ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

# Shedding some light into the black box

## An overview of the batch system of Brutus

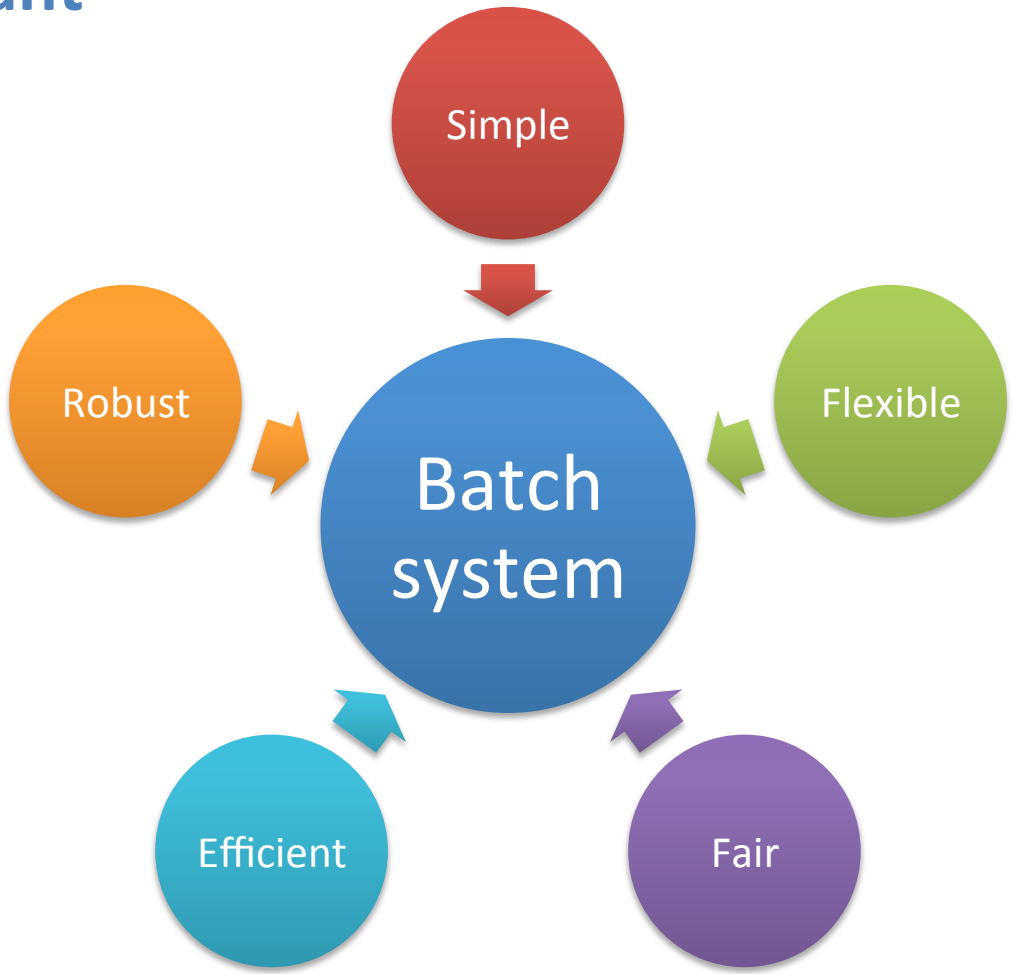
Olivier Byrde

*Head of High Performance Computing, ETH Zurich*

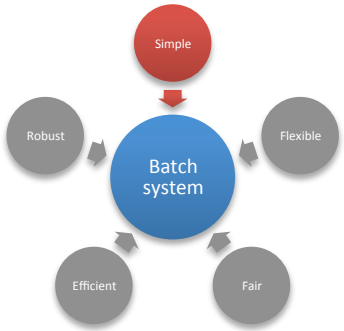
# Outline

- What we want
- What we use
- The black box
- Queues
- Host groups
- Job allocation
- Summary
- Final words

# What we want

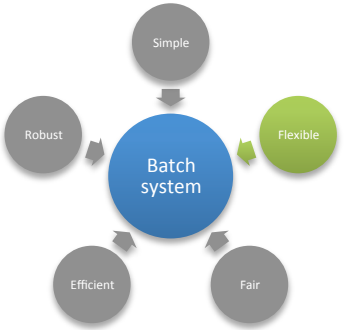


# Simple



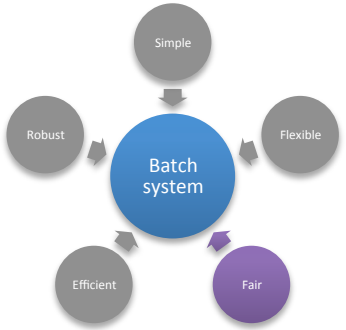

- Simple to use
  - People must use the batch system to run computations
  - The vast majority have never used a batch system before
  - Many have never used Linux or a command-line interface either
  - The system must adapt to these people — not the other way around
- Simple to manage
  - Most batch systems are highly complex pieces of software
  - Every new release adds more features
  - In a small team, everyone must understand how the system works
  - It is best to use only a small subset of features, and use them well

# Flexible



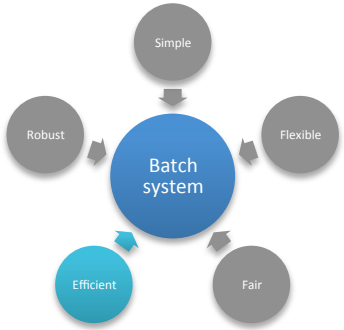

- Everything changes
  - Hardware: compute nodes, GPUs, networks, storage
  - Software: compilers, libraries, applications
  - People: new users, groups, shareholders
  - Usage patterns
- The batch system must be able to follow these changes
  - Seamless integration of new hardware & technologies
  - Some customization may be needed for 3rd party applications
  - Policies must follow users' needs — not the other way around
  - Different settings for day/night, week/weekend, holidays, etc.

# Fair



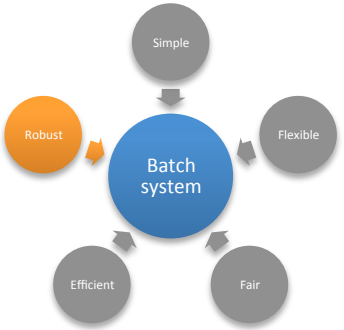

- Resources must be allocated in a fair manner
  - The Brutus cluster is co-owned by over 30 groups
  - Each group must receive a share of CPU time proportional to their investment
  - Each user within a group must also receive a fair share of CPU time
- Two categories of users
  - VIP users (also known as “shareholders”)
  - Normal users (aka “guests”)

# Efficient




- The Brutus cluster is highly heterogeneous
  - 10 kinds of compute nodes
  - Ethernet, Quadrics and InfiniBand networks
  - Local scratch, NFS (SAN/NAS), PanFS and Lustre storage
  - Nvidia and AMD GPUs
- Some nodes are tailored for certain applications
  - Large memory nodes
  - GPU nodes
- The batch system must ensure that jobs are allocated to the right nodes in the most efficient way
  - Each job should get the resources it needs… but not more!
  - Unused resources are wasted

# Robust




- The batch system must be able handle a huge number of jobs without flinching
  (Brutus: over 2 million jobs/month)
- No job should be lost as a result of a malfunction of the batch system itself
- Good support is needed when things go wrong

# What we use

- Platform LSF 7.0 (update 6)
  - Users and groups
  - Hosts and resources
  - Queues and limits
  - Priorities
  - Accounting
- Quadrics RMS 3.1
  - Resource Management System for the Quadrics QsNet$^{II}$ network
  - Works together with LSF
  - Will be removed later this year when we decommission the network
- Custom-made “black box”
  - Analyzes a job’s requirements (explicit or implicit)
  - Tells LSF where and how it should be executed

# Why Platform LSF

- The IT Services of ETH decided many years ago (2003?) to use the same scheduler for all central systems
  - Platform LSF was selected (I can't tell you why, I was not involved in this project)
  - It has been used on our two HP Superdome systems, *Stardust* and *Pegasus*, as well as *Hreidar*, *Gonzales*, *Obelix* and other clusters
- The HPC group has accumulated a lot of experience with LSF over the years, we're quite comfortable with it
- It is not cheap, but *you get what you pay for* ;-)

# Why Platform LSF (cont.)

- Pros
  - Industry standard, compatible with many 3rd party applications
  - Designed from the ground up for heterogeneous systems
  - More features than you can dream of — is there anything it *cannot* do?
  - Relatively open architecture, easy to customize
  - Mature software, rock solid
  - Professional support — troubleshooting, bug fixes, custom builds, etc.
  - Comprehensive documentation
- Cons
  - Not originally designed for HPC, workstation heritage still visible
  - Too many features — with so many ways to do things, how do you know which one is best? (answer: you have to try them all!)
  - Configuration and administration can be quite baffling at first
  - Commercial software, not open source

# Why Quadrics RMS

- Licensed with the QsNet$^{II}$ network of the old *Gonzales* cluster
  - Installed in 2004, expanded in 2005, integrated into Brutus in 2007
  - Will be decommissioned later this year
- Pros
  - Designed from the ground up for MPP systems
  - Control everything related to the QsNet$^{II}$ network — hard- and software, resources, jobs, performance data, etc.
  - All information is stored in a database, easy to query and modify via SQL
  - Works nicely with LSF
- Cons
  - Commercial software, not open source
  - Many sites chose not to use it for licensing reasons (hence SLURM)
  - Quadrics Ltd is no longer in business; no maintenance, no support

# Why a black box

- Users want to run computations on the cluster, not learn the details of a batch system
  - Even those who know these details will eventually make mistakes and submit their jobs incorrectly
- We want to be able to change the configuration of the batch system at any time
  - Easier to do if this configuration is hidden from the users

# Black box: basic principles

- Queues are hidden from the users
  - Actually they are visible, but irrelevant for job submission
- Users indicate the basic resources needed by their jobs
  - Number of processors — 1 by default
  - Run-time — 1 hour (wall-clock) by default
  - Memory — 1 GB/core by default
- Users may also indicate special requirements
  - Number and kind (model) of CPU cores per node
  - Number and kind (vendor, type, model) of GPUs per node
  - Storage — local scratch size; access to Panasas, Lustre or NAS
  - Dependency condition(s)
  - …

# Examples

- Basic jobs
  - bsub < script.sh — *1-hour serial job*
  - bsub –W 168:00 ./program arg1 arg2 — *7-day serial job*
  - bsub **matlab** –r matlab_prog.m — *Matlab serial job*
  - bsub –n 16 "**fluent** 3ddp < fluent.in > fluent.out" — *Fluent parallel job*
  - bsub –n 128 –W 24:00 **mpirun** ./mpi_program — *1-day MPI job*
- Special requirements
  - bsub –n 2 –R "span[**ptile**=1]" mpirun ./pingpong — *1 core per node*
  - bsub –n 128 –R "select[**model**==Opteron8384] span[**ptile**=8] \
    rusage[**mem**=3000]" mpirun ./mpi_program — *8 cores per node*
  - bsub –R "select[**gputype**==FireStream]" ./opencl_program
  - bsub –R "select[**gpumodel**==M2050] rusage[**gpu**=2]" ./cuda_program

# The black box…

- Is invoked automatically by LSF when a job is submitted
  - Cannot be bypassed — no cheating!
- Analyzes the job, its requirements and the user's environment
  - Serial or parallel — OpenMP, Quadrics MPI, Open MPI or MVAPICH2
  - Known application — Abaqus, Ansys, CFX, Comsol, Fluent, Gaussian, Materials Studio, Matlab, MSC Marc, OpenFOAM, R, StarCD, etc.
- Determines where and how the job should run
  - Binding rules for Quadrics MPI (QsNet$^{II}$) and Open MPI jobs (InfiniBand)
  - Heuristic rules — binding or not — for known applications
  - Default settings for unknown applications
- Modifies the job and/or its requirements as needed
  - If the job was not submitted correctly, explain it to the user and abort
- Passes the job on to LSF, which takes care of scheduling

# Queues

- Our goal is to keep the number of queues as small as possible
- We DO NOT define queues for specific…
  - Kinds of compute nodes
  - Users and groups
  - Applications
  - Job profiles
- We use queues only to enforce:
  - User limits (shareholders, guest users)
  - Time limits (1h, 8h, 36h, 7d)
- As a result, we have 2 categories of queues (vip, pub), each one with different time limits:
  - pub.{1h,8h,36h,7d} — open to everyone
  - vip.{36h,7d} — for shareholders only

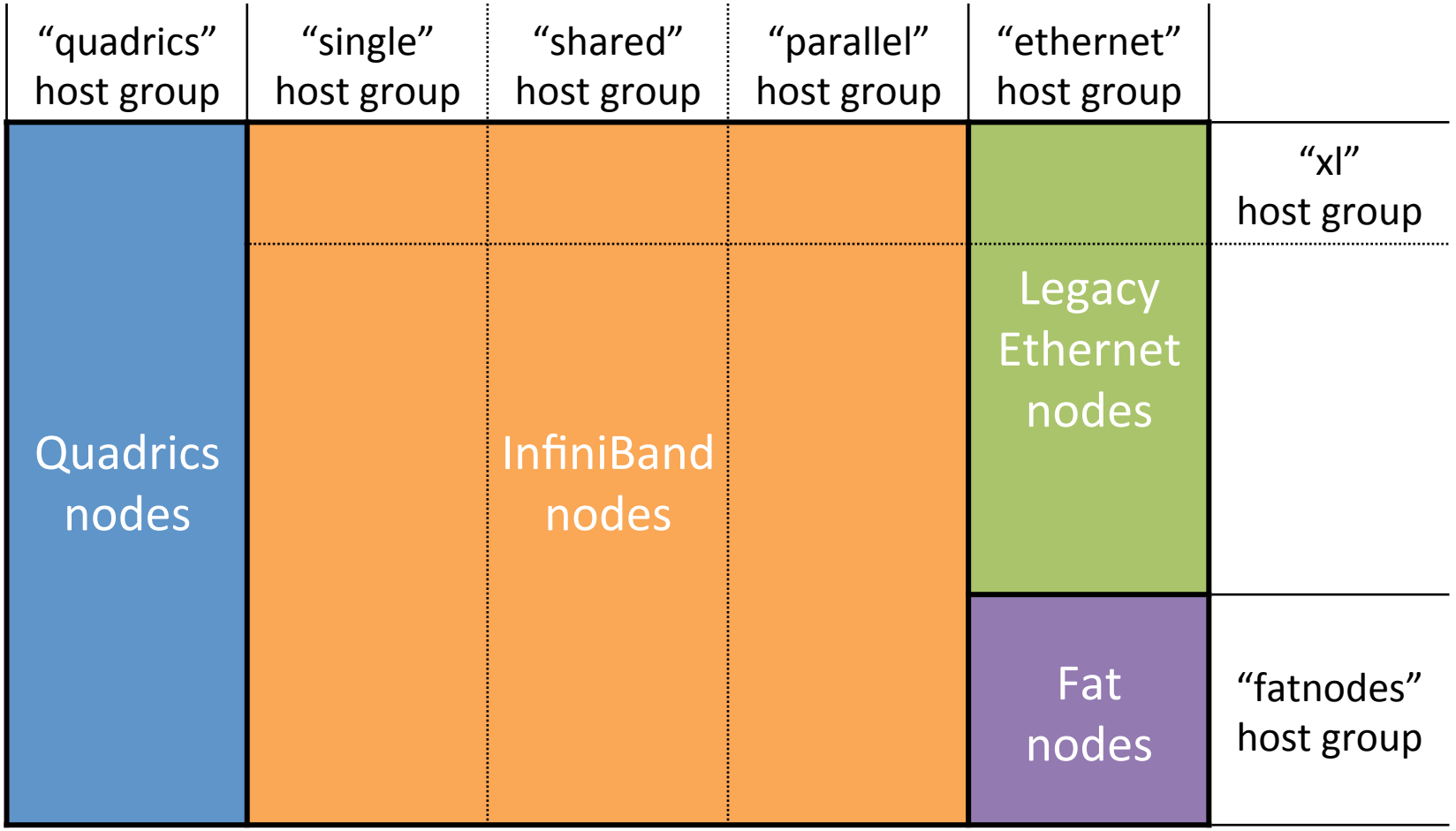
# Queues (cont.)

- We have a few other queues for special purposes:
  - qsnet.{s,m,l} — legacy queues under the control of Quadrics RMS
  - special — unlimited queue for exceptional jobs
  - fairshare — dummy queue for the definition of group and user priorities
  - purgatory — where jobs go in the event of a black box malfunction
- All queues can use all compute nodes in the cluster
  - Greatly simplifies the definition and management of the queues
  - Makes it easy to move a job from one queue to another if needed
- Exceptions
  - qsnet queues are specific to Quadrics nodes
  - 36h and 7d queues can use only a subset of nodes (for administrative reasons)

# Host groups

- We use host groups to control where a job should run
  - Groups can overlap: a node can belong to multiple groups
  - Groups can be organized hierarchically, e.g.: blade, chassis, rack, row
- Some host groups are based on hardware type
  - Quadrics nodes, fat nodes, GPU nodes, legacy nodes
- However, a host group does not need to be homogeneous
  - Resources can be used to select identical nodes within a group
- Some host groups are logical partitions within a set of nodes
  - Based on job type, e.g. single, shared, parallel, interactive
  - Boundaries between logical partitions can vary over time (day/night, weekday/weekend) or depending on the load of the cluster

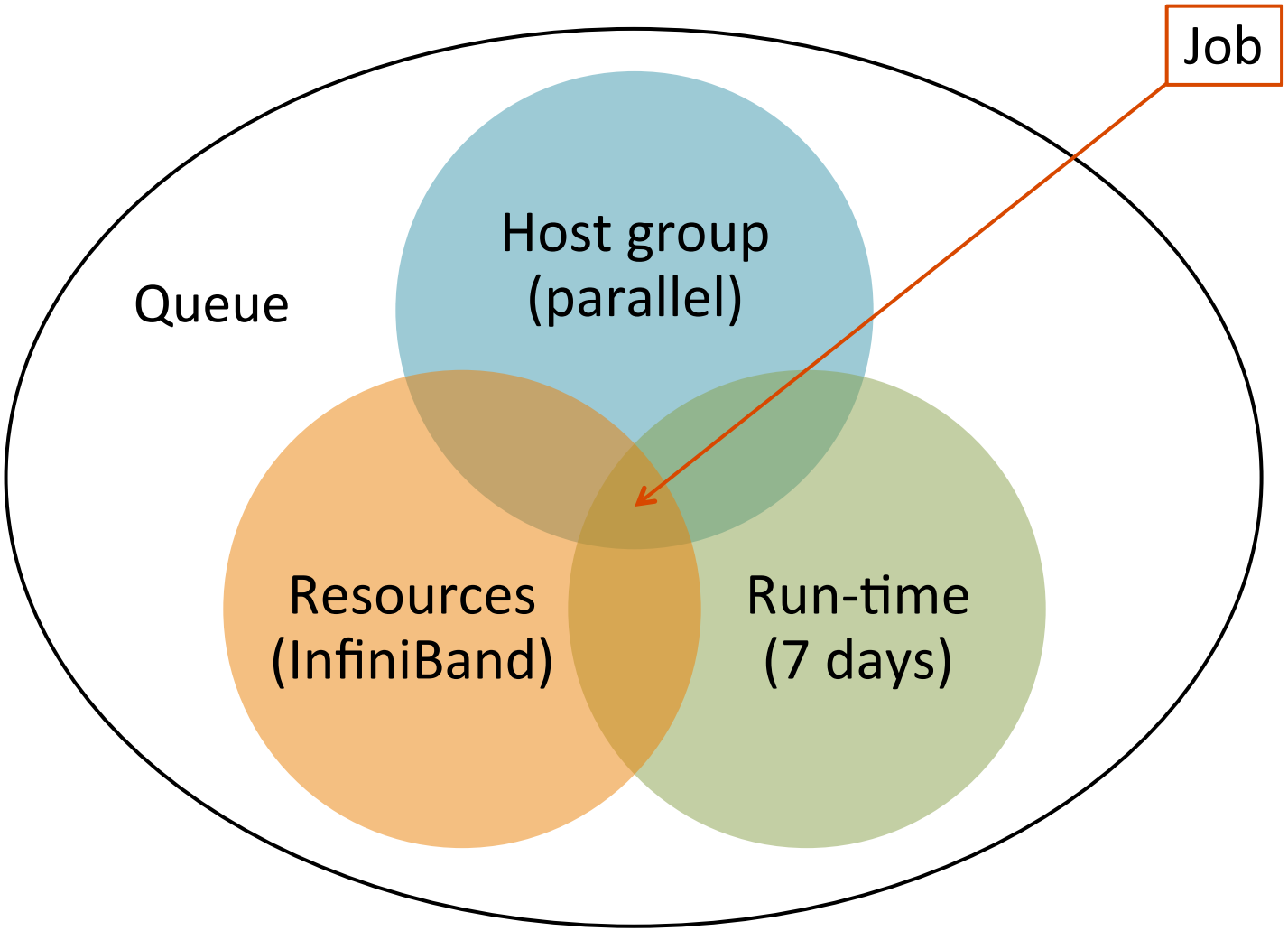
# Host groups (schematic view)



# Job allocation

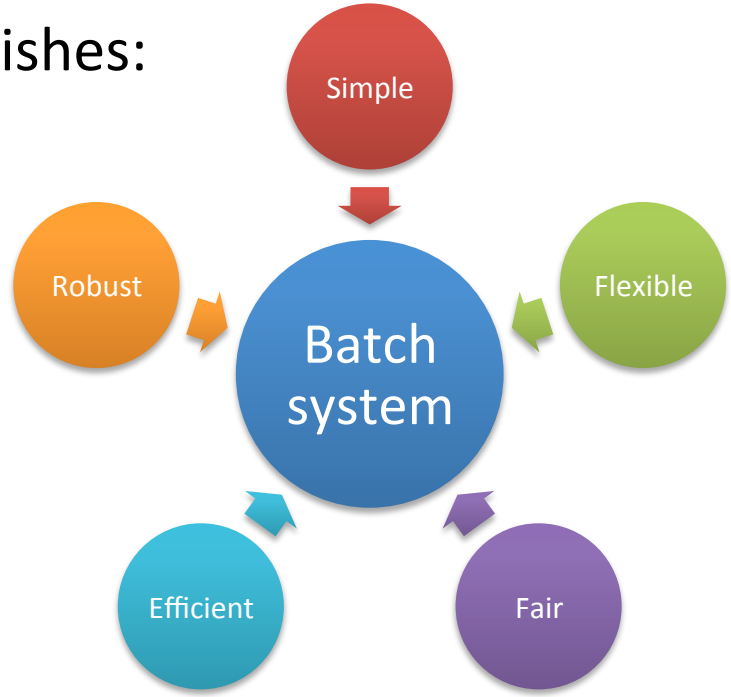
- The black box determines:
  - The queue category (qsnet or other) where the job must run
  - The host groups where the job may run, ranked by preference
- LSF selects the actual queue based on:
  - The resources (number of CPUs, run-time, etc.) requested by the job
  - Any additional requirements specified by the black box
  - The user's access rights
- The intersection of the queue, host groups and resources determines where the job will run
  - This intersection may be empty!

# Job allocation (schematic view)



# Summary

- The black box fulfills three of our wishes:
  - Simple
  - Flexible
  - Efficient
- LSF fulfills the remaining two:
  - Fair
  - Robust



# Final words

- The configuration of a batch system is a complex affair
  - There are many ways to do it
  - One must experiment — again and again! — to find the best one
- Experience gained on one system cannot always be transferred to another one
  - Different hardware and software
  - Different users, applications, workloads
- The batch system is just an implementation of usage policies
  - The definition of these policies is critical
- One must find the right balance between the needs of:
  - Individual users — long/large jobs, short wait time
  - The user community — high resource utilization, throughput