Q²ADPZ
['Kwod pi: 'si:]

D/I/S
Division of Intelligent Systems
AI & Learning

* QADPZ *
An Open System for Distributed Computing

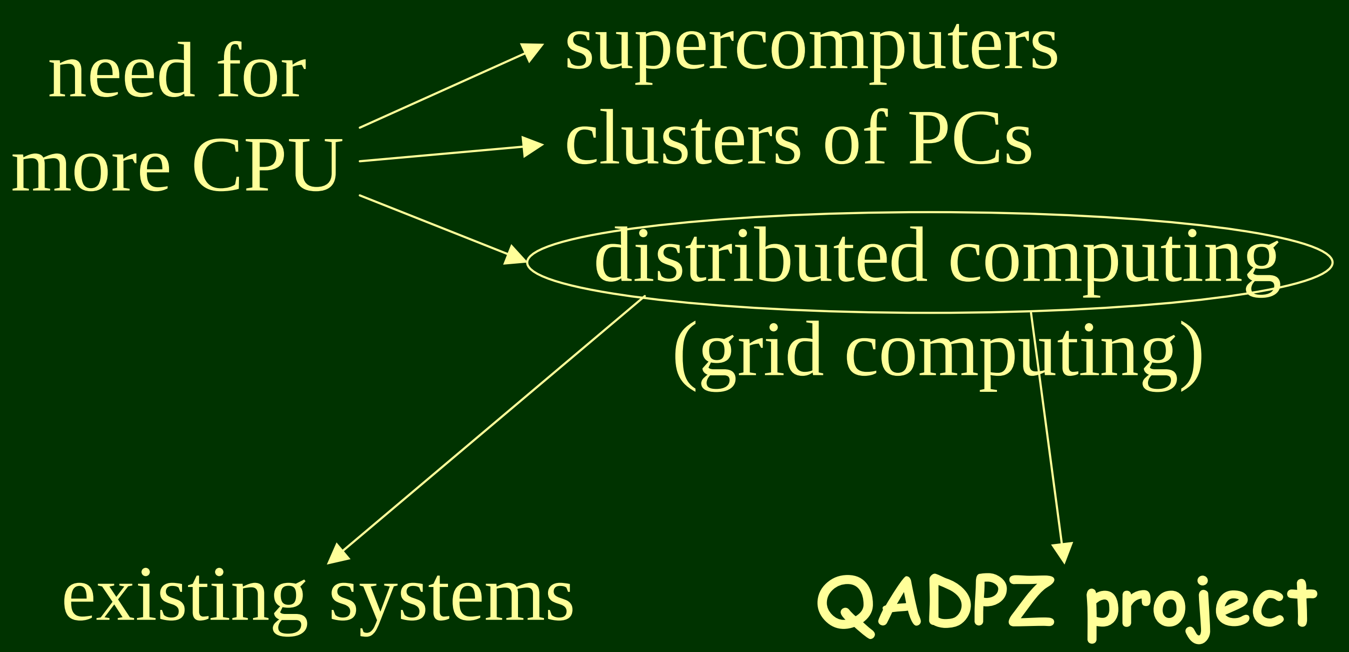
Zoran Constantinescu

zoran@idi.ntnu.no

http://qadpz.sourceforge.net

16-Jan-2003

outline



need for more CPU

supercomputers

clusters of PCs

distributed computing (grid computing)

existing systems

QADPZ project

- description
- advantages
- architecture
- application domains
- current status
- future work

problem(s)

- larger and larger amounts of data are generated every day (simulations, measurements, etc.)
- software applications used for handling this data are requiring more and more CPU power
  - simulation, visualization, data processing
- complex algorithms, e.g. evolutionary algorithms
  - large populations, evaluations very time consuming

à need for parallel processing

solution1

- use parallel supercomputers
  - access to tens/hundreds of CPUs

    e.g. NOTUR/NTNU embla (512) + gridur (384)
  - high speed interconnect, shared memory
  - usually for batch mode processing
  - very expensive (price, maintenance, upgrade)
  - these CPUs are not so powerful anymore

    e.g. 500 MHz RISC vs. 2.4 GHz Pentium4

solution2

- use clusters of PCs (Beowulf)
  - network of personal computers
  - usually running Linux operating system
  - powerful CPUs (Pentium3/4, Athlon)
  - high speed networking (100 MBps, 1 GBps, Myrinet)
  - much cheaper than supercomputers
  - still quite expensive (upgrade, maintenance)
  - trade higher availability and/or greater performance for lower cost

solution3

- use distributed computing
  - using existing networks of workstations (PCs connected by LAN from labs, offices, etc.)
  - usually running Windows or Linux operating system (also MacOS, Solaris, IRIX, etc.)
  - powerful CPUs (Pentium3/4, Athlon)
  - high speed networking (100 MBps)
  - already installed computers – very cheap
  - easy to have a network of tens/hundreds of computers

distributed computing

- specialized client applications run on each individual computer
- they talk to one or more central servers
- download a task, solve it, and send back results
- more suited (easier) for task-parallel applications

  (where the applic. can be decomposed into independent tasks)
- can also be used for data-parallel applications
- the number of available CPUs is more dynamic

existing systems

- seti@home
  - search for extraterrestrial intelligence
  - analysis of data from radio telescopes
  - client application is very specialized
  - using the Internet to connect clients to server, and to download/upload a task
  - no framework for other applications
  - no source code available

existing systems

- distributed.net
  - one of the largest "computer" in the world (~20TFlops)
  - used for solving computational challenges:
    - RC5, Optimal Golomb ruler
  - client application is very specialized
  - using the Internet to connect clients to server
  - no framework for other applications
  - no source code available

existing systems

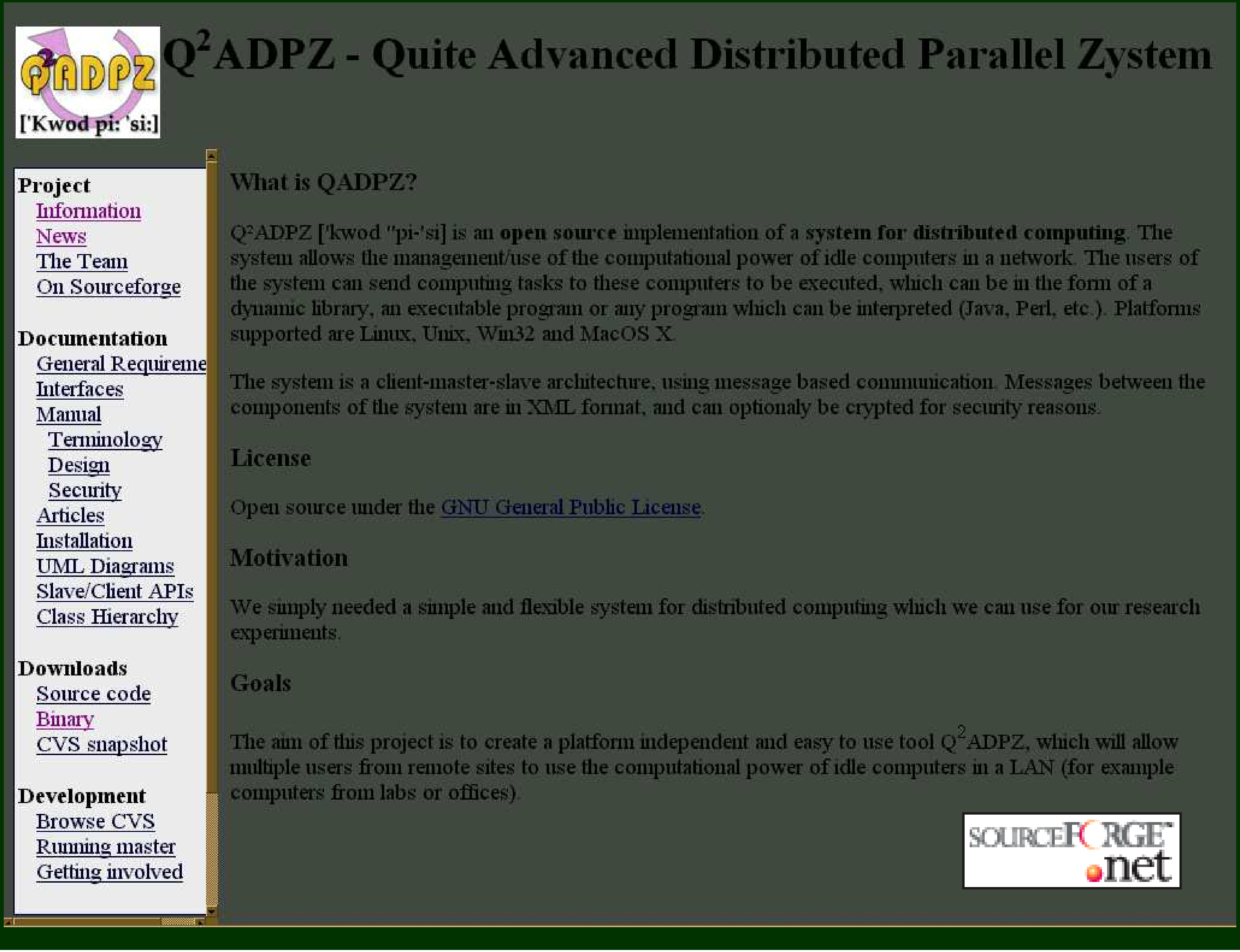
- Condor project (Univ.of Wisconsin)
  - – more research oriented computational projects
  - – more advanced features, user applications
  - – very difficult to install, problems with some OSes (started from a Unix environment)
  - – restrictive license (closed system)
- other commercial projects
  - Entropia, Parabon

['Kwod pi: 'si:]

a new system

- QADPZ project (NTNU)
  - initial application domains: large scale visualization, genetic algorithms, neural networks
  - prototype in early 2001, but abandoned (too viz oriented)
  - started in July 2001, first release v0.1 in Aug 2001
  - we are now close to release v0.8 (Feb-Mar 2003)
  - system independent of any specific application domain
  - open source project on SourceForge.net

qadpz.sourceforge.net



Q^2ADPZ - Quite Advanced Distributed Parallel Zystem

What is QADPZ?

Q^2ADPZ ['kwod "pi-'si] is an **open source** implementation of a **system for distributed computing**. The system allows the management/use of the computational power of idle computers in a network. The users of the system can send computing tasks to these computers to be executed, which can be in the form of a dynamic library, an executable program or any program which can be interpreted (Java, Perl, etc.). Platforms supported are Linux, Unix, Win32 and MacOS X.

The system is a client-master-slave architecture, using message based communication. Messages between the components of the system are in XML format, and can optionaly be crypted for security reasons.

License

Open source under the GNU General Public License.

Motivation

We simply needed a simple and flexible system for distributed computing which we can use for our research experiments.

Goals

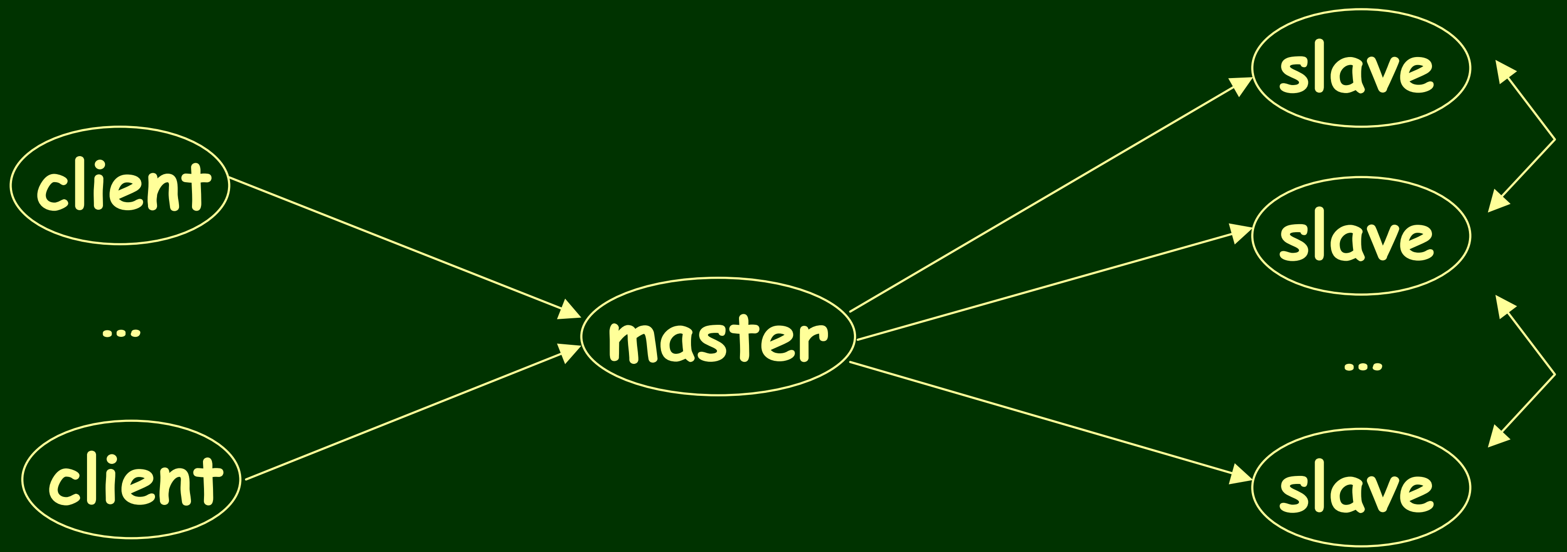
The aim of this project is to create a platform independent and easy to use tool Q^2ADPZ, which will allow multiple users from remote sites to use the computational power of idle computers in a LAN (for example computers from labs or offices).

QADPZ description

- QADPZ project (NTNU)
  - similar in many ways to Condor (submit computing tasks to idle computers running in a network)
  - easy to install, use, and maintain
  - modular and extensible
  - open source project, implemented in C++
  - support for many OSes (Linux, Windows, Unix, …)
  - support for multiple users, encryption
  - logging and statistics

QADPZ architecture

client à master à slave



- management of slaves
- scheduling tasks
- monitoring tasks
- controlling tasks

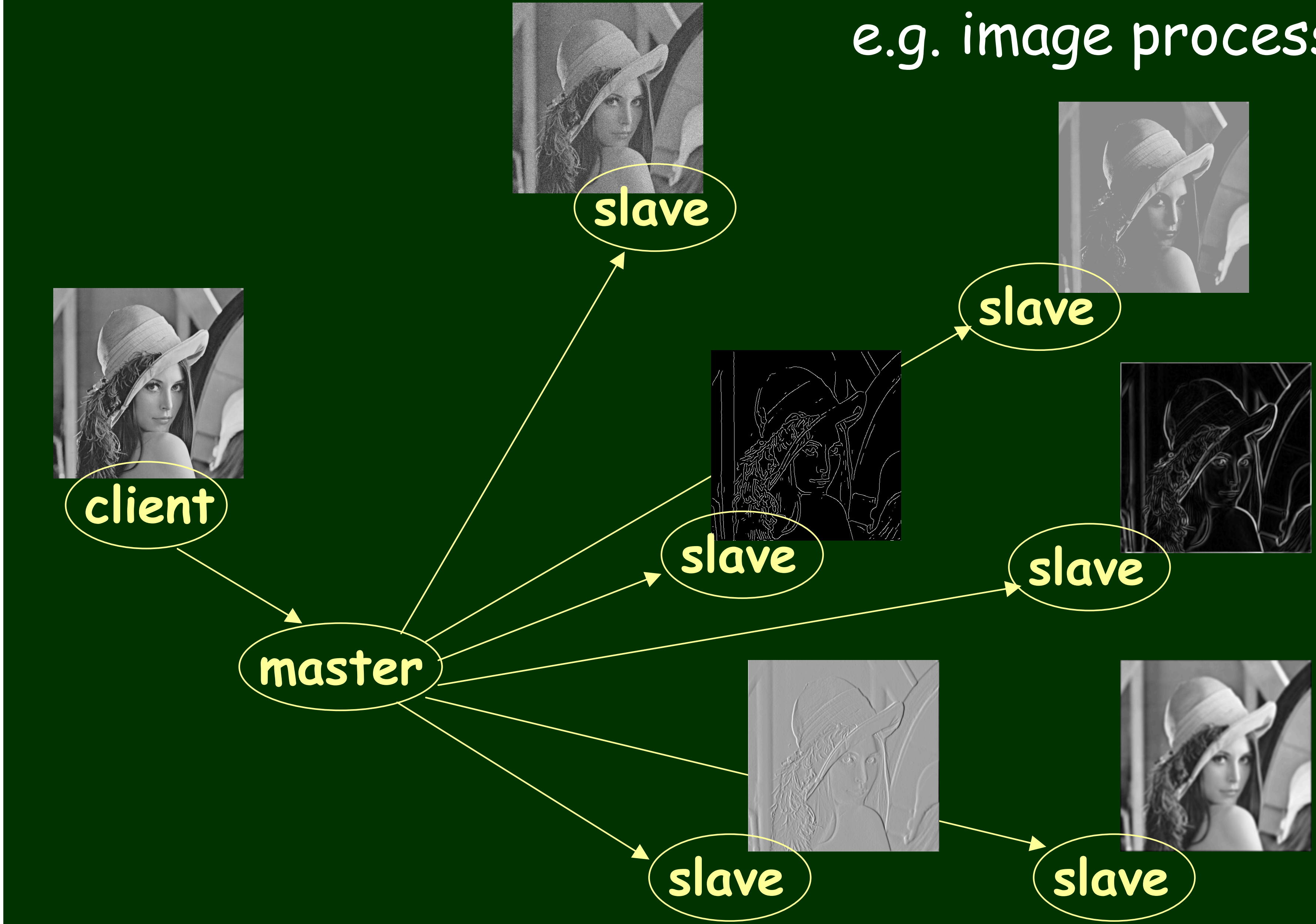
- user interface
- submit tasks/data

- background process
- download tasks/data
- executing tasks

parallelism

- task-parallelism ("coarse grain")
  - multiple independent code segments/programs are run concurrently
  - same initial data or different
  - same code or different
- data-parallelism ("fine grain")
  - same code runs concurrently on different data elements
  - usually requires synchronization (better network)

e.g. image processing



client

master

slave

slave

slave

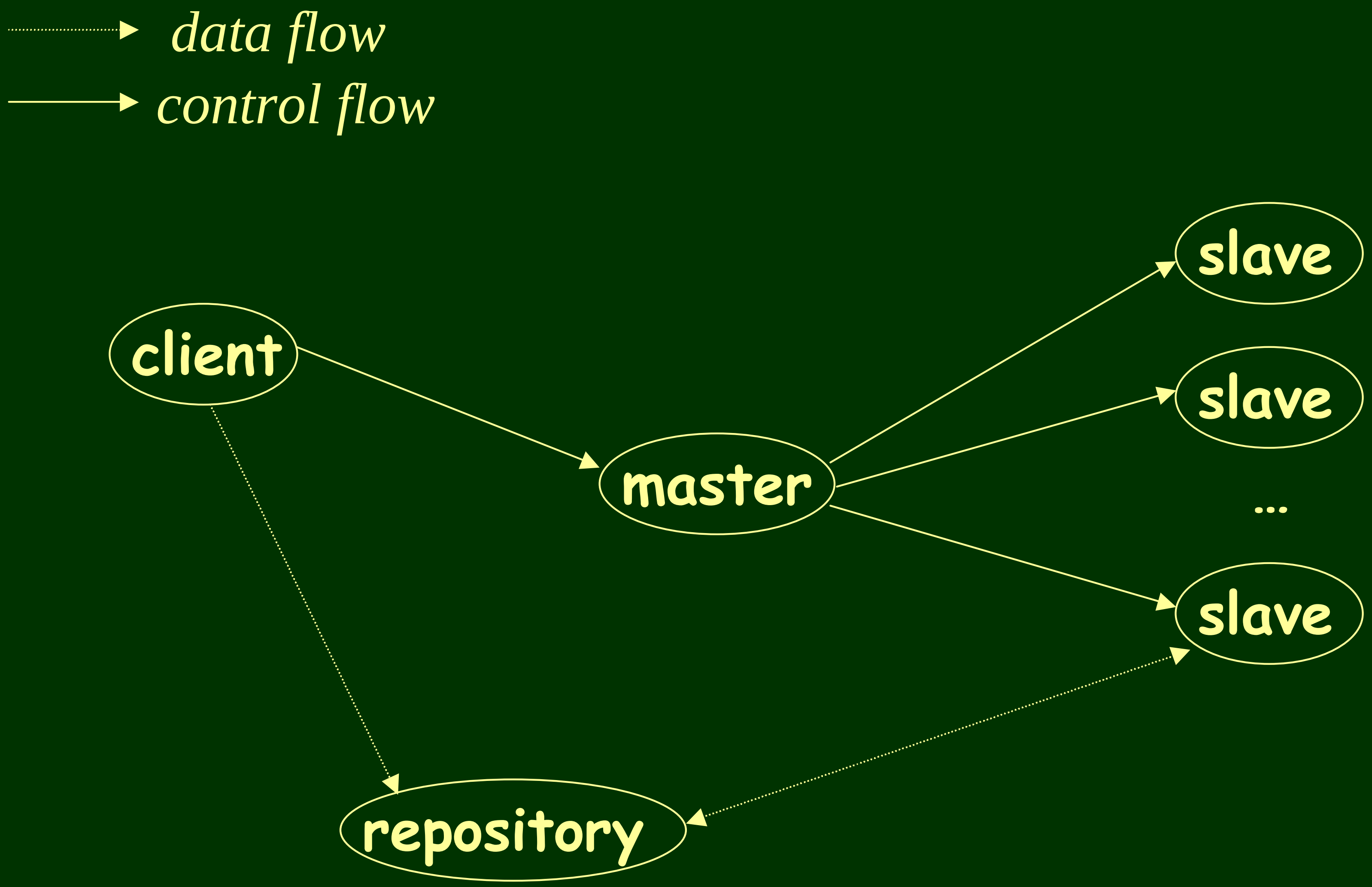
slave

slave

slave

task-parallelism

data



- external web server, ftp server
- internal lightweight web server

client

- can be automatic or manual (user)
- describe project file
- prepares task code
- prepares input files
- submit the tasks
- either wait for the results (stay connected to master), or detach from the tasks and get results later (master will keep all messages)

client programming

- basic level
  - the user has an executable to be run on multiple comps
  - uses our generic client to submit tasks
- intermediate level
  - submission script (XML interface) is changed
- advanced level
  - user writes his own client application using our API
- hacker level
  - modifies QADPZ source code for extra functionality

à *see manual*

jobs, tasks, subtasks

- **job:**
  - consists of groups of tasks executed sequentially or in parallel
  - a task can consist of subtasks (same executable is run but with different input data) – for parallel tasks
  - each task is submitted individually, the user specifies which OS and min. resource requirements (disk, mem)
  - the master allocates the most suitable slave for executing the tasks and notifies the client
  - when task is finished, results are stored as specified and the client is notified

the tasks

- regular binary executable code
    - no modifications required
    - must be compiled for each of the platforms
- regular interpreted program
    - shell script, Perl, Python
    - Java program
    - requires interpreter/VM on each slave
- dynamically loaded slave library (our API)
    - better performance
    - more flexibility

e.g. job description

```
<Job Name="brick">
    <Task ID="1" Type="Executable">
        <RunCount>15</RunCount>
        <FilesURL>http://server/cgi-bin/</FilesURL>
        <TaskInfo>
            <TimeOut>7200</TimeOut>
            <OS>Win32</OS>
            <CPU Speed="500">i386</CPU>
            <Memory>64</Memory>
            <Disk>5</Disk>
            <URL>http://server/slave_app.dll</URL>
            <Executable Type="File">../bin/evolve.exe</Executable>
            <CmdLine>sphere.prj 2 50</CmdLine>
        </TaskInfo>
        <InputFile>sph/sphere.txt</InputFile>
        <OutputFile>sph/layout.txt</OutputFile>
    </Task>
</Job>
```

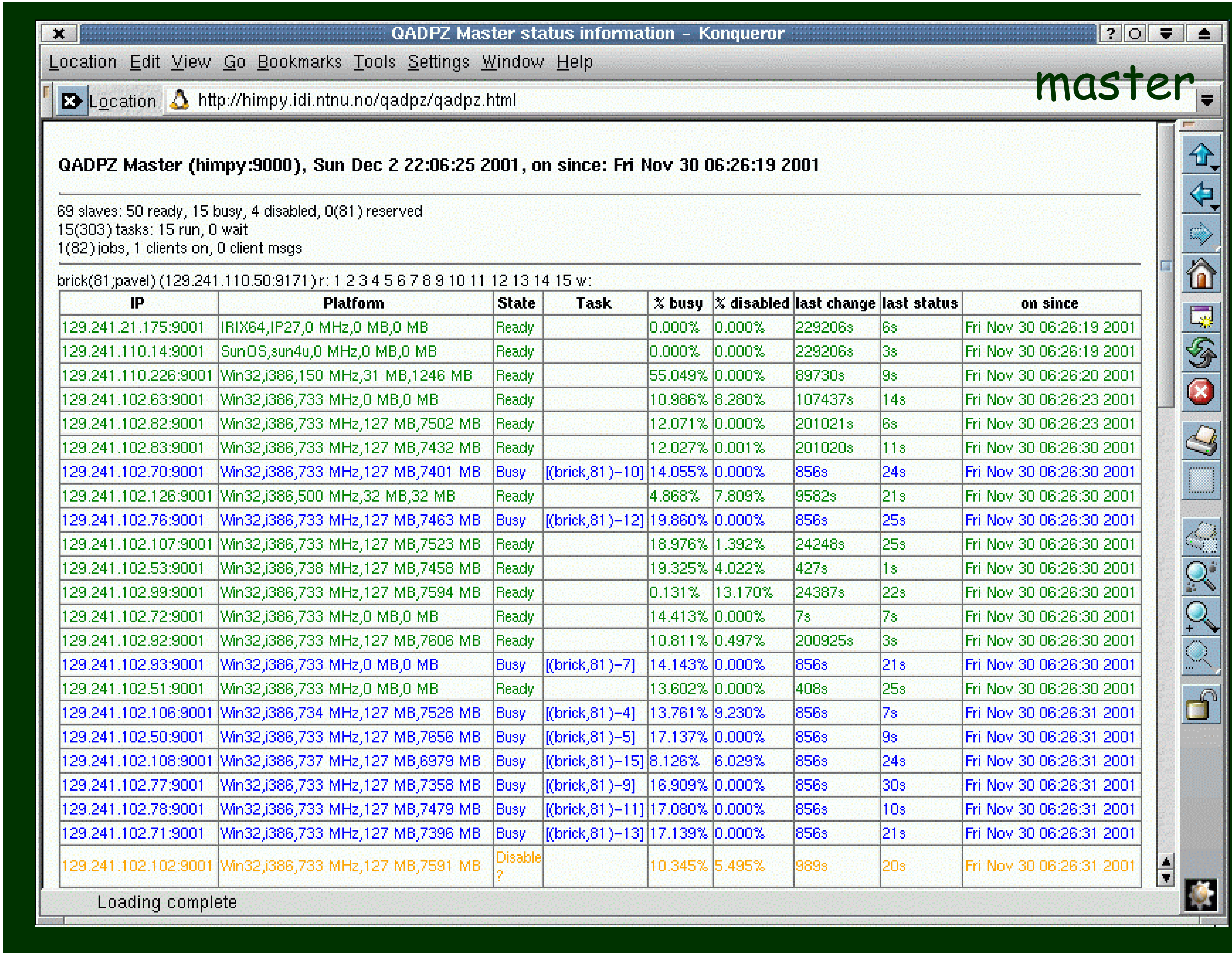
the master

- keeps account of all existing slaves (status, specifications)
- usually one master is enough
- more can be used if there are too many slaves (communication protocol allows one master to act as another client, but not fully implemented yet)
- keeps account of all submitted jobs/tasks
- keeps a list of accepted users (based on username/passwd)
- gathers statistics about slaves, tasks
- can optionally run the internal web server for the repository

installation

- one of our computer labs (Rose salen)
    - ~80 PCs Pentium3, 733 MHz, 128 MBytes
    - dual boot: Win2000 and FreeBSD
    - running for several month
    - when a student logs in into the computer, the slave running on that computer is set into disable mode (no new computing tasks are accepted, any current tasks in killed and/or restarted on another comp.)
    - obtained results worth weeks of computation in just a couple of days

master

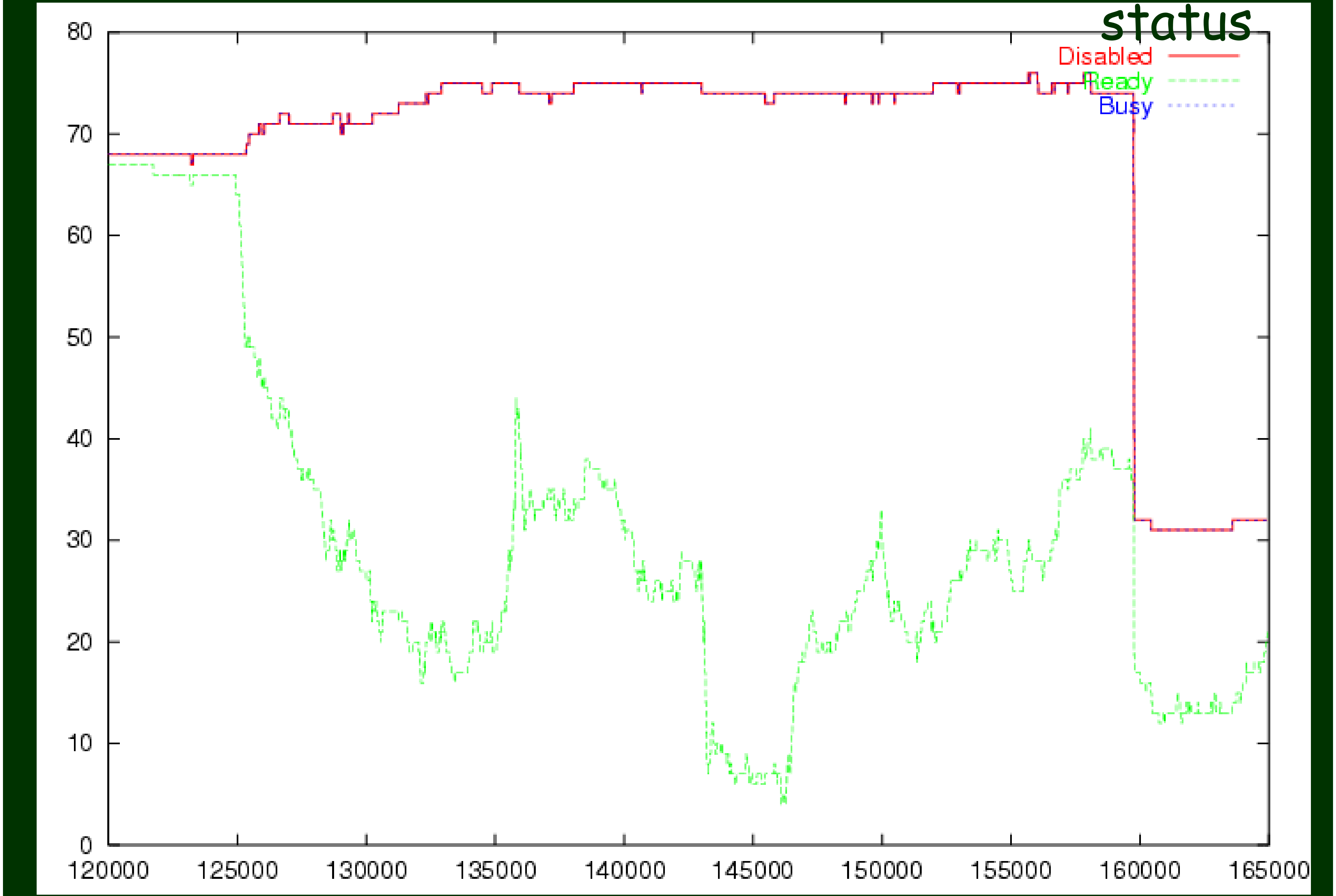

QADPZ Master status information - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Location http://himpy.idi.ntnu.no/qadpz/qadpz.html
Loading complete

QADPZ Master (himpy:9000), Sun Dec 2 22:06:25 2001, on since: Fri Nov 30 06:26:19 2001

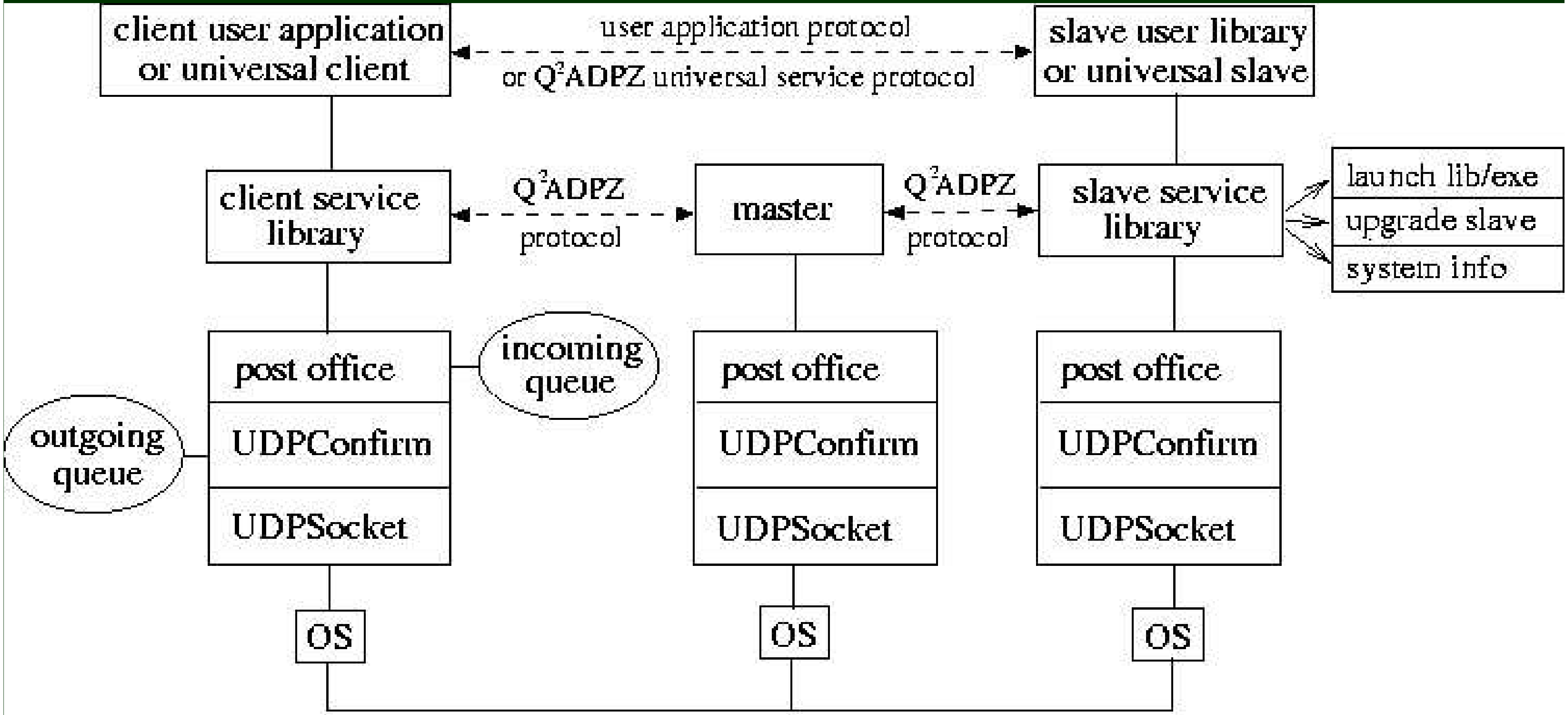
69 slaves: 50 ready, 15 busy, 4 disabled, 0(81) reserved
15(303) tasks: 15 run, 0 wait
1(82) jobs, 1 clients on, 0 client msgs

brick(81;pavel) (129.241.110.50:9171) r: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 w:

IP	Platform	State	Task	% busy	% disabled	last change	last status	on since
129.241.21.175:9001	IRIX64,IP27,0 MHz,0 MB,0 MB	Ready		0.000%	0.000%	229206s	6s	Fri Nov 30 06:26:19 2001
129.241.110.14:9001	SunOS,sun4u,0 MHz,0 MB,0 MB	Ready		0.000%	0.000%	229206s	3s	Fri Nov 30 06:26:19 2001
129.241.110.226:9001	Win32,i386,150 MHz,31 MB,1246 MB	Ready		55.049%	0.000%	89730s	9s	Fri Nov 30 06:26:20 2001
129.241.102.63:9001	Win32,i386,733 MHz,0 MB,0 MB	Ready		10.986%	8.280%	107437s	14s	Fri Nov 30 06:26:23 2001
129.241.102.82:9001	Win32,i386,733 MHz,127 MB,7502 MB	Ready		12.071%	0.000%	201021s	6s	Fri Nov 30 06:26:23 2001
129.241.102.83:9001	Win32,i386,733 MHz,127 MB,7432 MB	Ready		12.027%	0.001%	201020s	11s	Fri Nov 30 06:26:30 2001
129.241.102.70:9001	Win32,i386,733 MHz,127 MB,7401 MB	Busy	[(brick,81)-10]	14.055%	0.000%	856s	24s	Fri Nov 30 06:26:30 2001
129.241.102.126:9001	Win32,i386,500 MHz,32 MB,32 MB	Ready		4.868%	7.809%	9582s	21s	Fri Nov 30 06:26:30 2001
129.241.102.76:9001	Win32,i386,733 MHz,127 MB,7463 MB	Busy	[(brick,81)-12]	19.860%	0.000%	856s	25s	Fri Nov 30 06:26:30 2001
129.241.102.107:9001	Win32,i386,733 MHz,127 MB,7523 MB	Ready		18.976%	1.392%	24248s	25s	Fri Nov 30 06:26:30 2001
129.241.102.53:9001	Win32,i386,738 MHz,127 MB,7458 MB	Ready		19.325%	4.022%	427s	1s	Fri Nov 30 06:26:30 2001
129.241.102.99:9001	Win32,i386,733 MHz,127 MB,7594 MB	Ready		0.131%	13.170%	24387s	22s	Fri Nov 30 06:26:30 2001
129.241.102.72:9001	Win32,i386,733 MHz,0 MB,0 MB	Ready		14.413%	0.000%	7s	7s	Fri Nov 30 06:26:30 2001
129.241.102.92:9001	Win32,i386,733 MHz,127 MB,7606 MB	Ready		10.811%	0.497%	200925s	3s	Fri Nov 30 06:26:30 2001
129.241.102.93:9001	Win32,i386,733 MHz,0 MB,0 MB	Busy	[(brick,81)-7]	14.143%	0.000%	856s	21s	Fri Nov 30 06:26:30 2001
129.241.102.51:9001	Win32,i386,733 MHz,0 MB,0 MB	Ready		13.602%	0.000%	408s	25s	Fri Nov 30 06:26:30 2001
129.241.102.106:9001	Win32,i386,734 MHz,127 MB,7528 MB	Busy	[(brick,81)-4]	13.761%	9.230%	856s	7s	Fri Nov 30 06:26:31 2001
129.241.102.50:9001	Win32,i386,733 MHz,127 MB,7656 MB	Busy	[(brick,81)-5]	17.137%	0.000%	856s	9s	Fri Nov 30 06:26:31 2001
129.241.102.108:9001	Win32,i386,737 MHz,127 MB,6979 MB	Busy	[(brick,81)-15]	8.126%	6.029%	856s	24s	Fri Nov 30 06:26:31 2001
129.241.102.77:9001	Win32,i386,733 MHz,127 MB,7358 MB	Busy	[(brick,81)-9]	16.909%	0.000%	856s	30s	Fri Nov 30 06:26:31 2001
129.241.102.78:9001	Win32,i386,733 MHz,127 MB,7479 MB	Busy	[(brick,81)-11]	17.080%	0.000%	856s	10s	Fri Nov 30 06:26:31 2001
129.241.102.71:9001	Win32,i386,733 MHz,127 MB,7396 MB	Busy	[(brick,81)-13]	17.139%	0.000%	856s	21s	Fri Nov 30 06:26:31 2001
129.241.102.102:9001	Win32,i386,733 MHz,127 MB,7591 MB	Disable ?		10.345%	5.495%	989s	20s	Fri Nov 30 06:26:31 2001

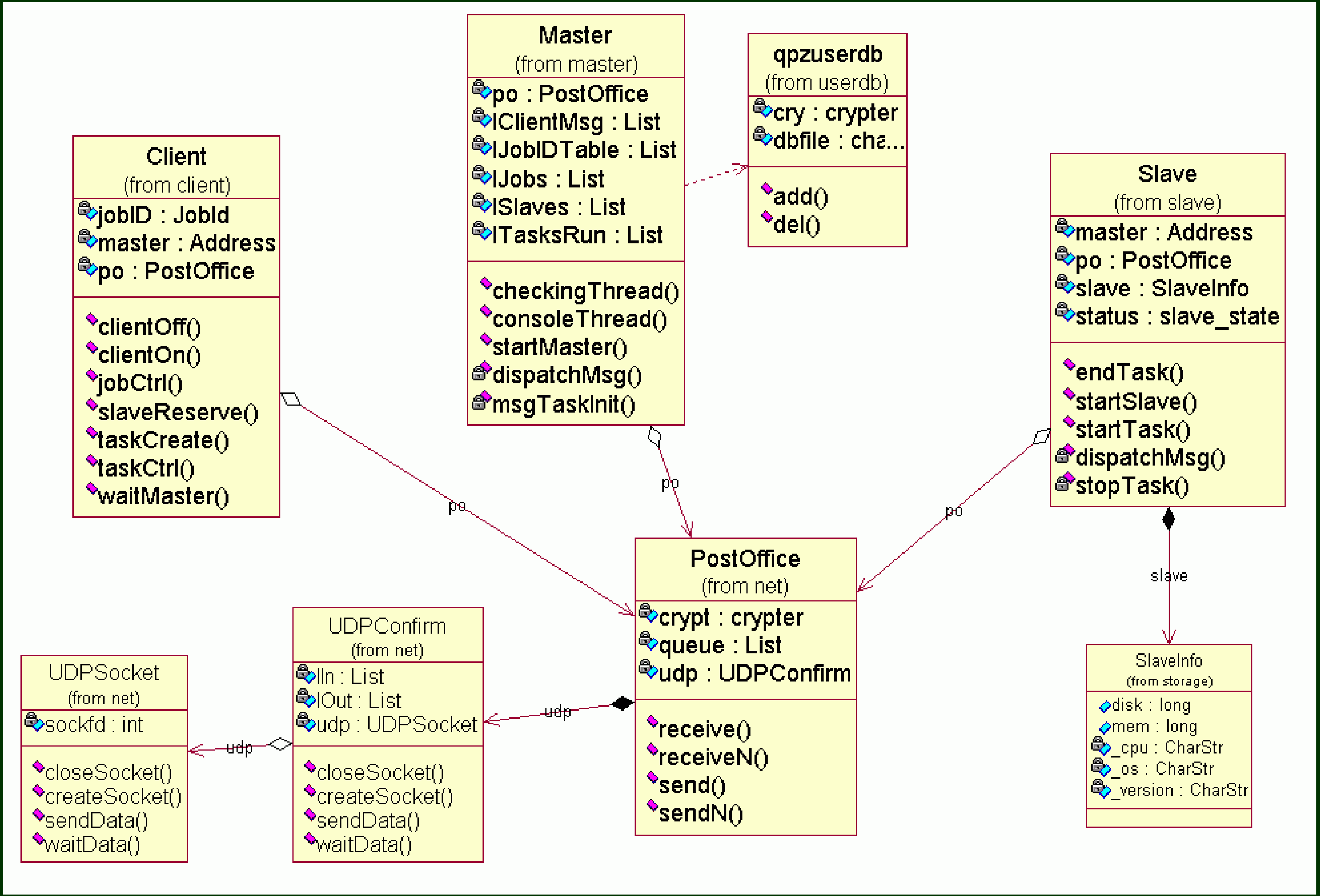

status
Disabled
Ready
Busy
0
10
20
30
40
50
60
70
80
120000
125000
130000
135000
140000
145000
150000
155000
160000
165000

communication



- layered communication protocol
- exchanged messages are in XML (w/ compress+encrypt)
- uses UDP with a reliable layer on top

OO design



Client
(from client)
jobID : JobId
master : Address
po : PostOffice
clientOff()
clientOn()
jobCtrl()
slaveReserve()
taskCreate()
taskCtrl()
waitMaster()

Master
(from master)
po : PostOffice
lClientMsg : List
lJobIDTable : List
lJobs : List
lSlaves : List
lTasksRun : List
checkingThread()
consoleThread()
startMaster()
dispatchMsg()
msgTaskInit()

qpzuserdb
(from userdb)
cry : crypter
dbfile : cha...
add()
del()

Slave
(from slave)
master : Address
po : PostOffice
slave : SlaveInfo
status : slave_state
endTask()
startSlave()
startTask()
dispatchMsg()
stopTask()

UDPSocket
(from net)
sockfd : int
closeSocket()
createSocket()
sendData()
waitData()

UDPConfirm
(from net)
lIn : List
lOut : List
udp : UDPSocket
closeSocket()
createSocket()
sendData()
waitData()

PostOffice
(from net)
crypt : crypter
queue : List
udp : UDPConfirm
receive()
receiveN()
send()
sendN()

SlaveInfo
(from storage)
disk : long
mem : long
_cpu : CharStr
_os : CharStr
_version : CharStr

installation

- **slave**
  - qadpz_slave (daemon) + slave.cfg
- **master**
  - qadpz_master (daemon) + master.cfg +
  - qadpz_admin + users.txt + privkey
- **client**
  - qadpz_run + client.cfg + pubkey

Linux, Win32 (9x,2K,XP), SunOS,
IRIX, FreeBSD, Darwin MacOSX

future work

- local caching of executables on the slaves
- different scheduling protocols on master
- web interface to the client
    - creating jobs easier, with input data
    - starting/stopping jobs
    - monitoring execution of jobs
    - easy access to the output of execution
    - should decrease learning effort for using the system

the team

- QADPZ =
  - Atle Pedersen
  - Diego Federici
  - Pavel Petrovic
  - Zoran Constantinescu

  from the Division of Intelligent Systems (DIS)

  http://www.idi.ntnu.no/seksjoner/dis

thank you

?