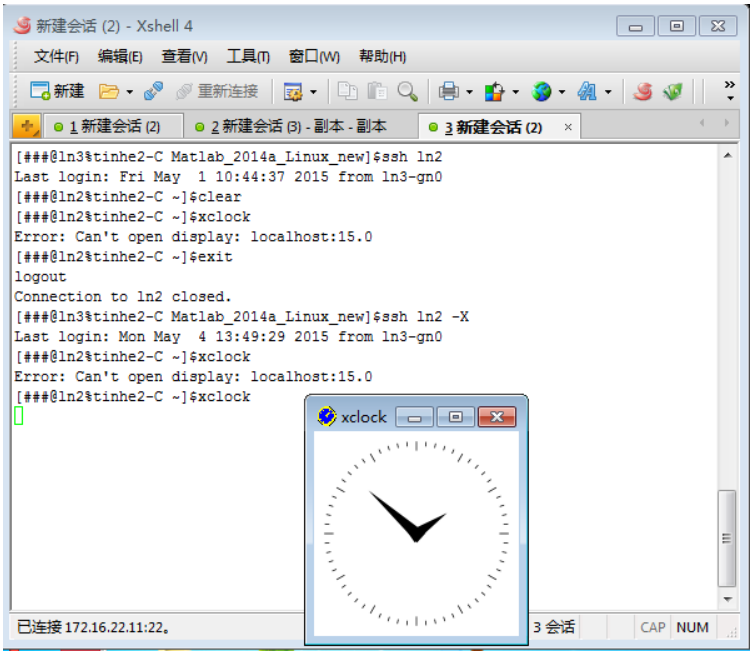
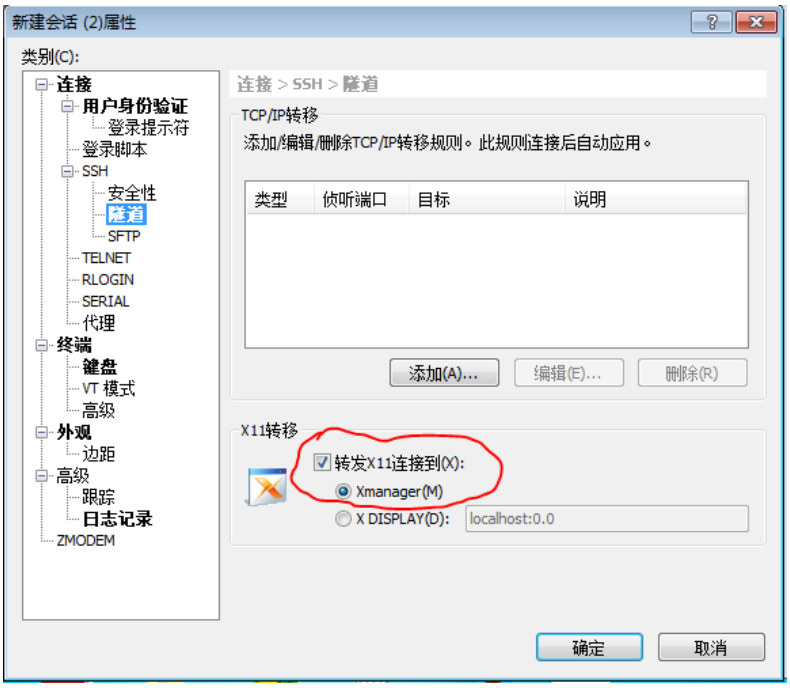
如何在天河 II 上安装和使用 MATLAB

安装

0） 安装需要使用支持图形界面的客户端，推荐 Xmanager ； 另外因为需要开图形界面，不能在 ln3 安装， 本例在 ln2 进行 ； 可以用 xclock 检查是否图形功能开启，可能需要设置连接属性。





1） 上传安装文件到系统上，一般是 ISO 文件 ，最好创建好文件夹

2） 解压缩 ，可使用系统提供的 7z 解压 iso 文件

```
[###@ln2%tinhe2-C Matlab_2014a_Linux_new]$module load p7zip/9.20.1
[###@ln2%tinhe2-C Matlab_2014a_Linux_new]$7z x MATHWORKS_R2014A.iso
```

有些解压过程中会出现一些询问，按 A 选择 Always 即可

不过解压过程会丢失一些文件的权限信息，可以将所有文件加上可执行权限

```
[###@ln2%tinhe2-C Matlab_2014a_Linux_new]$ls
Crack                activate.ini  etc      install_guide.pdf    license.txt  sys
InstallForMacOSX.app archives      help     installer_input.txt  patents.txt  trademarks.txt
MATHWORKS_R2014A.iso bin           install  java                 readme.txt   version.txt
[###@ln2%tinhe2-C Matlab_2014a_Linux_new]$chmod -R  +x .
[###@ln2%tinhe2-C Matlab_2014a_Linux_new]$ls
Crack                activate.ini  etc      install_guide.pdf    license.txt  sys
InstallForMacOSX.app archives      help     installer_input.txt  patents.txt  trademarks.txt
MATHWORKS_R2014A.iso bin           install  java                 readme.txt   version.txt
[###@ln2%tinhe2-C Matlab_2014a_Linux_new]$
```

3） 安装

一般来说用 ./install 即可启动安装过程

不过有些版本的包有些问题，会遇到各种错误，可以谷歌解决办法，比如本次安装就遇到了问题：

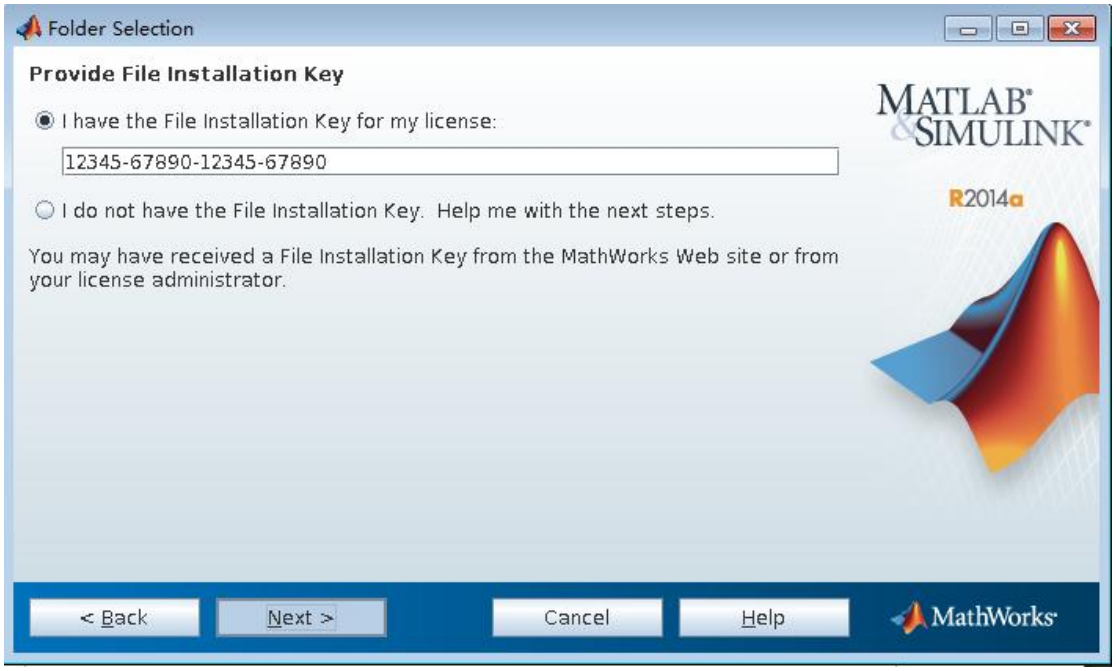
```
[###@ln2%tinhe2-C Matlab_2014a_Linux_new]$./install
Preparing installation files ...
Installing ...
/tmp/mathworks_9688/sys/java/jre/glnxa64/jre/bin/java: error while loading shared libraries: libjli.so: ca
nnot open shared object file: No such file or directory
Finished
[###@ln2%tinhe2-C Matlab_2014a_Linux_new]$
```

不过这个问题网上有解决办法,参考网上的方法，可以按如下的方法正常安装

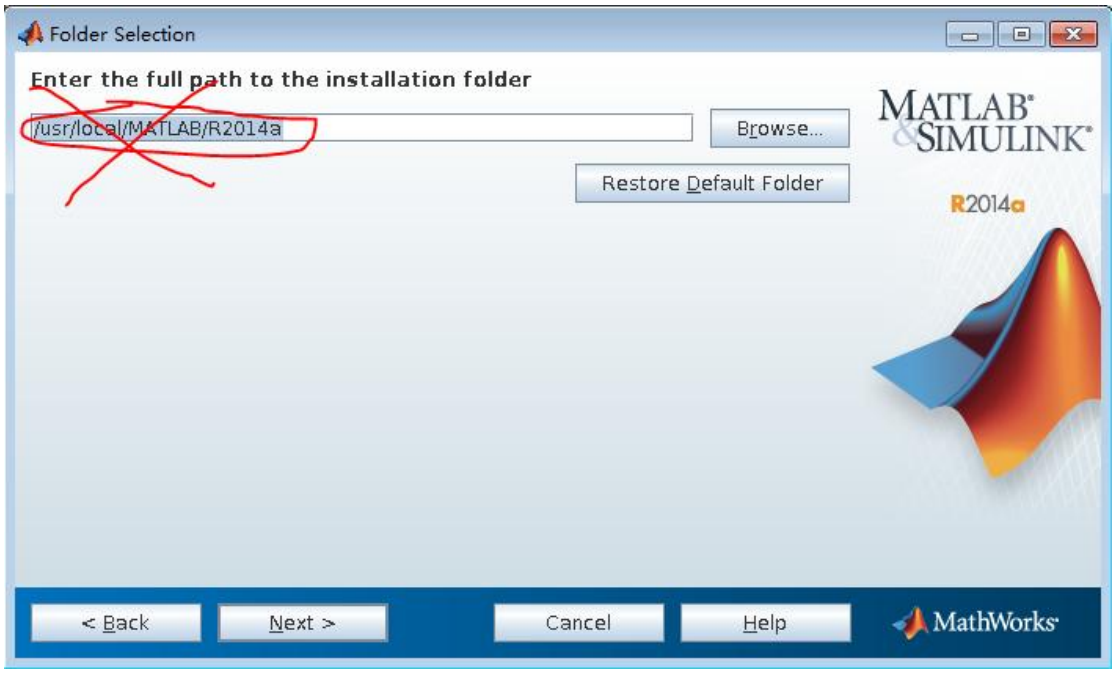
```
[###@ln2%tinhe2-C Matlab_2014a_Linux_new]$locate libjli.so
/usr/lib/jvm/java-1.6.0-openjdk-1.6.0.0.x86_64/jre/lib/amd64/jli/libjli.so
/usr/lib/jvm/java-1.6.0-openjdk-1.6.0.0.x86_64/lib/amd64/jli/libjli.so
/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.45.x86_64/jre/lib/amd64/jli/libjli.so
/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.45.x86_64/lib/amd64/jli/libjli.so
[###@ln2%tinhe2-C Matlab_2014a_Linux_new]$./install  -javadir /usr/lib/jvm/java-1.7.0-openjdk-1.7.0.45.x86
_64/jre
Preparing installation files ...
cp: cannot stat `/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.45.x86_64/jre/lib/audio/default.sf2': No such file
or directory
```

等待一会儿后会弹出安装界面

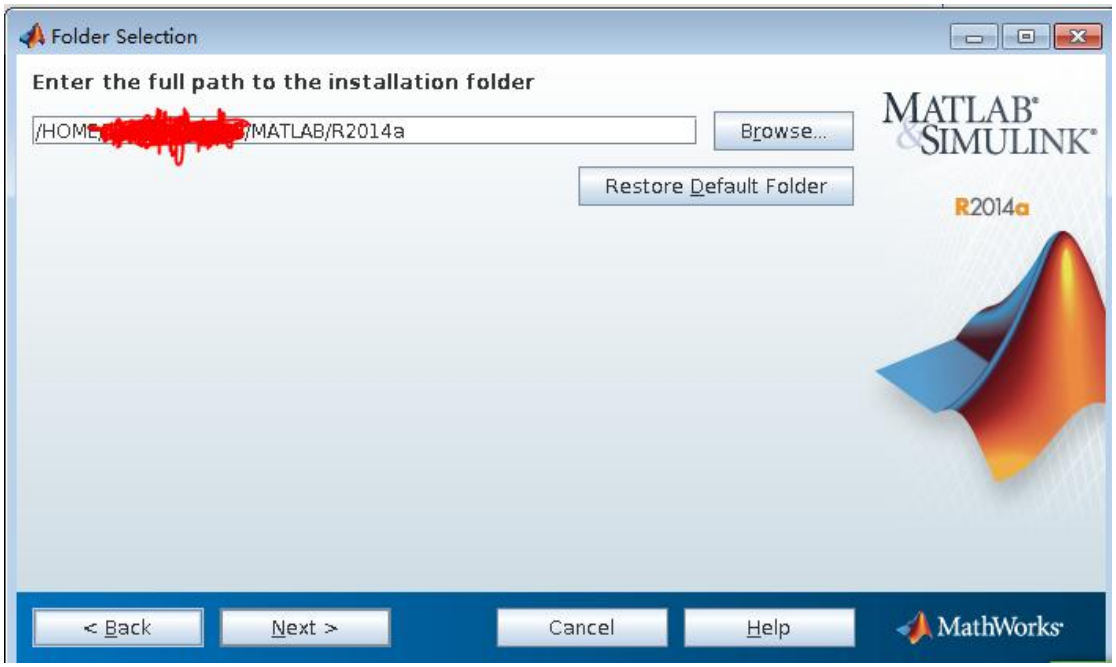
之后的过程就很平常了 ，选择 File Installation Key 安装， Next , Next



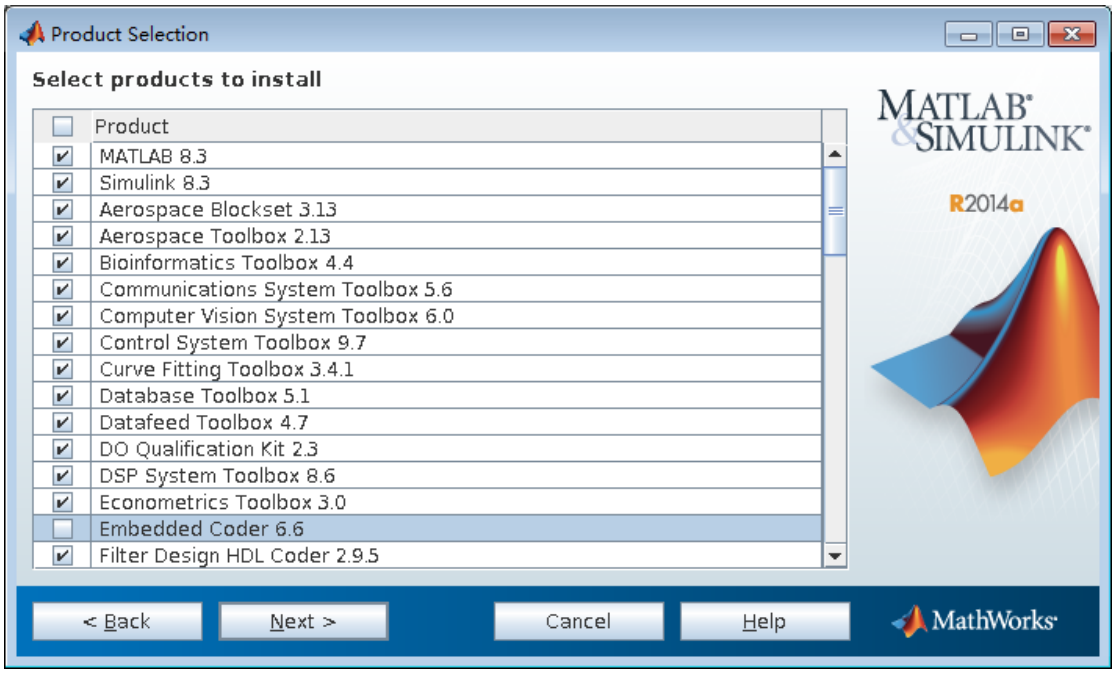
输入自己的 KEY



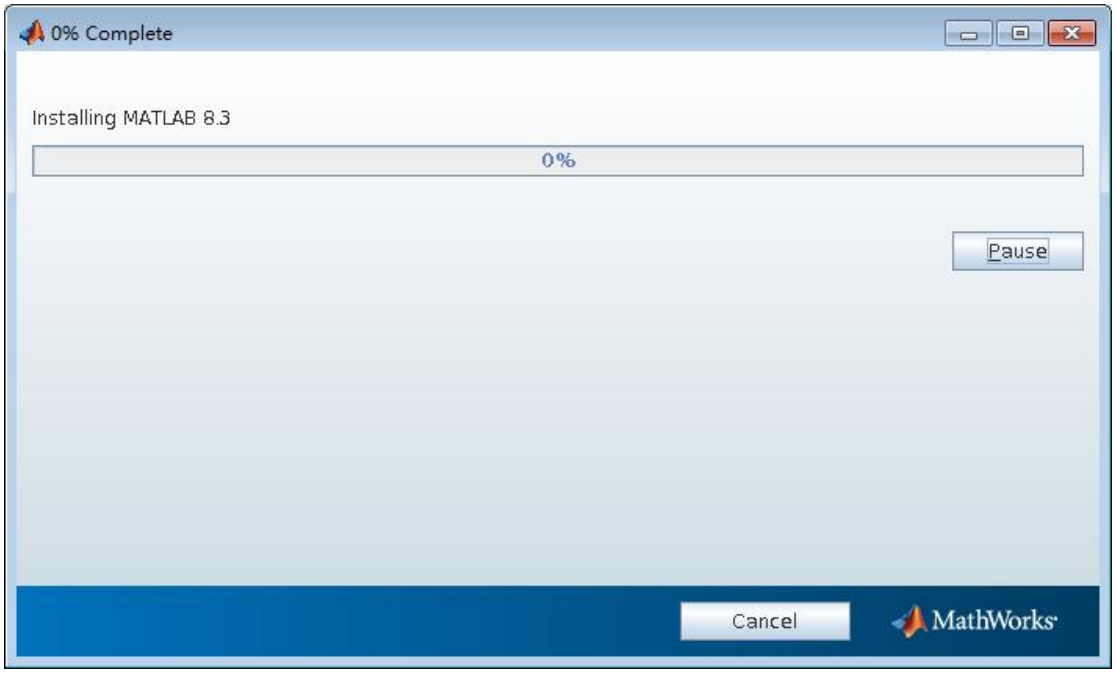
注意，这里不能使用默认的安装路径；需要使用自己有权限的路径,如：



继续 NEXT



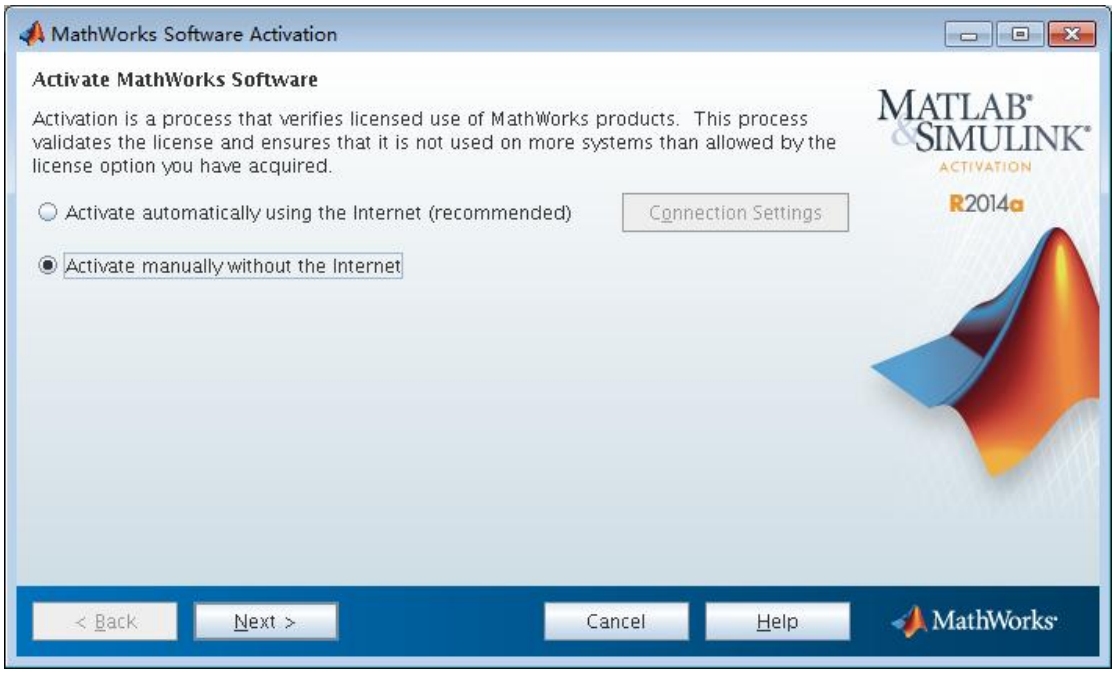
一路 NEXT ,直到漫长的等待

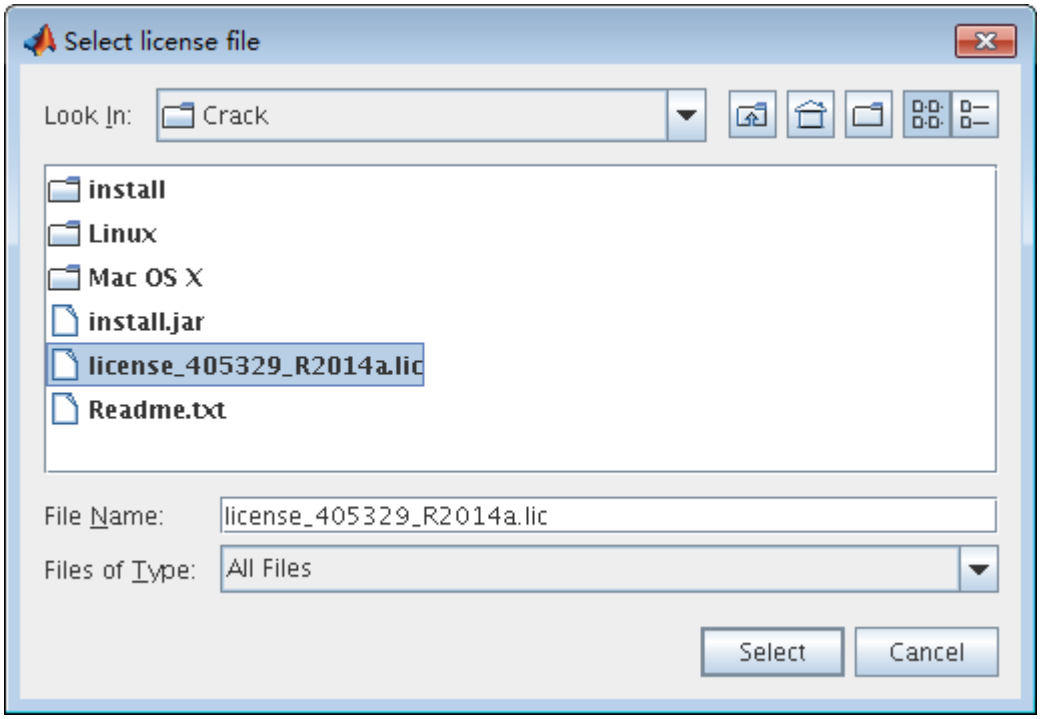


一直到最后结束 Finish

4） 破解

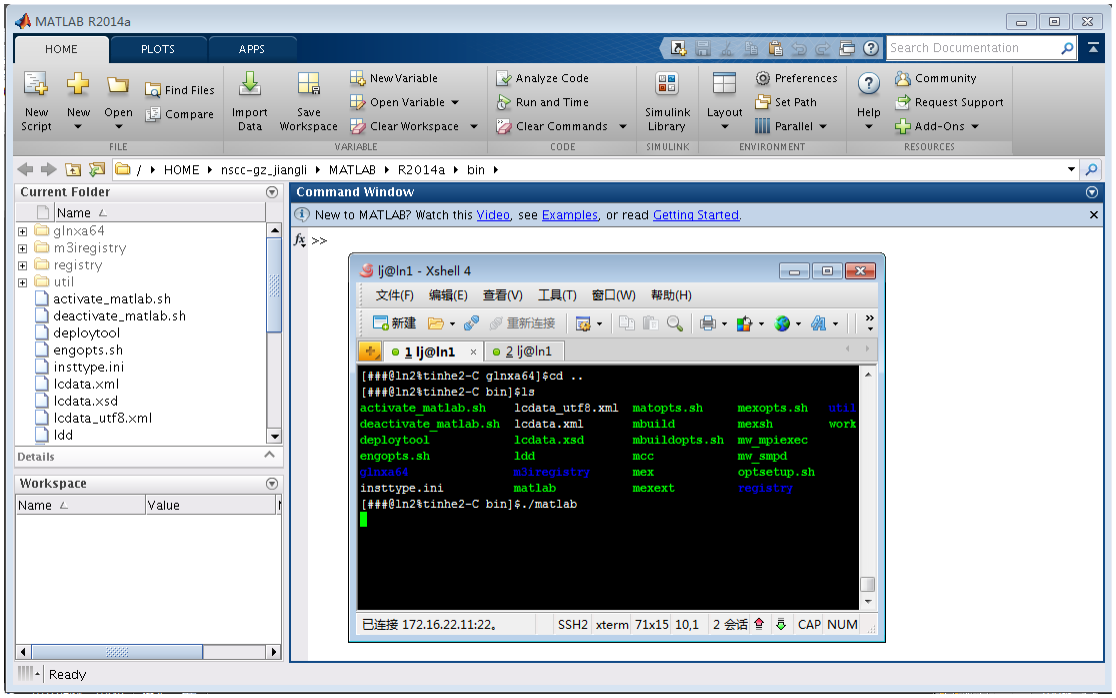
不同的安装包有不同的破解方式，一般都能在安装文件里找到， 本例参考 Crake 文件夹里面的 ReadMe 进行






```
[###@ln2%tinhe2-C glnxa64]$mv libmwservices.so libmwservices.so-bk
[###@ln2%tinhe2-C glnxa64]$cp ~/Matlab_2014a_Linux_new/Crack/Linux/libmwservices
.so .
[###@ln2%tinhe2-C glnxa64]$clear
[###@ln2%tinhe2-C glnxa64]$
```

这一部分请自行参考自己下载的 MATLAB 包里的破解说明进行。完成后就可以正常启动 Matlab 了 ：



设置

为了能够方便使用和在计算节点正确使用，需要一些设置，这些设置最好直接写到 ~/.bashrc 文件里：

export PATH=~/MATLAB/R2014a/bin:$PATH

将刚刚安装的 MATLAB 的 bin 文件夹加到 PATH 环境变量；这样就可以直接通过 matlab 启动 刚刚安装的 MATLAB 了

export LC_ALL=C

避免在计算节点使用 MATLAB 出错

source /WORK/app/osenv/ln1/set2.sh

#避免在计算节点使用 MATLAB 出错；这个是天河 II 提供的公共工具脚本，来一致计算节点和登录节点的环境 ； 不然在计算节点使用时会出现 error while loading shared libraries: libXt.so.6: cannot open shared object file: No such file or directory 的错误

使用

虽然可以通过 ./matlab 在一些登录节点启动 MATLAB ,但是会被管理员警告或者封号，而且在登录节点一般性能比较差， 正常使用还是需要使用天河 II 的作业调度系统来进行。

1） 交互式使用 ：

```
[###@ln2%tinhe2-C bin]$yhrun -n 1 matlab -nojvm -nodisplay
yhrun: job 445198 queued and waiting for resources
yhrun: job 445198 has been allocated resources

                            < M A T L A B (R) >
                  Copyright 1984-2014 The MathWorks, Inc.
                    R2014a (8.3.0.532) 64-bit (glnxa64)
                               February 11, 2014

Warning: X does not support locale C

To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

```

这样可以进行简单的交互式使用

```
>>
a =

     3

b=5
>>
b =

     5

a*b
>>
ans =

    15

exit
>> [###@ln2%tinhe2-C bin]$
```

当然，实际上一般是不会这么用的

2） **重要 ！** 正常的使用还是使用 脚本的方法 来进行， 使用方式如下：

如，有需要运行的 m 文件 test1.m :

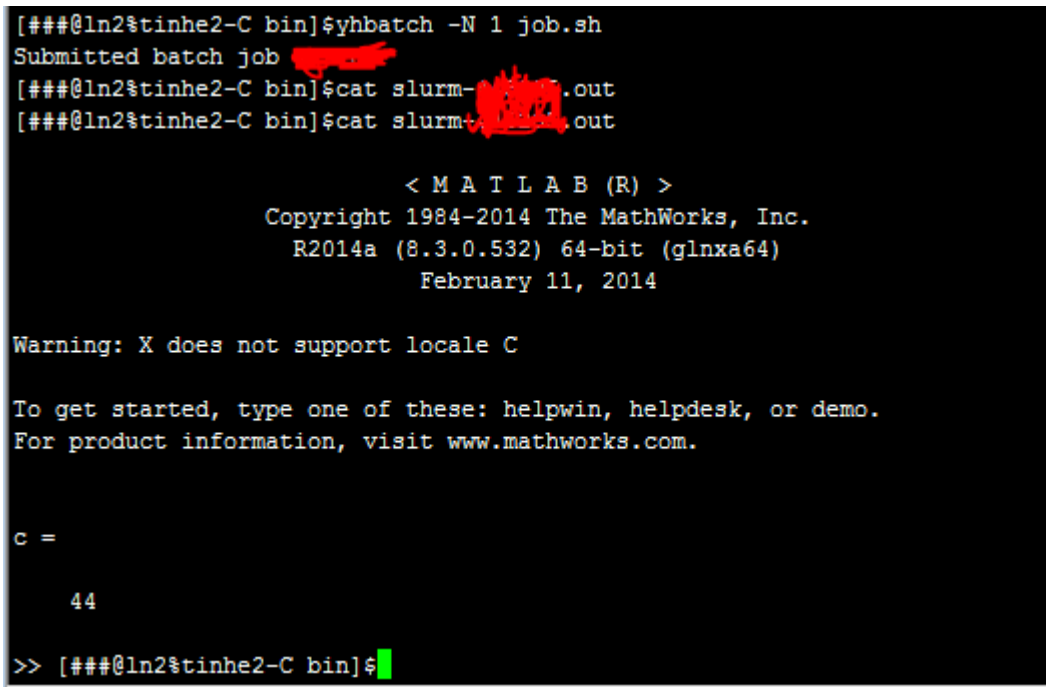
```
[###@ln2%tinhe2-C bin]$cat test1.m
a=[1 , 3 ,5 ] ;
b=[2 , 4 , 6 ]' ;
c= a*b

[###@ln2%tinhe2-C bin]$
```

可以编写任务提交的脚本，如下面的 job.sh ：

```
[###@ln2%tinhe2-C bin]$cat job.sh
#!/bin/sh
matlab -nojvm -nodisplay -r "test1"
[###@ln2%tinhe2-C bin]$
```

！ 如果要使用某些功能，则不能有 –nojvm 选项 。

使用 yhbatch 提交此 任务



(* 被涂掉的地方是作业号

并行计算

1） 什么都不做,利用 MATLAB 自带的 OpenMPA 并行
如本例（基于 JMBENCH
http://math.ucdenver.edu/~jmandel/matlab/matlab_benchmark.html ）：

```
[###@ln2%tinhe2-C matlab]$cat bench1.m
% function jmbench

disp('Matlab benchmark 1 ')

disp(['Matlab version ',version])

for iii=1:30
disp(' ')

f='%7.3f\n';
fprintf('1. large LU: ')
n=10000;m=3;
A=ones(n)+eye(n);
tic
for i=1:m,R=chol(A); end
t=toc; fprintf(f,t)


pause(1)

end
disp(' ')
disp('end of jmbench')


[###@ln2%tinhe2-C matlab]$cat jobb1.sh
#!/bin/sh
matlab -nojvm -nodisplay -r "bench1"
[###@ln2%tinhe2-C matlab]$yhbatch -N 1 jobb1.sh
```

提交完任务后，登录到对应的计算节点查看，可以看到虽然程序没有任何显示并行措施，CPU 利用率 却达到了 1821% ， 也就是用到了多个核！
(并不是所有的操作都已经进行了 OMP 自动并行)

```
top - 11:50:36 up  2:32,  1 user,  load average: 3.62, 0.86, 0.27
Tasks: 511 total,   2 running, 509 sleeping,   0 stopped,   0 zombie
Cpu(s): 74.8%us,  1.3%sy,  0.0%ni, 23.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  132274488k total,  4656744k used, 127617744k free,        0k buffers
Swap:        0k total,        0k used,        0k free,  1139788k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2723 [redacted]  20   0 6211m 1.7g  55m S 1821.3  1.4   4:59.22 MATLAB
```

一般来说可以通过设置 OMP_NUM_THREADS 环境变量控制 MATLAB 的线程并行， 但是并不一定起作用 。

2） parfor 这个显示的并行策略最适合在 HPC 上进行应用
http://cn.mathworks.com/help/distcomp/introduction-to-parfor.html
可参考下面的性能测试的内容

性能测试

参考资料：
http://cn.mathworks.com/help/distcomp/examples/benchmarking-independent-jobs-on-the-clus

ter.html

测试用文件：

```
[###@ln2%tinhe2-C matlab]$cat bench2.m
myCluster = parcluster;
numHands = 1e5;
numReps = 5;
numWorkers = myCluster.NumWorkers ;
if isinf(numWorkers) || (numWorkers == 0)
    error('pctexample:distribjobbench:InvalidNumWorkers', ...
        ['Cannot deduce the number of workers from the cluster.  ' ...
         'Set the NumWorkers on your default profile to be ' ...
         'a value other than 0 or inf.']);
end

numTasks = [pow2(0:ceil(log2(numWorkers) - 1)), numWorkers];

fprintf(['Starting weak scaling timing.  ' ...
        'Submitting a total of %d jobs.\n'], numReps*length(numTasks));
for j = 1:length(numTasks)
    n = numTasks(j);
    for itr = 1:numReps
        [rep(itr), description] = timeJob(myCluster, n, numHands);  %#ok<AGROW>
    end
    % Retain the iteration with the lowest total time.
    totalTime = [rep.totalTime];
    fastest = find(totalTime == min(totalTime), 1);
    weak(j) = rep(fastest);  %#ok<AGROW>
    fprintf('Job wait time with %d task(s): %f seconds\n', ...
            n, weak(j).jobWaitTime);
end

seqTime = inf;
for itr = 1:numReps
    start = tic;
    pctdemo_task_blackjack(numHands, 1);
    seqTime = min(seqTime, toc(start));
end
fprintf('Sequential execution time: %f seconds\n', seqTime);


fprintf(['Starting strong scaling timing.  ' ...
        'Submitting a total of %d jobs.\n'], numReps*length(numTasks))
for j = 1:length(numTasks)
```

```
    n = numTasks(j);
    strongNumHands = ceil(numHands/n);
    for itr = 1:numReps
        rep(itr) = timeJob(myCluster, n, strongNumHands);
    end
    ind = find([rep.totalTime] == min([rep.totalTime]), 1);
    strong(n) = rep(ind); %#ok<AGROW>
    fprintf('Job wait time with %d task(s): %f seconds\n', ...
            n, strong(n).jobWaitTime);
end

pool = parpool(numWorkers);
parforTime = inf;
strongNumHands = ceil(numHands/numWorkers);
for itr = 1:numReps
    start = tic;
    r = cell(1, numWorkers);
    parfor i = 1:numWorkers
        r{i} = pctdemo_task_blackjack(strongNumHands, 1); %#ok<PFOUS>
    end
    parforTime = min(parforTime, toc(start));
end
delete(pool);

fprintf('Execution time with parfor using %d workers: %f seconds\n', ...
    numWorkers, parforTime);
fprintf(['Speedup based on strong scaling with parfor using ', ...
    '%d workers: %f\n'], numWorkers, seqTime/parforTime);


[###@ln2%tinhe2-C matlab]$
```

提交方法：

```
[###@ln2%tinhe2-C matlab]$cat jobb2.sh
#!/bin/sh
matlab    -nodisplay -r "bench2"
[###@ln2%tinhe2-C matlab]$yhbatch -N 1 jobb2.sh
```

！ 注意这里的 jobb2.sh 没有 –nojvm 选项，不然会出错

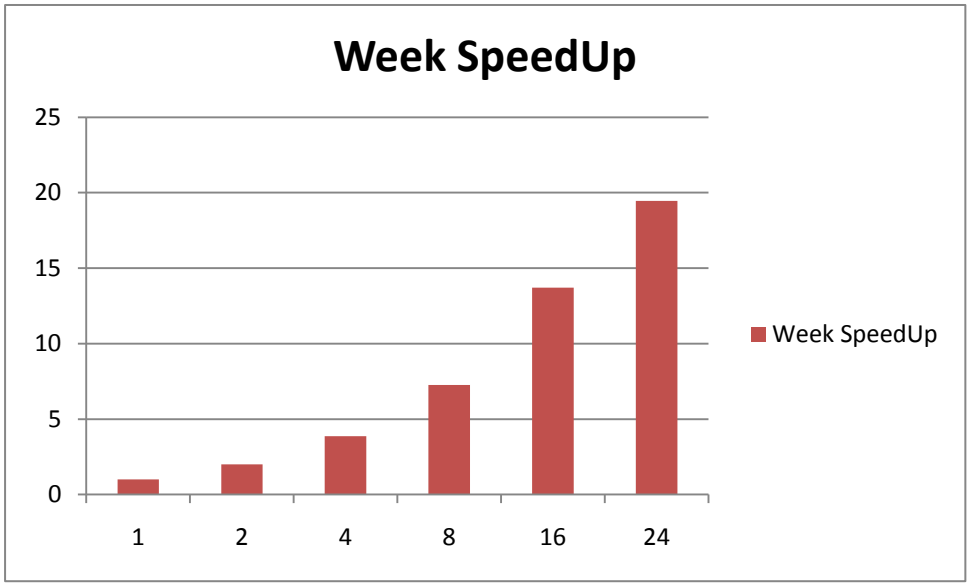
测试结果：

```
Starting weak scaling timing.    Submitting a total of 30 jobs.
Job wait time with 1 task(s): 44.601634 seconds
Job wait time with 2 task(s): 44.642674 seconds
Job wait time with 4 task(s): 46.256516 seconds
```

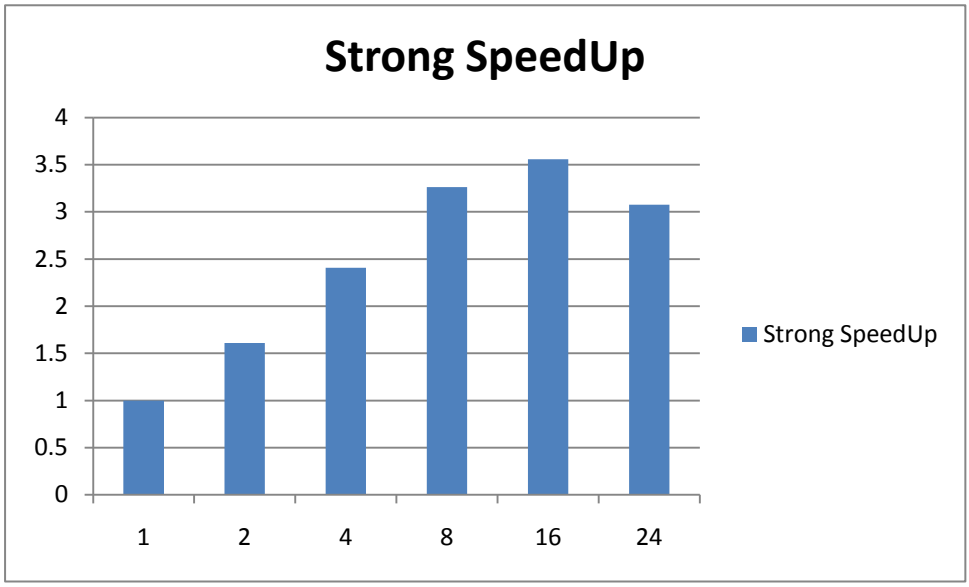
Job wait time with 8 task(s): 49.194334 seconds
Job wait time with 16 task(s): 52.036119 seconds
Job wait time with 24 task(s): 55.009453 seconds
Sequential execution time: 33.664355 seconds

Starting strong scaling timing. Submitting a total of 30 jobs.
Job wait time with 1 task(s): 44.933489 seconds
Job wait time with 2 task(s): 27.929175 seconds
Job wait time with 4 task(s): 18.663919 seconds
Job wait time with 8 task(s): 13.774714 seconds
Job wait time with 16 task(s): 12.632434 seconds
Job wait time with 24 task(s): 14.614805 seconds
Starting parallel pool (parpool) using the 'local' profile ... connected to 24 workers.
Parallel pool using the 'local' profile is shutting down.
Execution time with parfor using 24 workers: 2.137346 seconds
Speedup based on strong scaling with parfor using 24 workers: 15.75

以下是统计图：
弱可扩展性 ：

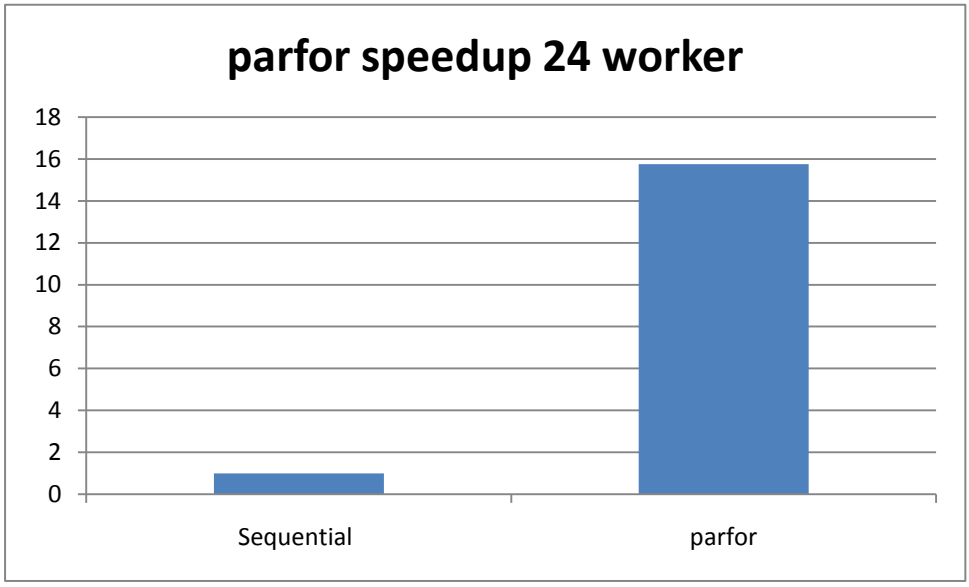


可以看出在 24 核的时候，可以获得接近 20 倍的加速比， 强可扩展性很好，对于计算量足够的任务可以有效利用一个节点的硬件资源。
强可扩展性 ：



可以看出，对于本例，16 核为最优加速比，说明对于计算量不是很大的任务，未必核越多越好。

使用 parfor 进行并行 ：



在本例中，Parfor 获取了接近 16 倍的加速比，还是不错的结果，不过比一般 C/Fortran 计算软件要差 。