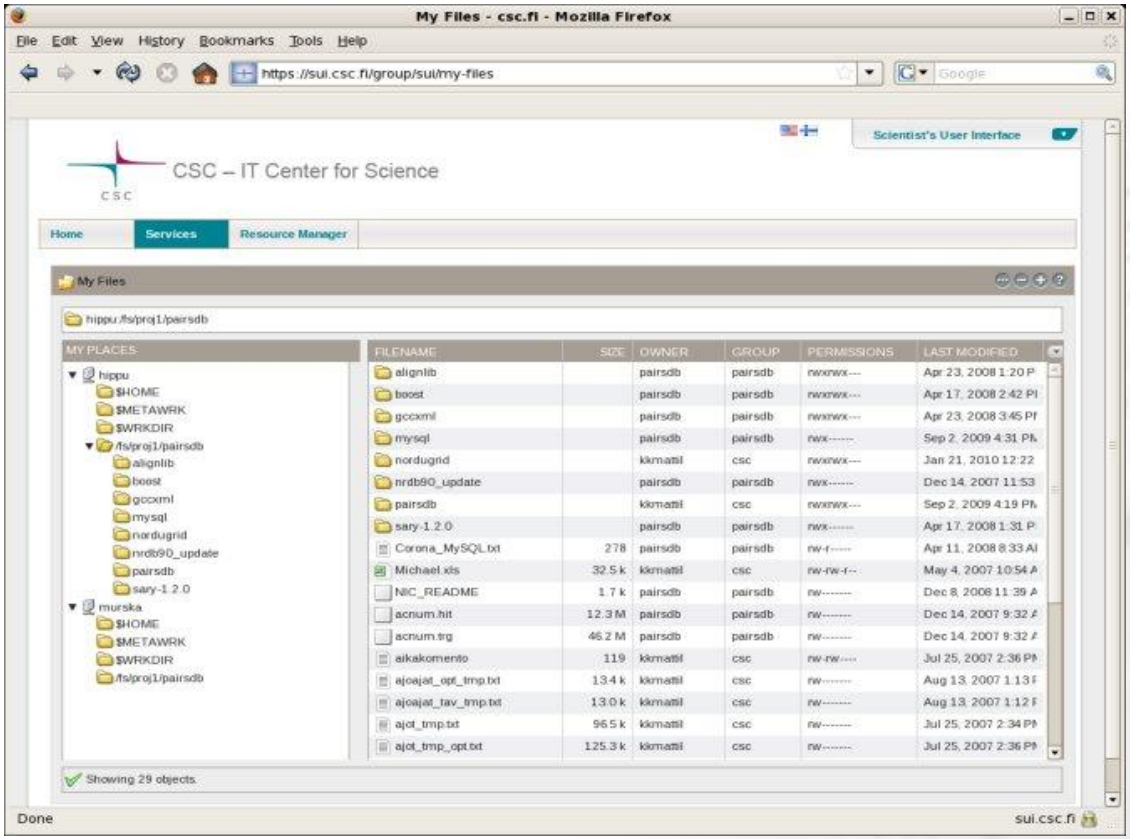
- **Mac**
  - tar, zip, gzip available on standard installation

Moving data to and from CSC

http://www.csc.fi/english/pages/data-services/transport

Scientist's Interface

- MyFiles tool for files up to 2 GB
  - Uploading files larger than 2 GB works on some browsers (Chrome, Safari)
- GSI-SSH Console based SFTP for larger files



http://www.csc.fi/english/pages/data-services/transport/scientists_interface

File transport tools for Linux and Mac

- scp is a standard tool and works well

```
scp myfiles.tar.gz 'user1@murska.csc.fi:$WRKDIR'
scp 'user1@murska.csc.fi:$WRKDIR/myfiles.tar.gz'
```

http://www.csc.fi/english/pages/data-services/transport/file_transport

- rsync can be used for data mirroring and moving very large files

```
rsync -avz -e ssh my_data kkayttaj@hippu.csc.fi:/wrk/kkayttaj
```

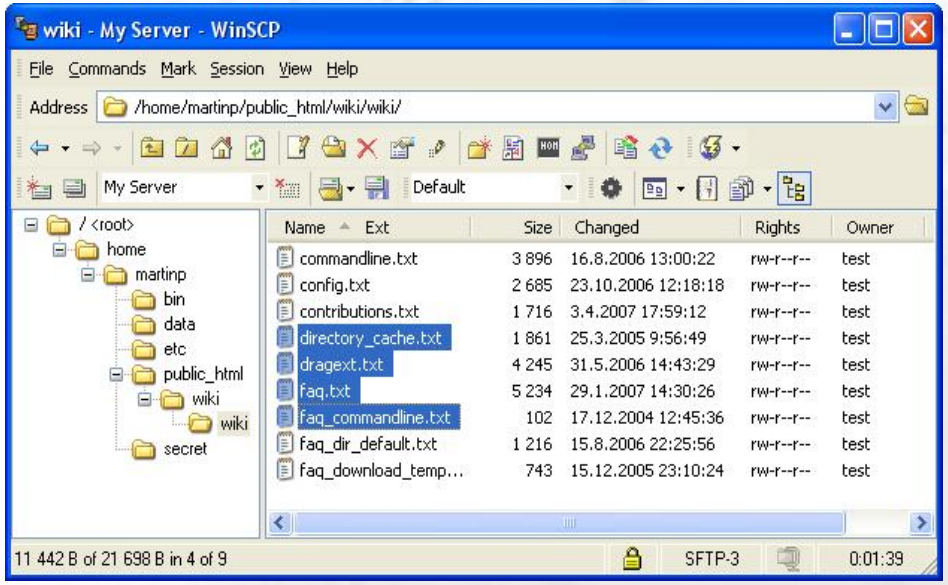
http://www.csc.fi/english/pages/data-services/transport/rsync

- Several graphical file transport tools exisits
  - e.g. Fugu for mac

File transport tools for Windows

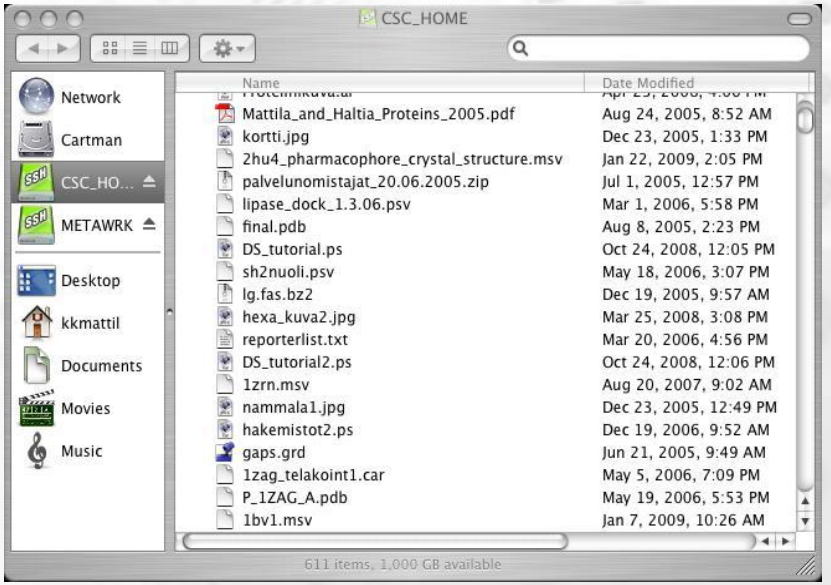
- most commercial ssh programs have graphical file moving interfaces
- commonly used PuTTY does not (it does have command line based scp and sftp)
- winSCP is good free option

http://winscp.net/eng/index.php



Remote disc mounts

- Fuse (linux) and MacFusion (Mac) allow you to mount you disk areas to your local computer
- With this arrangement you can use locally installed tools to work with data that locates at CSC
- You can also transport data in drag-and-drop style.



http://www.csc.fi/english/pages/data-services/transport/remote_mounts

Using the archive server

- Tape based storage
  - very large capacity (currently 560 TB)
  - retrieving the files may take a few minutes
- Can be accessed with normal unix commands: `cp`, `mv`, `rm` etc.
  - mounted as $ARCHIVE on Hippu and on log-in nodes of Murska, Vuori and Louhi
- Avoid archiving small individual files on the server
  - If you have to archive small files, you should first combine them to tar format and compress
- Default size 550 GB or 10 000 files
  - Can be increased by application
- Maximum file size 350 GB

http://www.csc.fi/english/pages/data-services/csc_disks/archive

IDA storage service @CSC

Based on iRODS technology

(Integrated Rule-Oriented Data System)

http://www.csc.fi/sivut/ida

IDA storage service

- iRODS based storage system for storing, archiving and sharing data
- Currently in pilot phase. Research IDA will be officially launched later on this year
- Usage through personal accounts and projects
- Each project has a shared directory too
- Speed: about 1 GB/min at the servers of CSC

Three interfaces:

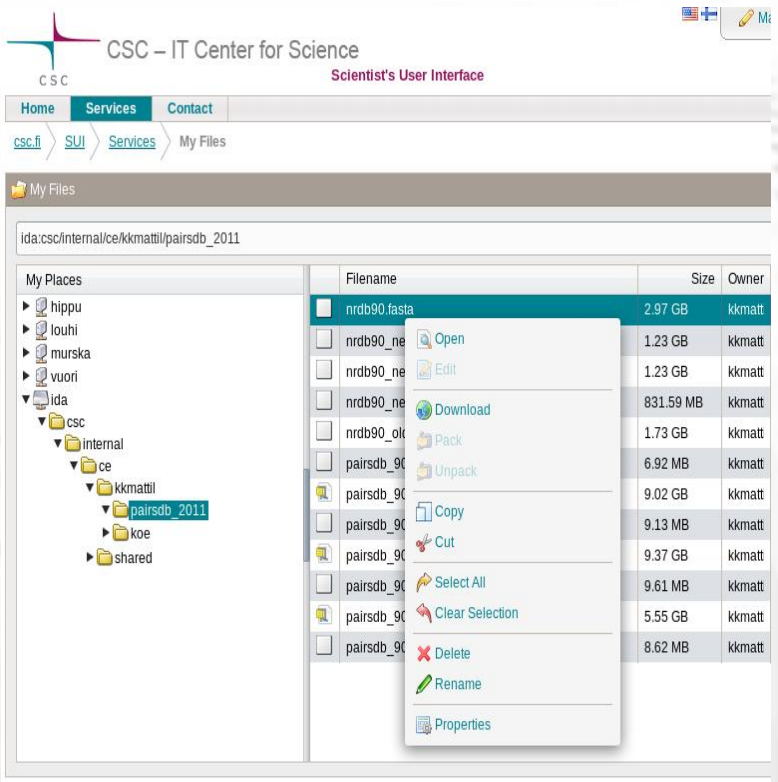
- WWW interface in Scientists' User Interface
- network directory interface for Linux, Mac (and Windows XP)
- command line tools (i-commands installed at the servers of CSC)

IDA interfaces at CSC

Some iRODS commands

- `iput` *`file`* move file to IDA
- `iget` *`file`* retrieve file from IDA
- `ils` list the current IDA directory
- `icd` *`dir`* change the IDA directory
- `irm` *`file`* remove file from IDA
- `imv` *`file file`* move file inside IDA
- `irsync` synchronize the local copy with the copy in IDA
- `imkdir` create a directory to IDA
- `iinit` Initialize your IDA account

IDA In Scientist's User Interface



How to run you jobs

- Interactive jobs
- Batch jobs
- Parallel jobs
- Array jobs
- Grid jobs

Interactive jobs

- Interactive jobs are best run on Hippu
  - no time limit on jobs
  - no need to reserve a node
  - if job takes long, it should be run as background job

    `myprogram &`

    or use `screen`

- If the job:
  - Can be run in batch mode
  - Takes a long time
  - Can use more than one core

  you should consider running it as a batch job in Vuori

Interactive jobs

- Interactive jobs can be run on Murska and Vuori by reserving resources
  - Time limit max 4 h
  - Maximum number of cores for an interactive job:
    - Murska: 32
    - Vuori: 12 (1 node)
  - Example: reserve 1 core with 1 GB memory for 2 hours
    - In Murska:
      ```
      bsub –n 1 –M 1048576  -W 02:00 -Ip $SHELL –I
      ```
    - In Vuori:
      ```
      salloc -n 1 --mem-per-cpu=1000 -t 02:00:00 -p interactive
      ```

screen

- `screen` is a virtual window manager
  - available on Vuori, Murska and Hippu
  - your session stays ”as is” even if you disconnect
- Basic commands
  - open a new screen
    `screen`
  - list open screens
    `screen –ls`
  - re-attach to a screen (if only one open)
    `screen –r`
  - re-attach to screen with id 12345 (as shown by `screen –ls`)
    `screen –r 12345`
  - detach from screen
    `screen -d`
  - screen exits when all processes (including the shell) exit. Or type
    `Ctrl+a Shift+k`

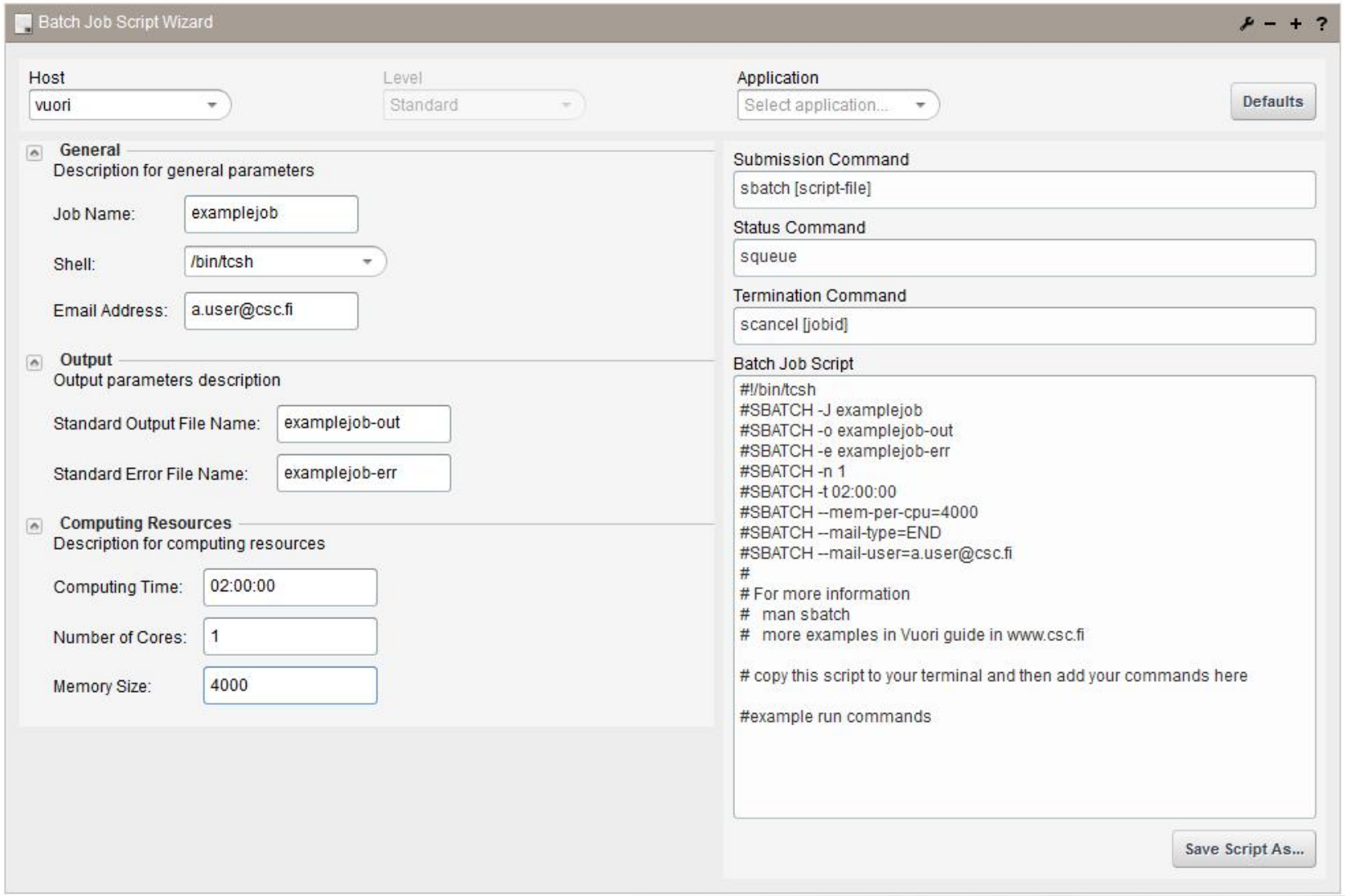
Batch jobs

- Best suited for single, long jobs
- Steps for running a batch job

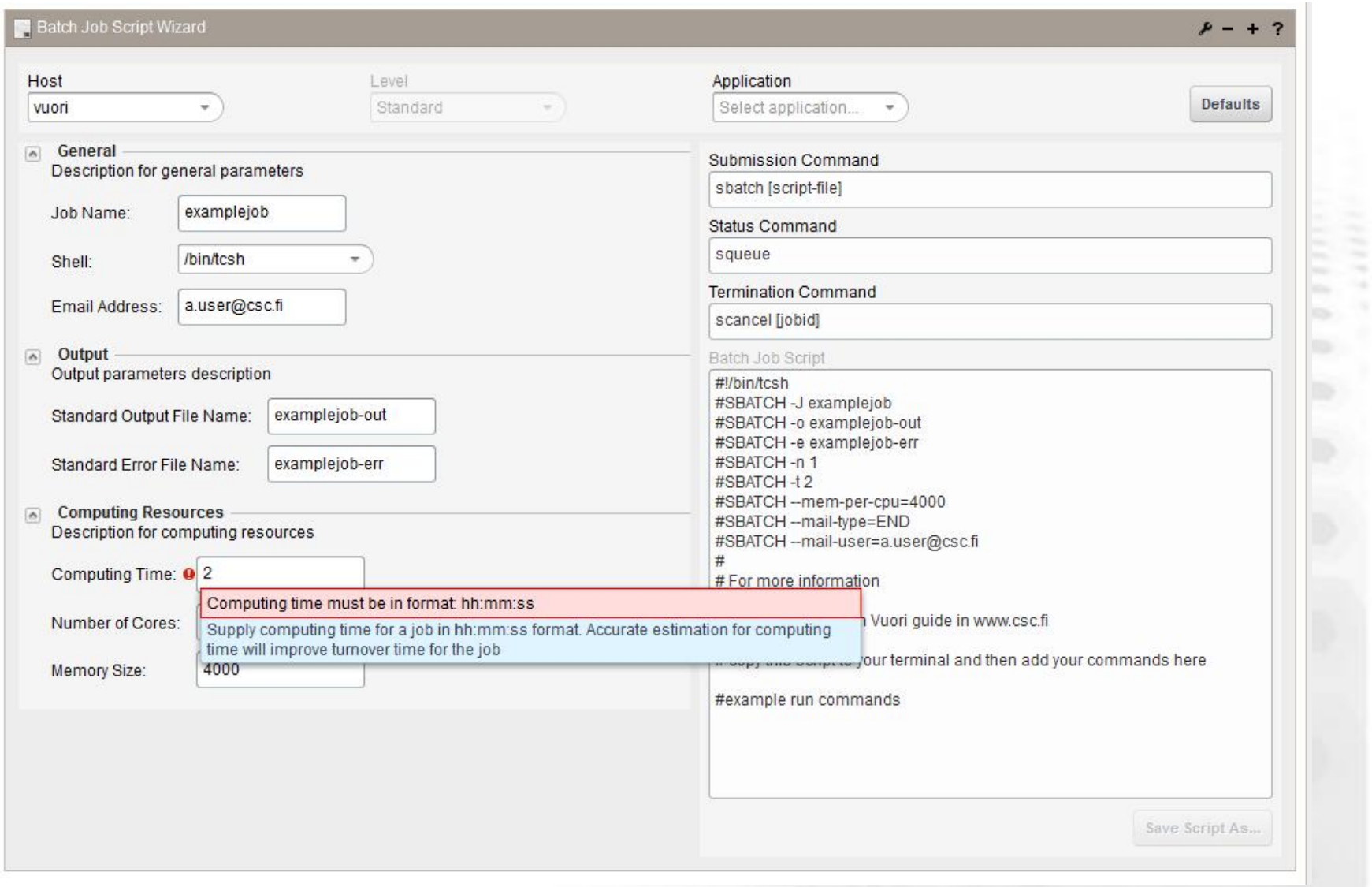
1. write a patch job script
   - Scrip format depends on server
   - You can use the Batch Job Script Wizard in Scientist's User Interface
2. make sure all the necessary files are in $WRKDIR
   - Cluster computing nodes can not see $HOME, $METAWRK, etc.
3. Submit your job

   On Murska: `bsub < myscript`
   On Vuori: `sbatch myscript`
   On Louhi: `qsub myscript`

Batch Job Script wizard in Scientist's User Interface



Batch Job Script wizard in Scientist's User Interface



Example serial batch job script on Vuori:

```
#!/bin/tcsh
#SBATCH -J myjob
#SBATCH -e myjob_err_%j
#SBATCH -o myjob_output_%j
#SBATCH --mail-type=END
#SBATCH --mail-user=a.user@foo.net
#SBATCH --mem-per-cpu=4000
#SBATCH -t 02:00:00
#SBATCH -n 1

module load myprog
srun myprog -option1 -option2
```

```
#!/bin/tcsh
```

- Tells the computer this is a script that should be run using tcsh shell
- Everything starting with ”#SBATCH” is passed on to the batch job system
- Everything starting with ”# ” is considered a comment
- Everything else is executed as a command
- Course examples use tcsh, but you can use other shells if you wish

```
#SBATCH -J myjob
```

- Sets the name of the job
- When listing jobs e.g. with `squeue`, only 8 first characters of job name are displayed.

```
#SBATCH -e myjob_err_%j
#SBATCH -o myjob_output_%j
```

- Option –e sets the name of the file where possible error messages (stderr) are written
- Option –o sets the name of the file where the standard output (stdout) is written
- When running the program interactively these would be written to the command promt
- What gets written to stderr and stderr depends on the program. If you are unfamiliar with the program, it's always safest to capture both
- `%j` is replaced with the job id number in the actual file name

```
#SBATCH --mail-type=END
#SBATCH --mail-user=a.user@foo.net
```

- Option `--mail-type=END` = send email when the job finishes
- Option `--mail-user` = your email address.
- If these are selected you get a email message when the job is done. This message also has a resource usage summary that can help in setting batch script parameters in the future.

`#SBATCH -n 1`

- Number of cores to use
- It's also possible to control on how many nodes you job is distributed
  - This is sometimes necessary for memory management
  - `--ntasks-per-node=12`
- Check software documentation
  - most bioinformatics software can not utilize more than one core
- OpenMP applications can only use cores in one node

`#SBATCH --mem-per-cpu=4000`

- The amount of memory reserved for the job in MB
  - 1000 MB = 1 GB
- Memory is reserved on per-core basis even for shared memory (OpenMP) jobs
- Keep in mind the specifications for the nodes. Impossible requests are rejected directly
- If you reserve too little memory the job will use swap disk and become very slow
- If you reserve too much memory your job will spend much longer in queue

```
#SBATCH -t 02:00:00
```

- Time reserved for the job in hh:mm:ss
- When the time runs out the job will be terminated!
- With longer reservations the job might spend longer in the queue
- Limit for normal serial jobs is 7d (168 h)
  - if you reserve longer time, the job will go to ”longrun” queue (limit 21d)
  - In the longrun queue you run at your own risk. If a batch job in that queue stops prematurely no compensation is given for lost cpu time!

```
module load myprog
srun myprog -option1 -option2
```

- Your commands
  - See application documentation for correct syntax
- Also remember to load modules if necessary
- By default the working directory is the directory where you submitted the job
  - If you include a `cd` command, make sure it points to correct directory
- Remember that input and output files need to be in $WRKDIR

Managing batch jobs in Vuori

- The script file is submitted with command
  `sbatch batch_job.file`
- The job can be followed with commands

`squeue`	(shows all jobs in queue)
`squeue –u <username> -l`	(shows all jobs for a single user)
`squeue –j <jobid>`	(shows status of a single job)

- You can delete a job with command
  `scancel <jobid>`
- more information:
  - http://www.csc.fi/english/pages/vuori_guide/batch_jobs

Parallel jobs

- Only applicable if your program supports parallel running
- Check application documentation on number of cores to use
  - Speed-up is often not linear
  - Maximum number can be limited by the algorithms
- Mainly two types: MPI jobs and shared memory (OpenMP) jobs
  - OpenMP jobs can be run only inside one node
    - All cores access same memory space
  - MPI jobs can span several nodes
    - Each core has its own memory space
- Memory is always reserved per-core basis
  - For OpenMP jobs divide total memory by number of cores
  - Take care to only request possible configurations

Parallel jobs

- Each server has different configuration so setting up parallel jobs in optimal way requires some thought

- See server manuals for specifics
  - http://www.csc.fi/english/pages/louhi_guide/batch_jobs/parallel_jobs/index_html
  - http://www.csc.fi/english/pages/murska_guide/batch_jobs/parallel_jobs/index_html
  - http://www.csc.fi/english/pages/vuori_guide/using_vuori/running_programs/index_html

Array jobs

- Best suited for running the same analysis for large number of files
- On Vuori array jobs are submitted with command `array_sbatch`

```
array_sbatch -file slurm_batch_job_file -from integer -to integer
```

- When submitted variable $VUORI_JOBINDEX will be replaced with the array job index
  - Example:

```
srun myprog –i input.$VUORI_JOBINDEX –o output.$VUORI_JOBINDEX
```

- For information on running array jobs in Murska see:
  - http://www.csc.fi/english/pages/murska_guide/batch_jobs/serial_batch_jobs/

Using grid resources

The Finnish Grid Initiative – FGI

http://www.csc.fi/grids

FGI and NDGF BIO VO

- In grid computing you can use several computing clusters to run your jobs
- Grids suit well for array job like tasks where you need to run a large amount of independent sub-jobs.
- FGI: 10 computing clusters, about 6000 CPUs. (Gradually emerging during 2012)
- NDGF Bio VO: about 10 clusters and over 10 000 computing cores.
- FGI and NDGF Bio VO use same middleware and authentication

Getting started with FGI-Grid

1. Apply for a grid certificate from TERENA (a kind of grid passport)
2. Join the FGI VO (Access to the resources)
3. Install the certificate to Scientists' User Interface and Hippu.

Instructions:
http://www.csc.fi/english/research/sciences/bioscience/programs/blast/gb

Please ask help to get started!
grid-support@csc.fi

Using Grid

You can submit jobs using the ARC middleware

- Using ARC resembles submitting batch jobs in Hippu and Vuori
- ARC is installed in Hippu and Vuori, but you can install it to your local machine too. Setup command:

```
module load nordugrid-arc
```

Basic ARC commands:

`grid-proxy-init -rfc`	(Set up grid proxy certificate for 24 h)
`ngsub` *`job.xrsl`*	(Submit job described in file job.xrsl)
`ngstat -a`	(Show the status of all grid jobs)
`ngkill job_id`	(kill the given grid job)
`ngclean -a`	(remove job related data from the grid)

Sample ARC job description file

```
&
(executable=runbwa.sh)
(jobname=bwa_1)
(stdout=std.out)
(stderr=std.err)
(gmlog=gridlog_1)
(walltime=24h)
(memory=8000)
(disk=4000)
(runtimeenvironment>="APPS/BIO/BWA_0.6.1")
(inputfiles=
( "query.fastq" "query.fastq" )
( "genome.fa" "genome.fa" )
)
(outputfiles=
  ( "output.sam" "output.sam" )
)
```

Sample ARC job script (runbwa.sh)

```
#!/bin/sh
echo "Hello BWA!"
bwa index genome.fasra
bwa aln -t $BWA_NUM_CPUS genome.fasta query.fastq > out.sai
bwa samse genome.fasta out.sai query.fastq > output.sam
echo "Bye BWA!"
exit
```

Using Grid

- At CSC you can use “Gridified” versions of some tools.
- These command line interfaces automatically split and submit the given task to be executed in the grid. The results are also automatically collected and merged.
- You don't have to know ARC to use these tools!

Gridified tools:

- BWA
- SHRiMP
- BLAST
- HHsearch
- Matlab (through Matlab to C conversion)
- AutoDock

- Please suggest a tool that should be “gridified”

Managing files in unix command line

Unix commands

Basic syntax:

comand -option argument

```
ls
ls -l
ls -l myDirectory
```

Use `man` command to get information about possible options

```
man ls
```

Commands for directories:

`cd`	change directory
`ls`	list the contents of a directory
`pwd`	print (=show) working directory
`mkdir`	make directory
`rmdir`	remove directory

Commands for files:

`cat`	print file to screen
`cp`	copy
`less`	view text file
`rm`	remove
`mv`	move/rename a file
`head`	show beginning of a file
`tail`	show end of a file
`grep`	find lines containing given text
`wc`	count number of words or lines

Special characters:

*(asterisk), wild card, means any text

```
ls *.fasta
```

| (pipe) guides output of a command to an input of another commands

```
ls *.fasta | less
```

> Writes output to a new file

```
ls > files_of_the_directory.txt
```

~ (tilde) means your home directory as does $HOME

```
cp test.txt ~/file.txt
cp text.txt $HOME
```

& runs command in background

```
gzip my_big_file.tar  &
```

Quotes

- The tcsh shell is very picky about quotes.

  `''` Take text enclosed within quotes literally

  `` ` ` `` Take text enclosed within quotes as command and replace with output

  `""` Take text within quotes literally after substituting any variables

- Compare the results of these commands:

```
set var = "test"; echo 'echo $var'
set var = "test"; echo `echo $var`
set var = "test"; echo "echo $var"
```

Some useful commands for parsing lines

Try these to see what they do!

sed

```
echo "one this two this three" | sed s/this/that/
echo "one this two this three" | sed s/this/that/g
```

awk

```
echo ”one two three” | awk ’{print $2}’
echo ”one;two;three” | awk –F”;” ’{print $2 $3}’
```

cut

```
echo ”123456789” | cut –c 4
echo ”123456789” | cut –c -4
echo ”123456789” | cut –c 4-
echo ”123456789” | cut –c 4-7
echo ”one_two_three” | cut –d ”_” –f 2
```

All of these have much more options. See man pages for details.

Automating analysis: Introduction to shell scripting

Scripting

- For more complex operations it is advisable to use a full-fledged programming language (*e.g.* python, perl)
- Shell scripting can be very usefull for automating analysis, handling large number of files, doing repetitive tasks etc.
- You can get a lot done with just a handful of commands
- Course examples are in tcsh, but other common shells available

Getting started:

- Write & save the script using a text editor
  - Make sure to save in text format (not in Word .doc or similar)
  - If importing files from Windows it's good idea to run dos2unix:
    `dos2unix myfile`
- Every tcsh script should start with
  `#!/bin/tcsh`
- Remember to give yourself execution rights to the script
  `chmod u+x filename`
- If the scripts are not in your $PATH, remember to give path in command (same as with all other programs)
  - to run a script in your current directory
    `./myscript`
- Remember: `man` command and Google are your friends

Variables

```
set variable = (value)
$variable

set array = (a b c)
$array[1], $array[2], $array[3]
```

Note:
`$argv[]` holds command line arguments

Example:

```
set len = (1)
echo $len
```

Conditional structures

```
if (condition) command

if (condition) then
   commands
else
   commands
endif
```

Example:

```
if ($n < 10) then
   echo ”small”
else
   echo ”large”
endif
```

while loop

```
while (condition)
   commands
end
```

Example:

```
set n = 1
while ( $n < 10)
   echo $n
   set n = (`expr $n + 1`)
end
```

Foreach structure

```
foreach variable (value_list)
   commands
end
```

Examples:

```
foreach gap ( 5 10 15 20 25 )

foreach sequence (`ls *.seq`)

foreach sequence (`ls | grep .seq$`)

foreach line (”`cat myfile.txt`”)
```

Logical operators

- You can use the normal conditional operands

==	equal to
!=	not equal to
=~	similar to (allows wildcards)
!~	not similar to (allows wildcards)
>	greater than
<	smaller than
>=	greater or equal to
<=	smaller or equal to
&&	AND
||	OR
!	NOT

File-test operators

- There are a number of operators you can use to test different attributes of a file:

`-e file`	true if file exists
`-z file`	true if file exists and is zero size
`-d dir`	true if dir exists and is a directory
`-o file`	true if file exists and is owned by the user
`-r file`	true if file exists and is readable
`-w file`	true if file exists and is writable
`-x file`	true if file exists and is executable

- Examples

```
if (-e file) echo ”file exists”
if !(-e file) echo ”file does not exist”
```

Example scripts

- The following two scripts use two loops to study the gap opening penalty and gap extension penalty for global alignments of sequences swiss:cas1_human and swiss:cas1_sheep. Gap creation penalty is varied from 5 to 50 with step size of 5 and the gap extension penalty from 1 to 5 with step size 1. The resulting similarity and identity values are sorted according to the similarity and stored to file ”sorted.out”

- Note that the `expr` comand can only handle integers, so if you would like to use step size less than one, you should use different commands.

- The first version of the sample scripts uses a `foreach` loop and the second version a `while` loop. Results are the same in both cases

Example 1

```
#!/bin/tcsh

use emboss
echo "Testing matcher" > result.out

set gap = (5)
set n = (1)

foreach gap (5 10 15 20 25 30 35 40 45 50)

    foreach len (1 2 3 4 5)

        stretcher swiss:cas1_human swiss:cas1_sheep -gapopen $gap -gapextend \
            $len -outfile out.pair -auto

        echo "`grep Similarity out.pair` Len= $len Gap $gap" >> result.out
        rm -f out.pair

    end
end

sort result.out > sorted.out
```

Example 2

```
#!/bin/tcsh
use emboss
echo "testing matcher" > result.out

set gap = (5)
set n = (1)

while ($gap <= 50)
    set len = (1)

    while ($len <= 5)

        stretcher swiss:cas1_human swiss:cas1_sheep -gapopen $gap \
           -gapextend $len -outfile out.pair -auto

        echo "`grep Similarity out.pair` Len= $len Gap $gap" >> result.out
        rm -f out.pair
        set len = (`expr $len + 1`)
    end

    set gap = (`expr $gap + 5`)
end

sort result.out > sorted.out
```

MySQL-database services @CSC

kaivos.csc.fi

MySQL-database services @CSC

- Kaivos.csc.fi can be accessed form Hippu and Murska
- No login to the server, just remote MySQL-client connections
- Size and the number of simultaneous connections (5) is limited
- No personal database accounts, but database specific
  - **DBname_admin** can create and remove tables and indexes for the database.
  - **DBname_user** can read, write, change and delete data from the database tables. However, this account can't change the structure of the database
  - **DBname_read** can do only queries to the database.
- The database administrator (i.e. you) is responsible of the structure and content of the database!
- Database application
- http://www.csc.fi/english/customers/university/useraccounts/databaseservice.pdf

MySQL-database services @CSC

Tools

- MySQL-client
- Graphical interfaces mysqlcc (Murska) mysql-query-browser (Hippu)
- Perl and Python API (not officially supported)

Benefits

- SQL-queries
- Same data can be used from Hippu, Murska and Vuori and from you local computer (requires ssh-tunnel)
- You can use the database from batch jobs in Murska and Vuori
- Up to 100 GB backuped database size

http://www.csc.fi/english/pages/data-services/databases