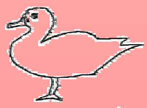
A Web Based Job Submission System for a Physics Computing Cluster

David Jones

THE UNIVERSITY of LIVERPOOL

Introduction & Motivation

- MAP (***M**onte-Carlo **A**rray **P**rocessor*)
- Simplify the process of submitting a computing Job to MAP
- Provide more feedback to the user
- Provide insight into how to turn MAP into a Grid resource



What is MAP?

- **M**onte Carlo **A**rray **P**rocessor
- '*Commodity*' Personal computers mounted in rack –
    - MAP 1 300 nodes
    - MAP 2 1000 nodes
- Wired together with a *private network*
- Work in parallel running same simulation with different parameters
- Good for Physics and Engineering problem solving
- One of the World's most powerful computers
- Concept developed & refined at Liverpool
- Software written and maintained in house; *More to it than just this slide*

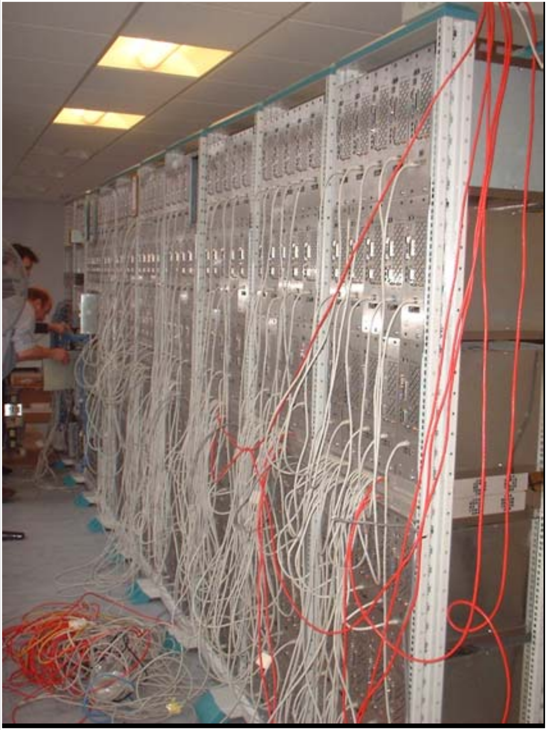
MAP Philosophy

- A scientist sets up a job on their machine
- Once the job is working
  - Job files and libraries are *tarred* into one file
  - A second file is used;-
    - Don’t want 1000 identical jobs running
    - Introduces differences between nodes
    - By, for example, random number initialisation
- A good alternative to batch systems

Pictures MAP and MAP2

MAP 2

- Each Node:
- CPU
    - 3000 GHz
    - Xeon Pentium 4
    - 1GB RAM
- 100TB of direct attached storage
- Slide mounted Nodes



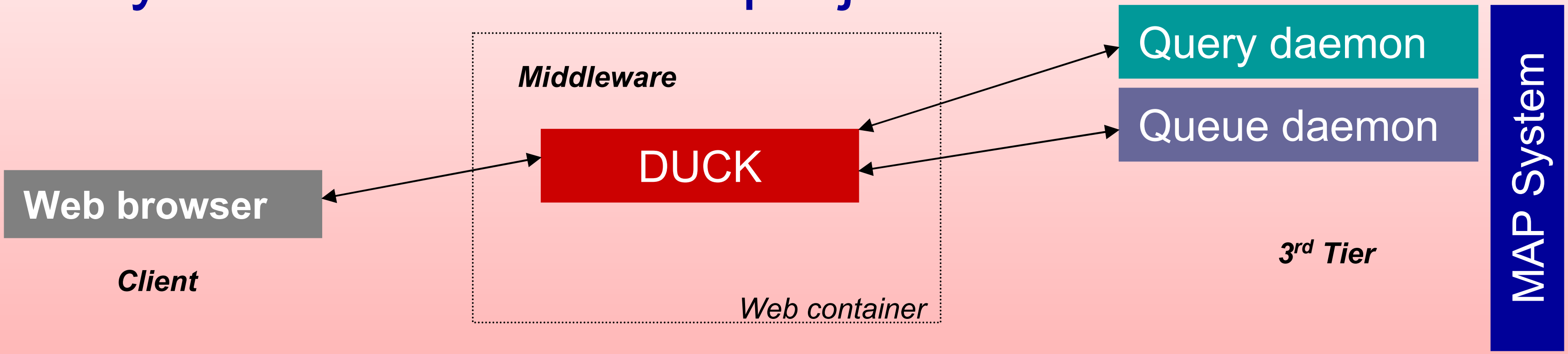
***Lower left**: a bank of MAP 2 nodes.*

***Upper left**: the innards of a MAP2 node*

***Right**: MAP1 with Private network wiring visible*

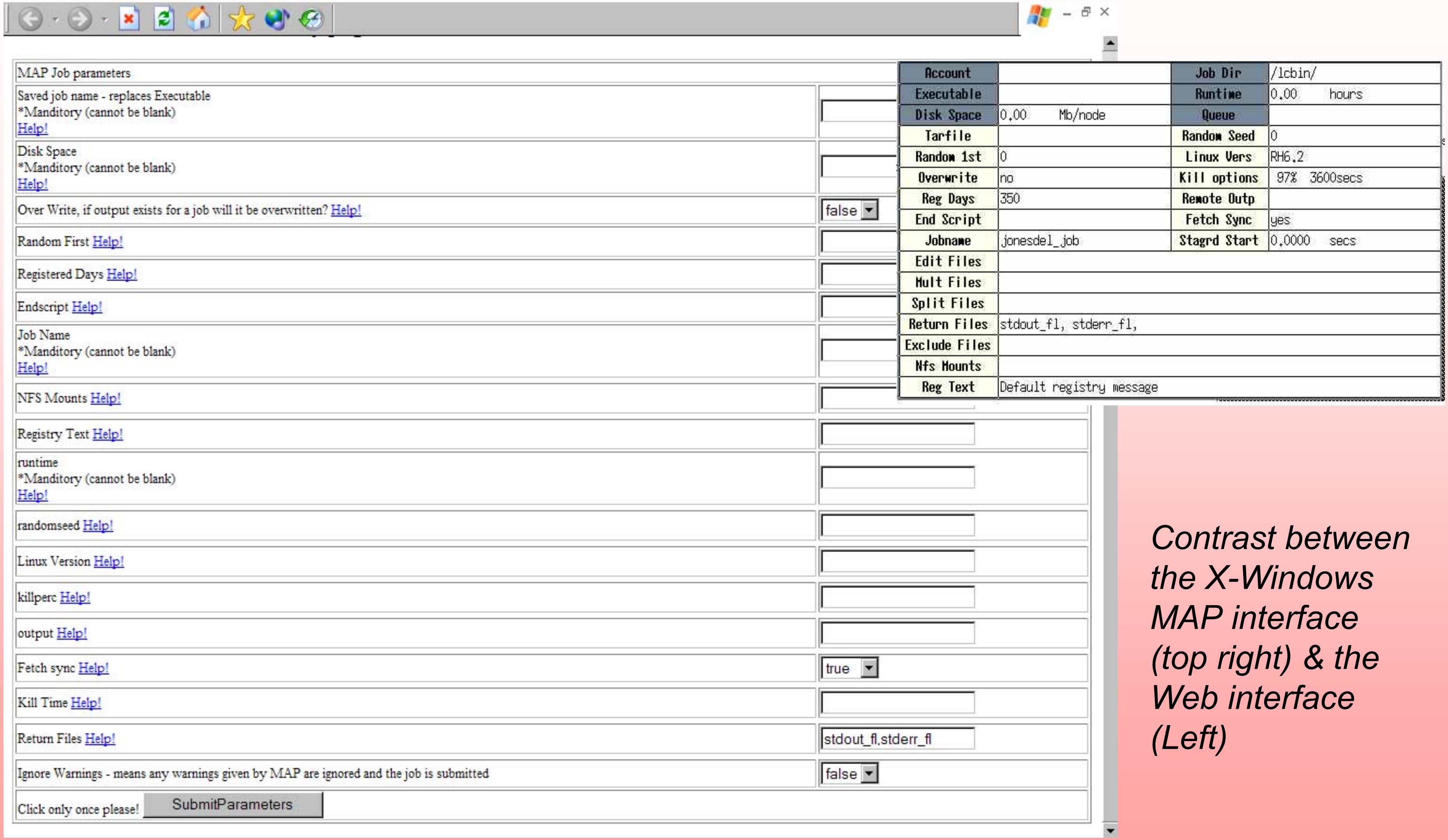
A Web Interface

- Existing Job Submission Tool is MAPXQS
  - An X-Windows based Program
- DUCK (***D**evolved **U**ser **C**ontrol **K**onsole*)
- Submit a MAP job a web browser
- My contribution to the project



- Forms middle part of three tier system
  - Client - User's Web browser
  - DUCK - *Middleware* (A Web application)
  - MAP - The computing resource

User Interfaces



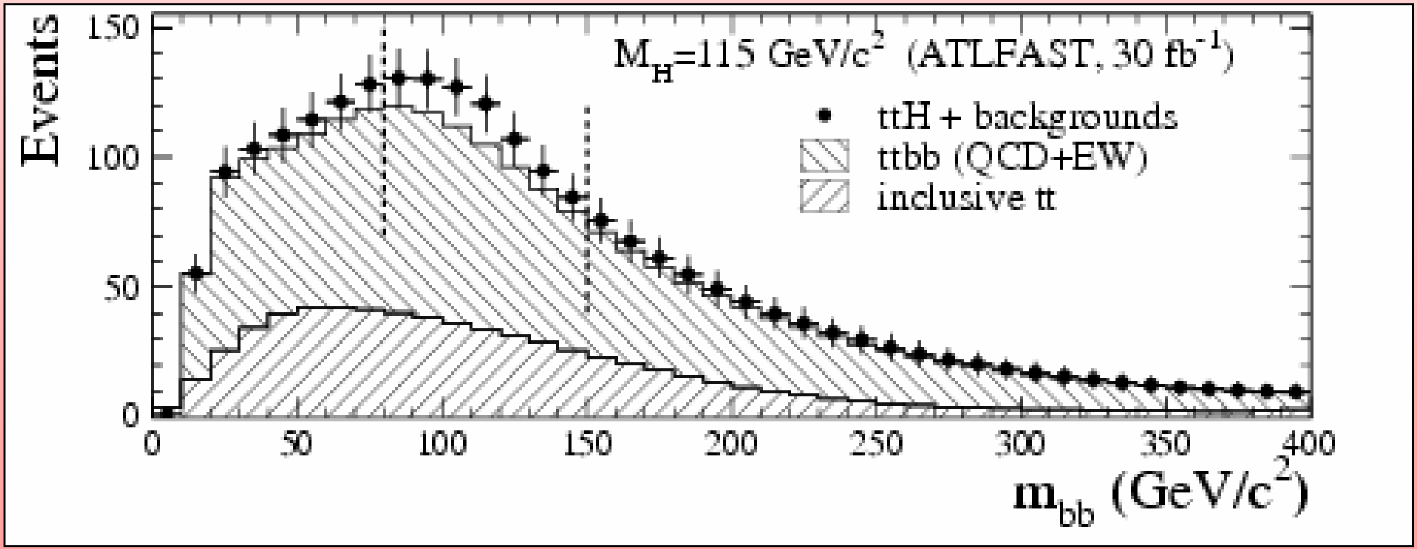
MAP Job parameters	
Saved job name - replaces Executable *Manditory (cannot be blank) Help!	
Disk Space *Manditory (cannot be blank) Help!	
Over Write, if output exists for a job will it be overwritten? Help!	false
Random First Help!	
Registered Days Help!	
Endscript Help!	
Job Name *Manditory (cannot be blank) Help!	
NFS Mounts Help!	
Registry Text Help!	
runtime *Manditory (cannot be blank) Help!	
randomseed Help!	
Linux Version Help!	
killperc Help!	
output Help!	
Fetch sync Help!	true
Kill Time Help!	
Return Files Help!	stdout_fl,stderr_fl
Ignore Warnings - means any warnings given by MAP are ignored and the job is submitted	false
Click only once please! SubmitParameters	

Account		Job Dir	/lcbin/
Executable		Runtime	0.00 hours
Disk Space	0.00 Mb/node	Queue	
Tarfile		Random Seed	0
Random 1st	0	Linux Vers	RH6.2
Overwrite	no	Kill options	97% 3600secs
Reg Days	350	Remote Outp	
End Script		Fetch Sync	yes
Jobname	jonesdel_job	Stagrd Start	0.0000 secs
Edit Files			
Mult Files			
Split Files			
Return Files	stdout_fl, stderr_fl,		
Exclude Files			
Nfs Mounts			
Reg Text	Default registry message		

Contrast between the X-Windows MAP interface (top right) & the Web interface (Left)

Milestones so far

- Successful submission of jobs using DUCK (Atlas Monte-Carlo Production)
  - 200 Million events for MSSM SUSY studies
  - 500 Million events for Higgs studies
  - Generated events used for Physics studies
  - E.g. Search for a light Higgs in ttH channel
- Successfully deployed DUCK on Linux server



Queue Daemon

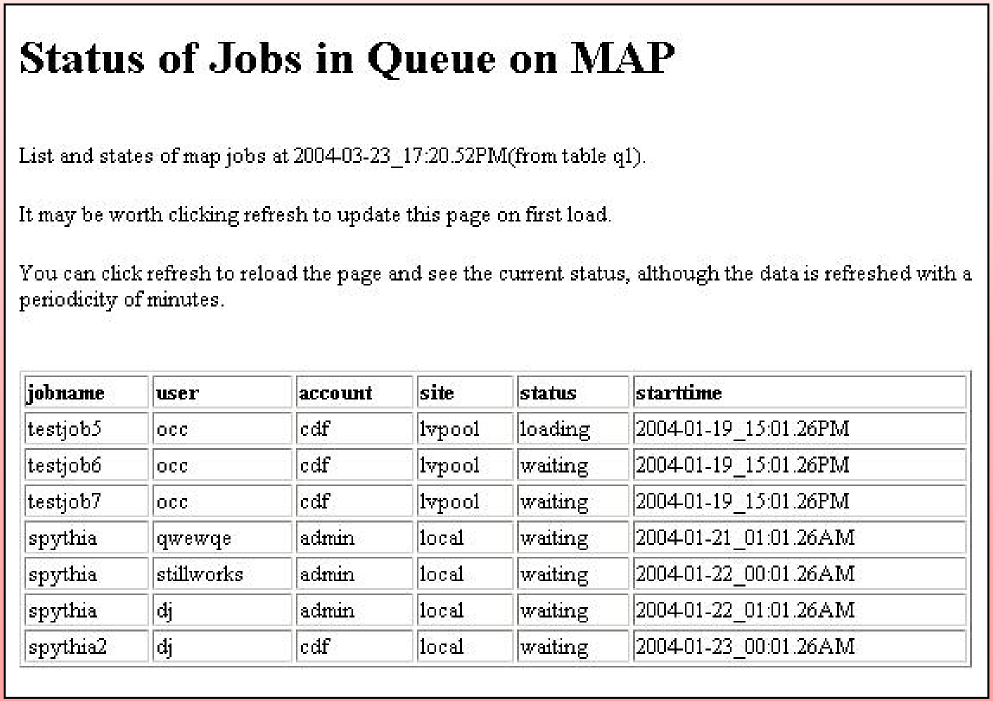
- *Daemon* – a program that runs constantly
- New web queue daemon developed
- Acts as the communication between DUCK and MAP
- *Socket-based* communication of Job parameters;-
- MAP setup parameters
- Application-specific parameters

Application Parameters

- Parameters that describe the application i.e.
  - Physics Parameters
  - Engineering Parameters
- Database system allows jobs with an arbitrary number of application parameters to be specified
- Name, description, default value/ value triplet
- Values are typically random number seeds etc.

Query Daemon

- Allows users to see what jobs are waiting to run on MAP over the web
- Information updated with periodicity of minutes from the MAP *Query daemon*
- Avoids traffic saturation of the MAP software



Status of Jobs in Queue on MAP

List and states of map jobs at 2004-03-23_17:20.52PM(from table q1).

It may be worth clicking refresh to update this page on first load.

You can click refresh to reload the page and see the current status, although the data is refreshed with a periodicity of minutes.

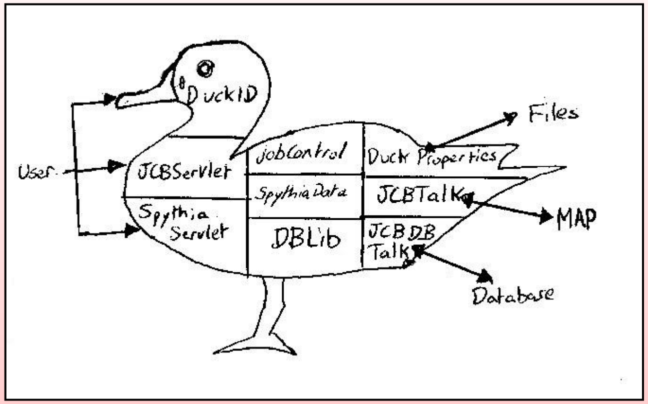
jobname	user	account	site	status	starttime
testjob5	occ	cdf	lvpool	loading	2004-01-19_15:01.26PM
testjob6	occ	cdf	lvpool	waiting	2004-01-19_15:01.26PM
testjob7	occ	cdf	lvpool	waiting	2004-01-19_15:01.26PM
spythia	qwewqe	admin	local	waiting	2004-01-21_01:01.26AM
spythia	stillworks	admin	local	waiting	2004-01-22_00:01.26AM
spythia	dj	admin	local	waiting	2004-01-22_01:01.26AM
spythia2	dj	cdf	local	waiting	2004-01-23_00:01.26AM

Inside DUCK - Java

- Core MAP software written in C for performance reasons
- DUCK written in Java
- Takes advantage of Java’s
  - Database bindings
  - Networking packages
  - Web delivery mechanism (servlets)
- *Servlet engine* Tomcat (Apache Project product)

Inside DUCK - Servlets

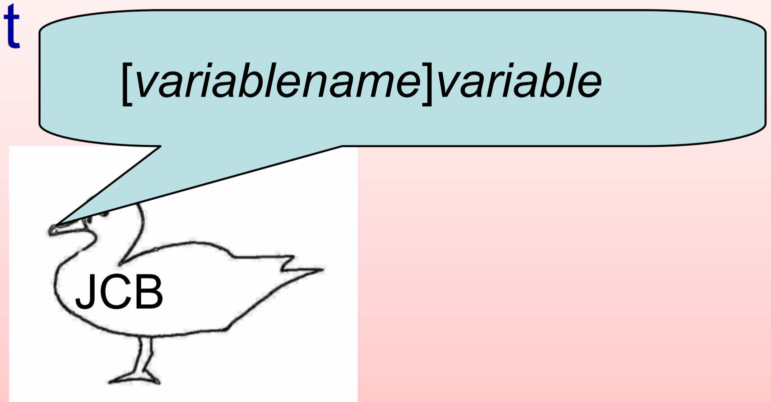
- DUCK is made up of multiple Servlets
- Servlet are
  - small programs
  - that produce web pages as output
  - as response to a request from a web browser



- Client (web browser) doesn't store data between pages
  - MySQL database provides a *persistency mechanism*

Inside DUCK – MAP Interface

- Connection between DUCK and MAP is a simple text protocol over a Network Socket
- Currently two listeners
  - Queue
  - Query
- Queue data is sent in the form
  - **`[variablename]variable`**
- Messages sent back at the end of communication in 3 groups
  - *Errors* – things that stop a MAP job running
  - *Warnings* – things that are possibly a problem
  - *Infos* – Information about the job



Concluding

- DUCK is;
  - Middleware for MAP
  - deployed on Linux (developed on a win32 platform)
  - In action; Users are submitting jobs using the interface (Atlas)
- There are features to be added to the interface if time permits
- Can already see ways to ‘Grid-ify’ the middleware