



Training Outreach and Education

<http://www.nesc.ac.uk/training>



<http://www.ngs.ac.uk>

# NGS computation services: APIs and Parallel Jobs



<http://www.pparc.ac.uk/>



<http://www.eu-egee.org/>

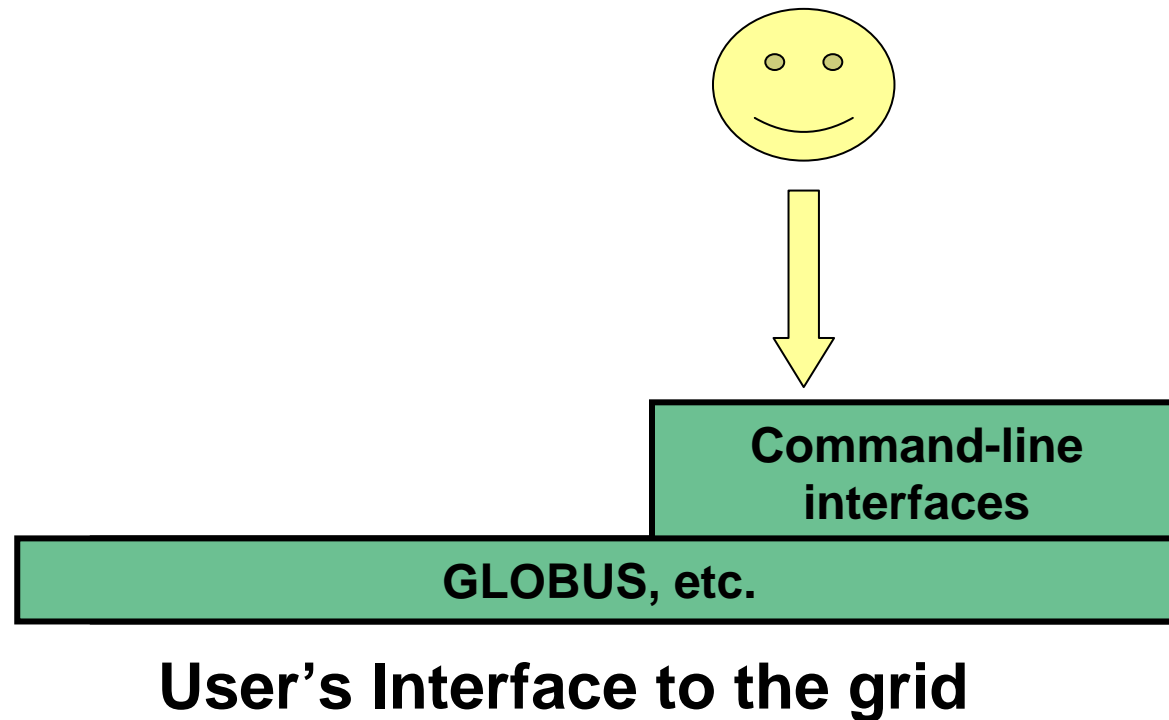
# Policy for re-use

- This presentation can be re-used, in part or in whole, provided its sources are acknowledged.
- However if you re-use a substantial part of this presentation please inform [training-support@nesc.ac.uk](mailto:training-support@nesc.ac.uk). We need to gather statistics of re-use: number of events and number of people trained. Thank you!!

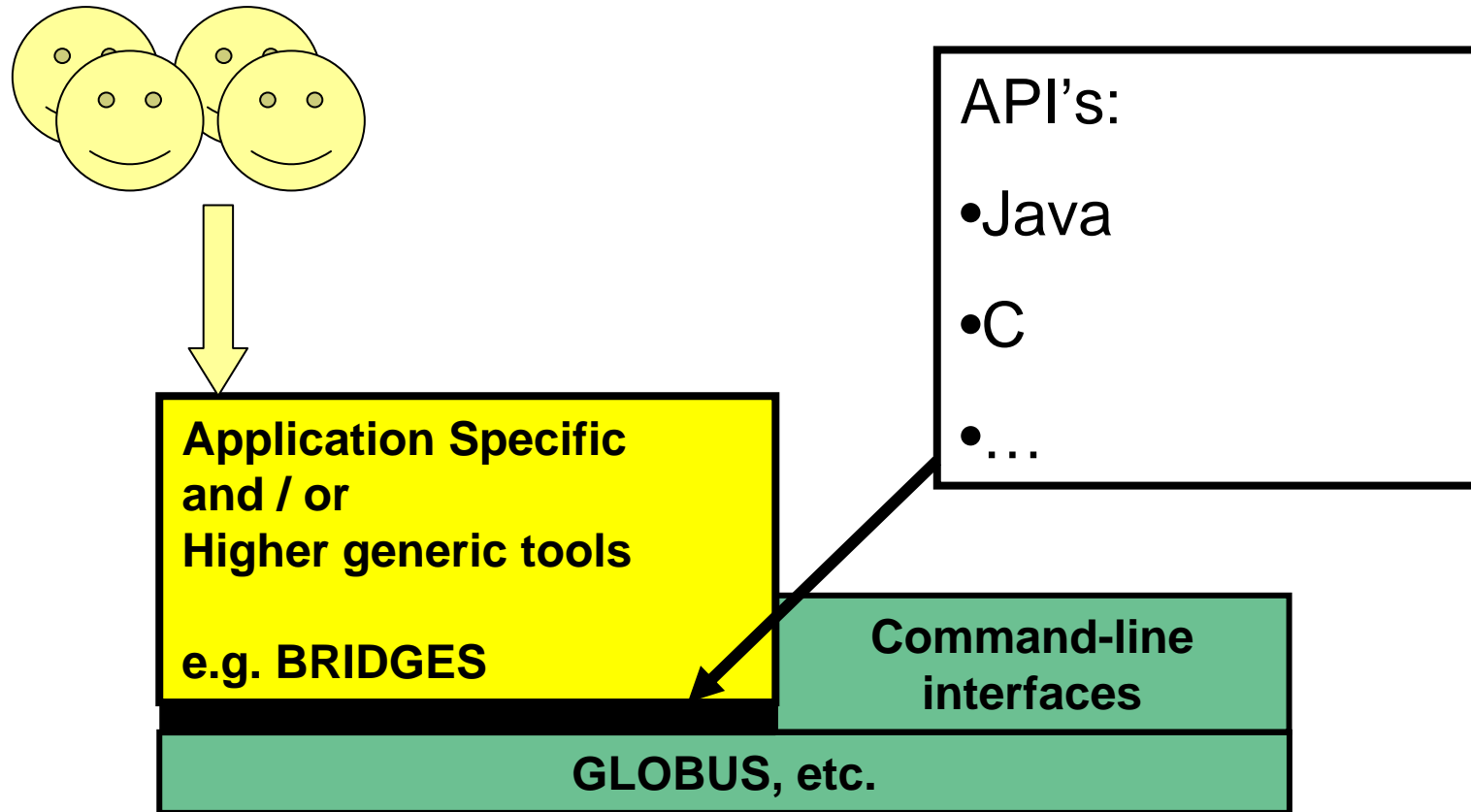
# Overview

- The C and Java API's to the low-level tools
- Using multiple processors

# Job submission so far



# Application-specific tools

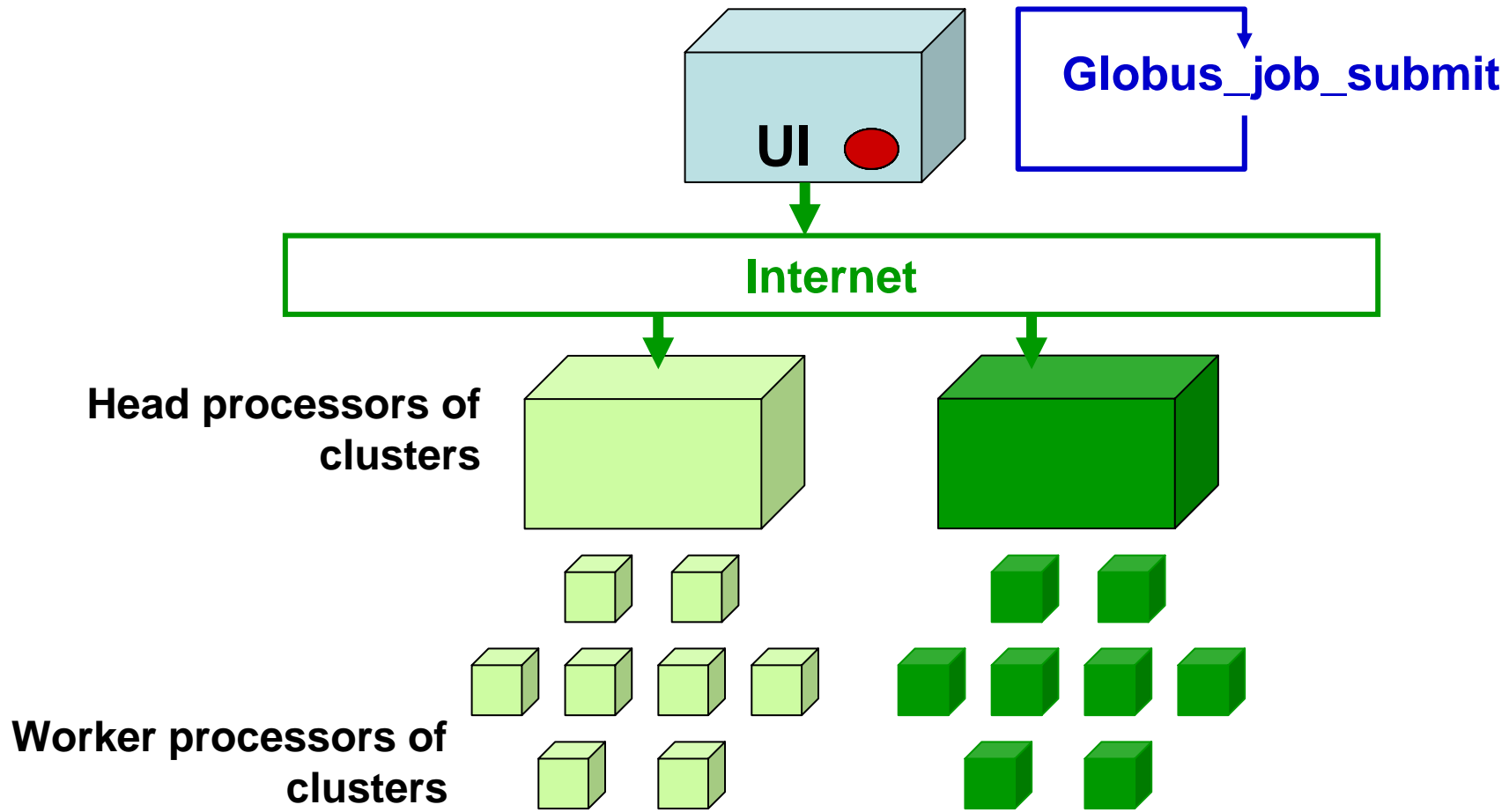


**User's Interface to the grid**

# Available API's

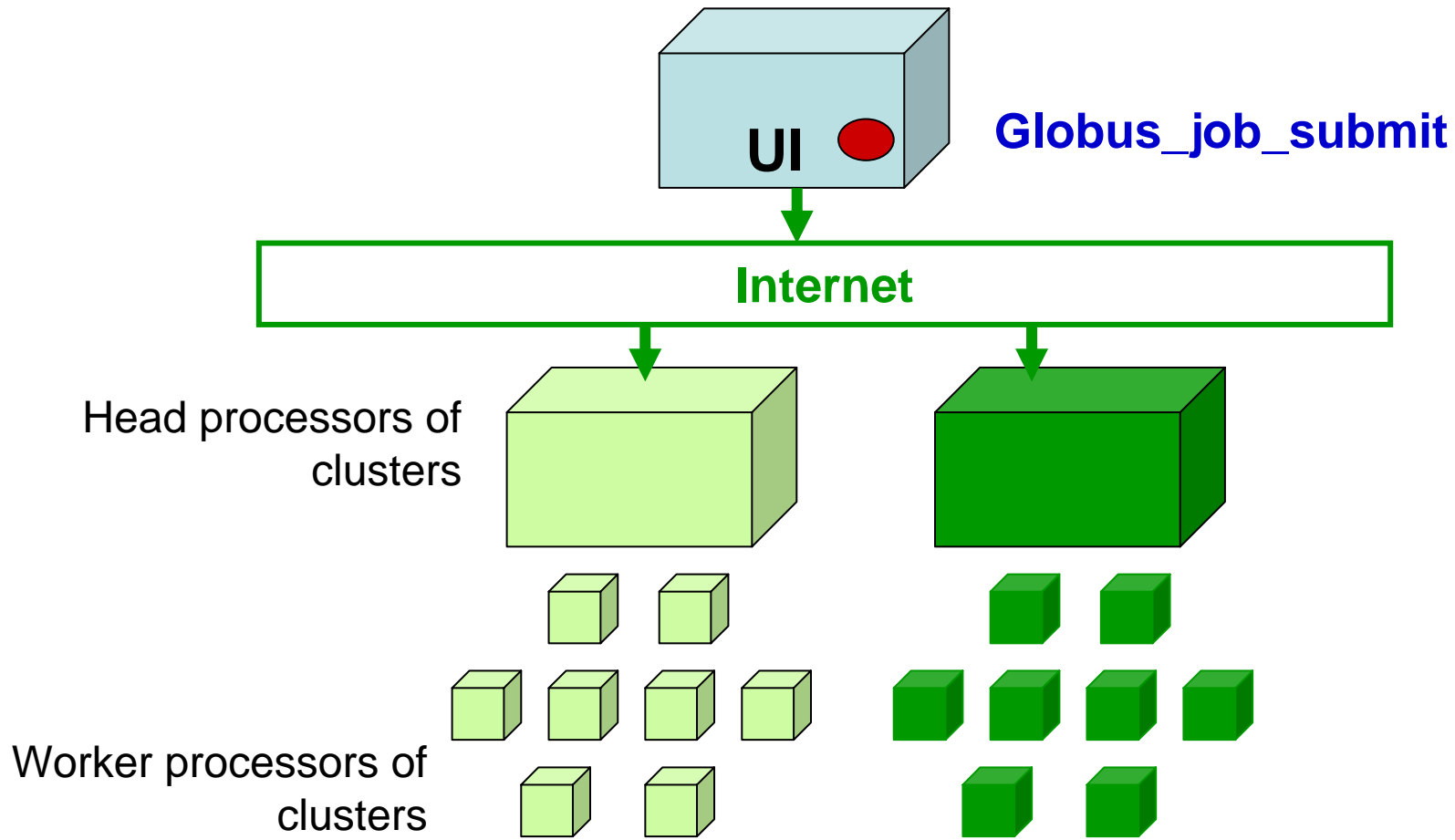
- C <http://www.globus.org/developer/api-reference.html>
- “Community Grid” CoG <http://www.cogkit.org/>
  - Java, Python, Matlab
  - (very limited functionality on Windows – no GSI)

# Non-communicating Processes



Processes run without any communication between them

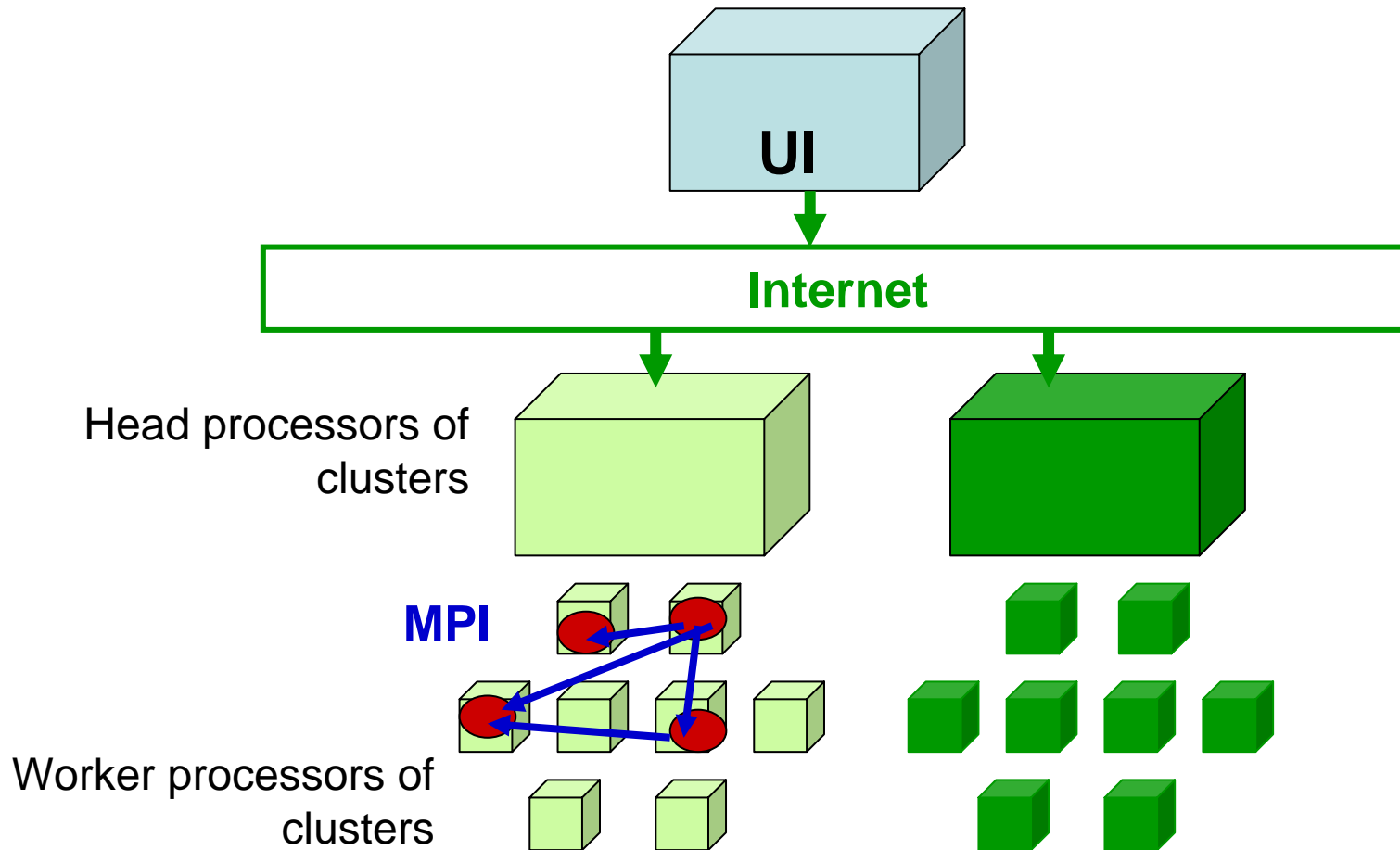
# Communicating Processes



Processes send messages to each other – Must run on same cluster



# Communicating Processes



Processes send messages to each other – Must run on same cluster

# Modes of Parallelism

The NGS nodes open these routes to you – but you have to do a bit of work! (Grid is not magic!...)



- Non-communicating processes: on NGS, multiple executables run from a script on the UI
- Communicating processes: on NGS, you run one globus-job-submit command – but need to code and build program so it is parallelised
  - MPI for distributed memory
  - OpenMP, multithreading – only on a Cardiff node

# Set up for next practical

- If you want to use a graphical text editor then run exceed

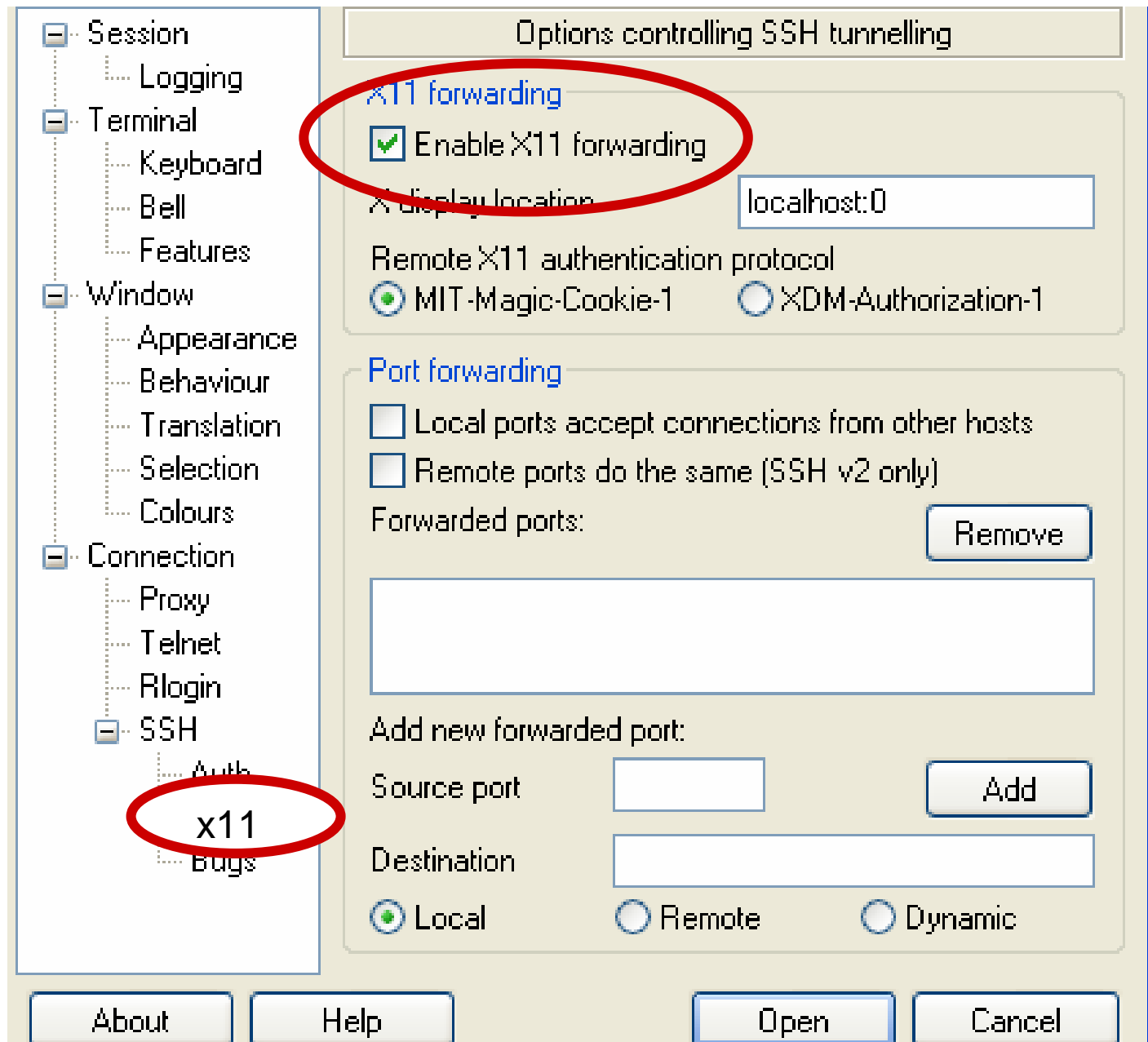
# Run Exceed

- Double-left-click on Hummingbird Connectivity (on desktop)
- Double-Left-click on “Exceed”
- Double-Left-click on “Exceed” shortcut
- Observe flash screen and new task entered in task bar



## How to start putty to enable x11

1. Run exceed
2. Run putty
3. Set X11 before opening session
4. (kwrite editor available)



The screenshot shows the PuTTY Options dialog box. The left sidebar has a tree view with categories: Session, Terminal, Window, Connection, and SSH. The 'x11' option under the SSH category is circled in red. The main panel is titled 'Options controlling SSH tunnelling' and contains two sections: 'X11 forwarding' and 'Port forwarding'. In the 'X11 forwarding' section, the 'Enable X11 forwarding' checkbox is checked and circled in red. Below it, the 'display location' is set to 'localhost:0'. The 'Remote X11 authentication protocol' section has 'MIT-Magic-Cookie-1' selected. The 'Port forwarding' section has two unchecked checkboxes: 'Local ports accept connections from other hosts' and 'Remote ports do the same (SSH v2 only)'. There is a 'Forwarded ports:' list with a 'Remove' button. Below that is an 'Add new forwarded port:' section with fields for 'Source port', 'Destination', and radio buttons for 'Local', 'Remote', and 'Dynamic'. At the bottom of the dialog are buttons for 'About', 'Help', 'Open', and 'Cancel'.

# MPI notes

- How could the task be split into sub-tasks?
  - By functions that could run in parallel??!
  - By sending different subsets of data to different processes?  
More usual ! Overheads of scatter and gather
- Need to design and code carefully: **be alert to**
  - sequential parts of your program (if half your runtime is sequential, speedup will never be more than 2)
  - how load can be balanced (64 processes with 65 tasks will achieve no speedup over 33 processes)
  - Deadlock!
- MPI functions are usually invoked from C, Fortran programs, but also Java
- Several example patterns are given in the practical.  
Many MPI tutorials are on the Web!

# Practical

1. C API Example
  2. Java API usage
  3. Concurrent processing – from Java
  4. MPI example
- Follow link from agenda page