Chirag Dekate
Department of Computer Science
Louisiana State University
April 17, 2008

HIGH PERFORMANCE COMPUTING: MODELS, METHODS, & MEANS

SCHEDULING

Department of Computer Science
Louisiana State University

Topics

- Introduction
- CPU Scheduling
- Intro to Workload Management Systems
- Scheduling Algorithms
- Portable Batch System : PBS
- LoadLeveler
- Platform LSF
- Summary – Materials for Test

Topics

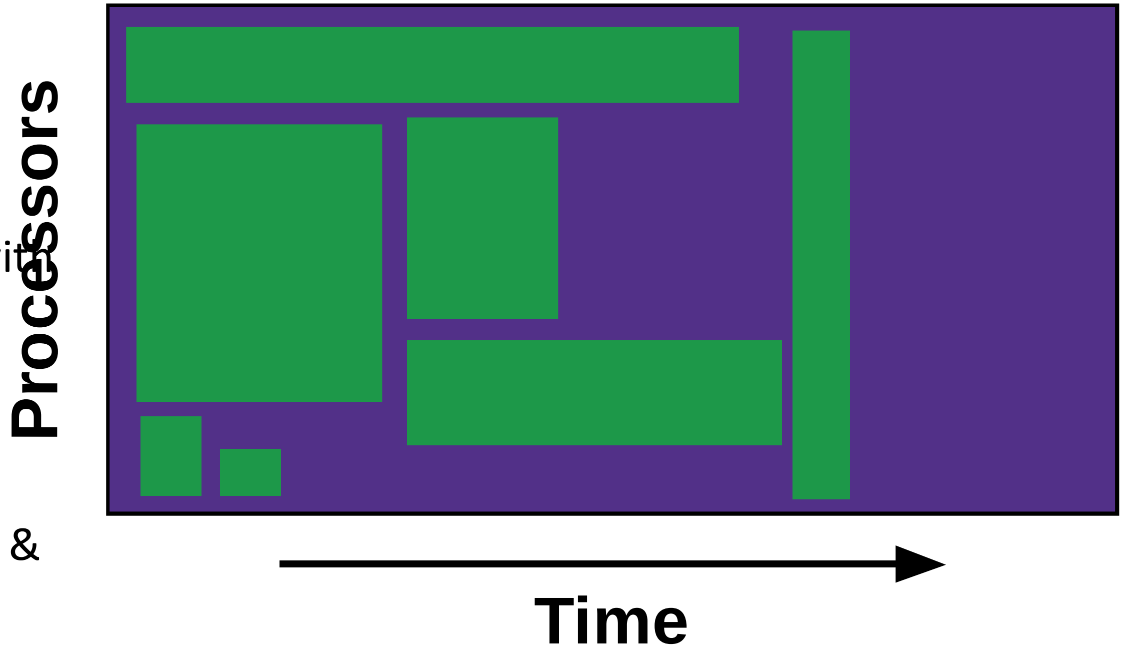
- Introduction
- CPU Scheduling
- Intro to Workload Management Systems
- Scheduling Algorithms
- Portable Batch System : PBS
- LoadLeveler
- Platform LSF
- Summary – Materials for Test

Introduction

- The term “*scheduling*” impacts several levels of the system, from application/task scheduling which deals with scheduling of threads to meta-scheduling where jobs are scheduled across multiple supercomputers.
- This lecture focuses on *job-scheduling*, scheduling of jobs on a supercomputer, the enabling software infrastructure and underlying algorithms.

Optimization problem in *packing* with space (processor) and time constraints

Dynamic environment
Tradeoffs in turnaround, utilization & fairness



Introduction : Job Scheduling

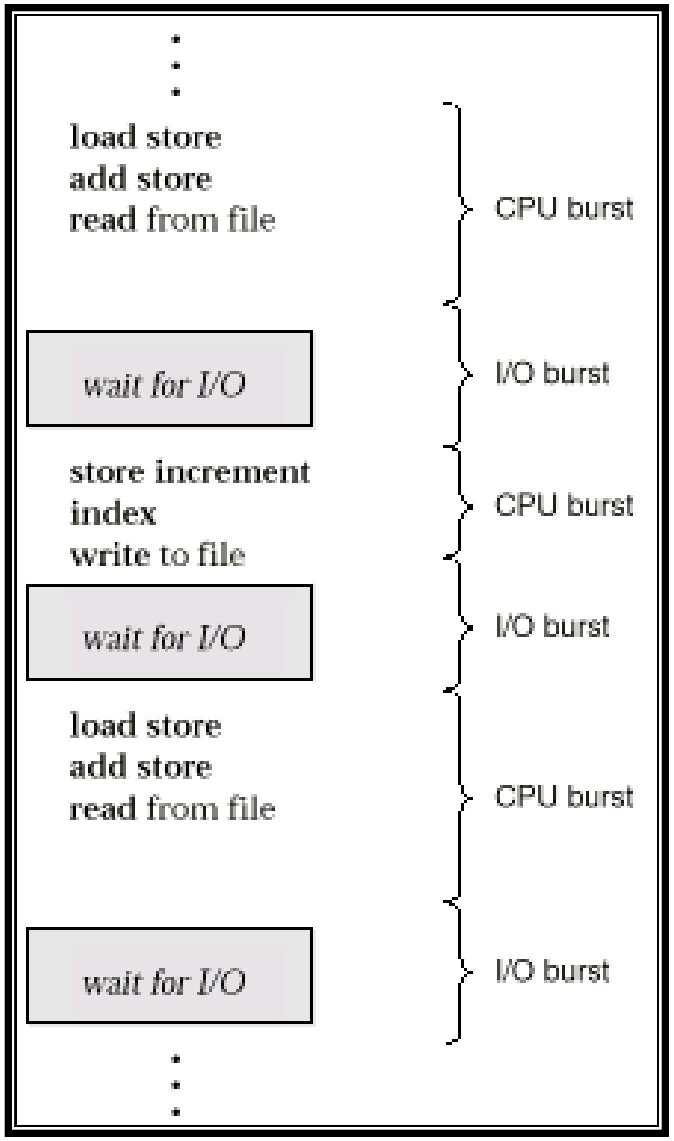
- Simplest way to schedule in a parallel system is using a queue
- Each job is submitted to a queue and each job upon appropriate resource match executes to completion while other jobs wait.
- Since each application utilizes only a subset of systems processors, the processors not in the subset could potentially remain idle during execution. Reduction of unused processors and maximization of system utilization is the focus of much of scheduling research.
- Space-Sharing
  - A natural extension of the queue where the system allows for another job in the queue to execute on the idle processors if enough are available. Eg. Batch Scheduling
- Time-Sharing
  - Another model where a processor's time is shared among threads of different parallel programs. In such approaches each processor executes a thread for some time, pauses it and then begins executing a new thread. Eg. Gang Scheduling
- Since context switching in Time-Sharing involves overhead,complex job resource requirements and memory management, most supercomputing installations prefer space-sharing scheduling systems
- **FOCUS : Batch Scheduling Systems**

Topics

- Introduction
- CPU Scheduling
- Intro to Workload Management Systems
- Scheduling Algorithms
- Portable Batch System : PBS
- LoadLeveler
- Platform LSF
- Summary – Materials for Test

CPU Scheduling

- Scheduling of CPUs is fundamental to Operating System design
- Process execution consists of a cycle of CPU execution and I/O wait. The process execution begins with a CPU burst followed by an I/O burst etc.
- OS selects one of the processes from the short-term scheduler / CPU scheduler.
- The scheduler selects from among the process in memory that are ready to execute and allocates the CPU to one of them.
- Scheduling happens under one of 4 conditions:
  - When process switches from running state to the waiting state **(Non - Preemptive)**
  - When a process switches from the running state to the ready state **(Preemptive)**
  - When a process switches from the waiting state to the ready state **(Preemptive)**
  - When a process terminates **(Non - Preemptive)**



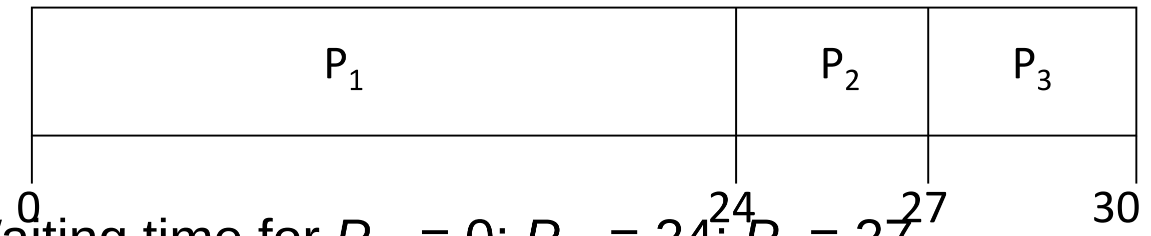
CPU Scheduling

- Scheduling Criteria:
  - **MAX** CPU Utilization – needs to keep the CPU as busy as possible.
  - **MAX** Throughput – Number of processes completed per time unit
  - **MIN** Turnaround Time – the interval from the time of submission of a process to the time of completion
  - **MIN** Waiting Time – Sum of the periods spent by the process waiting in the ready queue
  - **MIN** Response Time – The measure of time from the submission of a request until the first response is produced

First-Come, First-Served (FCFS) Scheduling

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule is:



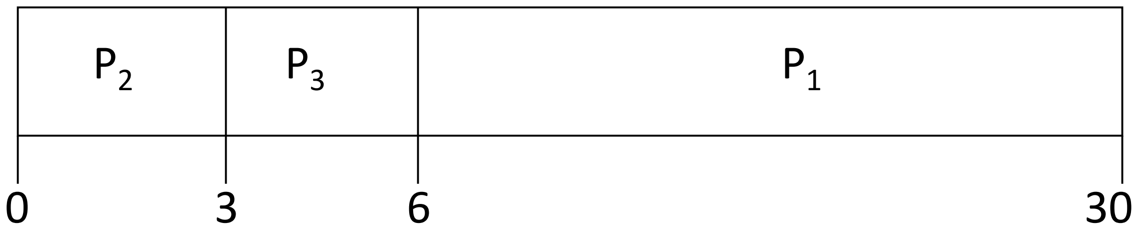
- Waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Average waiting time: $(0 + 24 + 27)/3 = 17$

FCFS Scheduling (Cont.)

Suppose that the processes arrive in the order

P_2 , P_3 , P_1 .

- The Gantt chart for the schedule is:



- Waiting time for $P_1 = 6; P_2 = 0; P_3 = 3$
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case.
- ***Convoy effect*** short process behind long process

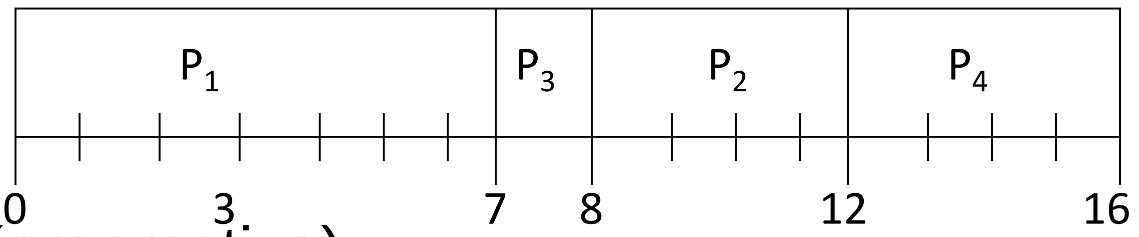
Shortest-Job-First (SJR) Scheduling

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time.
- Two schemes:
  - nonpreemptive – once CPU given to the process it cannot be preempted until completes its CPU burst.
  - preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is know as the Shortest-Remaining-Time-First (SRTF).
- SJF is optimal – gives minimum average waiting time for a given set of processes.

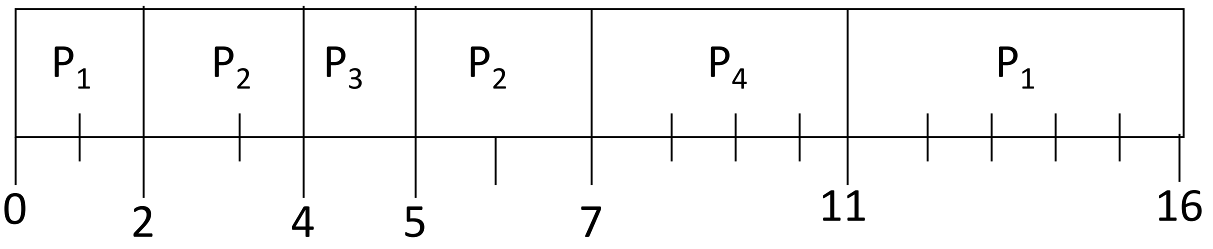
Example of Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)



- SJF (preemptive)



Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority).
  - Preemptive
  - nonpreemptive
- SJF is a priority scheduling where priority is the predicted next CPU burst time.
- Problem ≡ Starvation – low priority processes may never execute.
- Solution ≡ Aging – as time progresses increase the priority of the process.

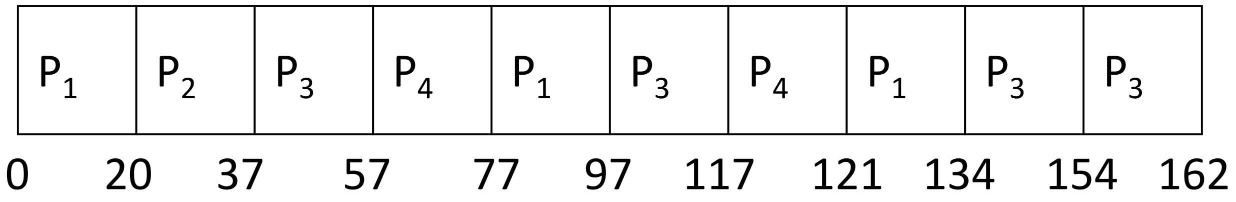
Round Robin (RR)

- Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the CPU time in chunks of at most *q* time units at once. No process waits more than (*n*-1)*q* time units.
- Performance
  - *q* large $\Rightarrow$ FIFO
  - *q* small $\Rightarrow$ *q* must be large with respect to context switch, otherwise overhead is too high.

Example of RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

- The Gantt chart is:

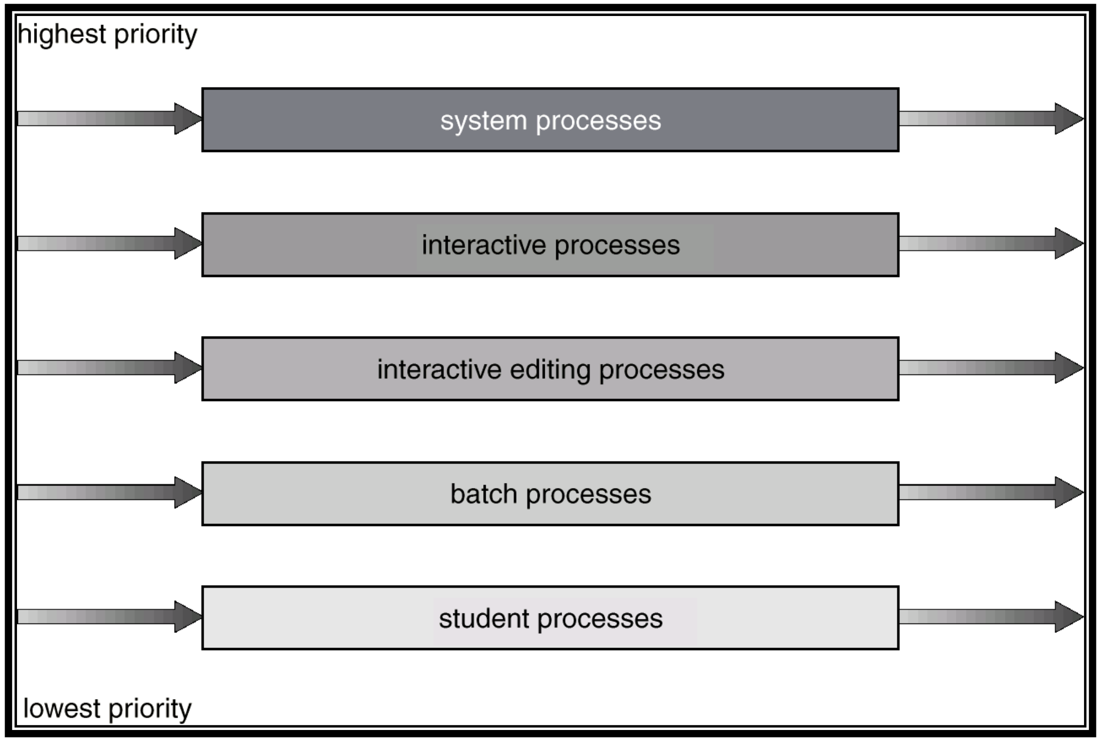


- Typically, higher average turnaround than SJF, but better *response*.

Multilevel Queue

- Ready queue is partitioned into separate queues:
  foreground (interactive)
  background (batch)
- Each queue has its own scheduling algorithm,
  foreground – RR
  background – FCFS
- Scheduling must be done between the queues.
  – Fixed priority scheduling; (i.e., serve all from foreground then from background). Possibility of starvation.
  – Time slice – each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e.,
    - 80% to foreground in RR
    - 20% to background in FCFS

Multilevel Queue Scheduling



Topics

- Introduction
- CPU Scheduling
- Intro to Workload Management Systems
- Scheduling Algorithms
- Portable Batch System : PBS
- LoadLeveler
- Platform LSF
- Summary – Materials for Test

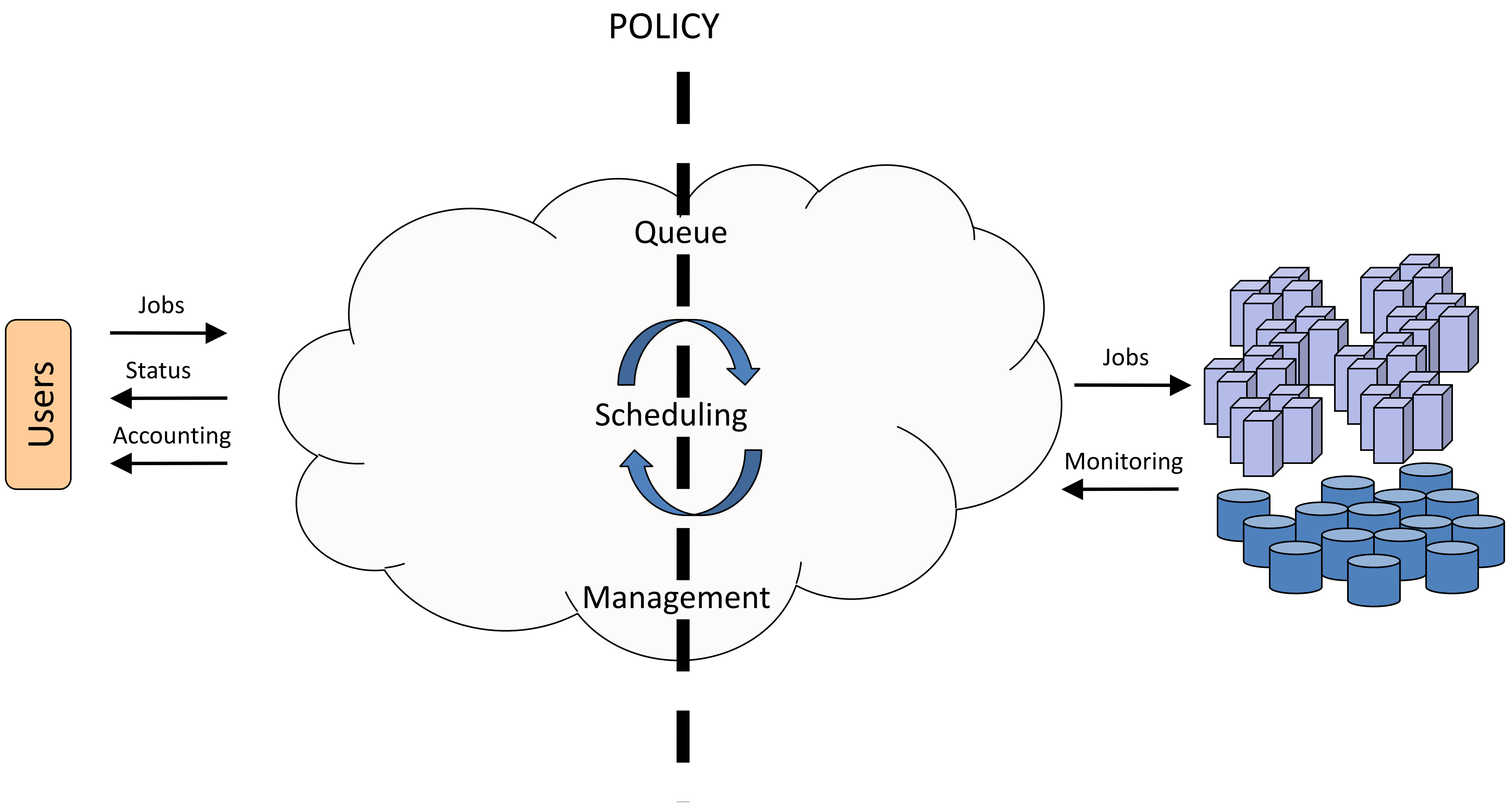
Introduction

- Supercomputing Centers often support several hundreds of users running thousands of jobs across a shared supercomputing resources consisting of large number of compute nodes and storage centers.
- It can be extremely difficult for an administrator of such a resource to manually manage users, resource allocations and ensure optimal utilization of the large supercomputing resources.
- Workload management systems (WMS) help address this problem by providing users with simple resource access and for the administrator of the supercomputing resource, a set of powerful tools and utilities for resource management, monitoring, scheduling, accounting, and policy enforcement.

Workload Management Systems

- WMS - a layer between the users and the end compute resources.
  - Users submit jobs, specifying the work to be performed, to a queue
  - Jobs wait in queue until they are scheduled to start on the cluster.
  - Scheduling is governed by stipulated policies and algorithms that implement the policy. (Policies usually ensure fair sharing of resources and attempt to optimize overall system utilization.)
  - Resource management mechanisms handle launching of jobs and subsequent cleanup.
  - The workload management system is simultaneously monitoring the status of various system resources and accounting resource utilization.

Activities of a WMS



WMS Main Activities

- Main activities performed by workload management systems include :
  - Queuing
  - Scheduling
  - Monitoring
  - Resource Management
  - Accounting

WMS : Queueing

- The most visible part of the WMS process where the system collects the jobs to be executed.
- Submission of jobs is usually performed in a container called a “batch job” (usually specified in the form of a file)
- The batch job consists of two primary parts :
  - A set of resource directives (number of CPUs, amount of memory etc.)
  - A description of the task(s) to be executed (executable, arguments etc.)
- Upon submission the batch job is held in a queue until a matching resource is found. Queue wait time for submitted jobs could vary depending on the demand for the resource among users.
- Production or real-world supercomputing resources often have multiple queues, each of which can be preconfigured to run certain kinds of jobs. ExampleTezpurcluster has a debug queue and workq
- *Discover what queues and associated restrictions at your local sites.*

WMS : Scheduling

- Scheduling selects the best job to run based on the current resource availability and scheduling policy.
- Scheduling can be generally broken into two primary activities :
  - **Policy enforcement** : To enforce resource utilization based on policies set by supercomputing sites (controls job priority and schedulability).
  - **Resource Optimization** : Packs jobs efficiently, and exploit underused resources to boost overall resource utilization.
- Balancing policy enforcement with resource optimization in order to pick the best job to run is the difficult part of scheduling
- Common scheduling algorithms include First In First Out, Backfill, Fairshare.

WMS : Monitoring

- Resource monitoring by WMS, provides administrators, users and scheduling systems with status information of jobs and resources. Monitoring is often performed for 3 critical states :
- For *idlenodes*, to verify their working order and readiness to run another job.
- For *busynodes*, to monitor memory, CPU, network, I/O and utilization of system resources to ensure proper distribution of workload and effective utilization of nodes.
- For *completed jobs*, to ensure that no processes remain from the completed job and that the node is still in working order before a new job is started on it.

WMS : Resource Management

- Resource Management area is responsible for starting, stopping and cleaning up after jobs.
- A batch system resource management is setup in such a way so as to run the jobs using the identity of a user in such a way that the user need not be present at that time.
- Jobs are started only on the nodes that are functioning properly.
- Resource management also includes removing or adding of resources to the available pool of systems
- Clusters are dynamic resources, systems go down, or additional resources are added.
- “Registration” of new nodes and the marking of nodes as unavailable are additional aspects of resource management

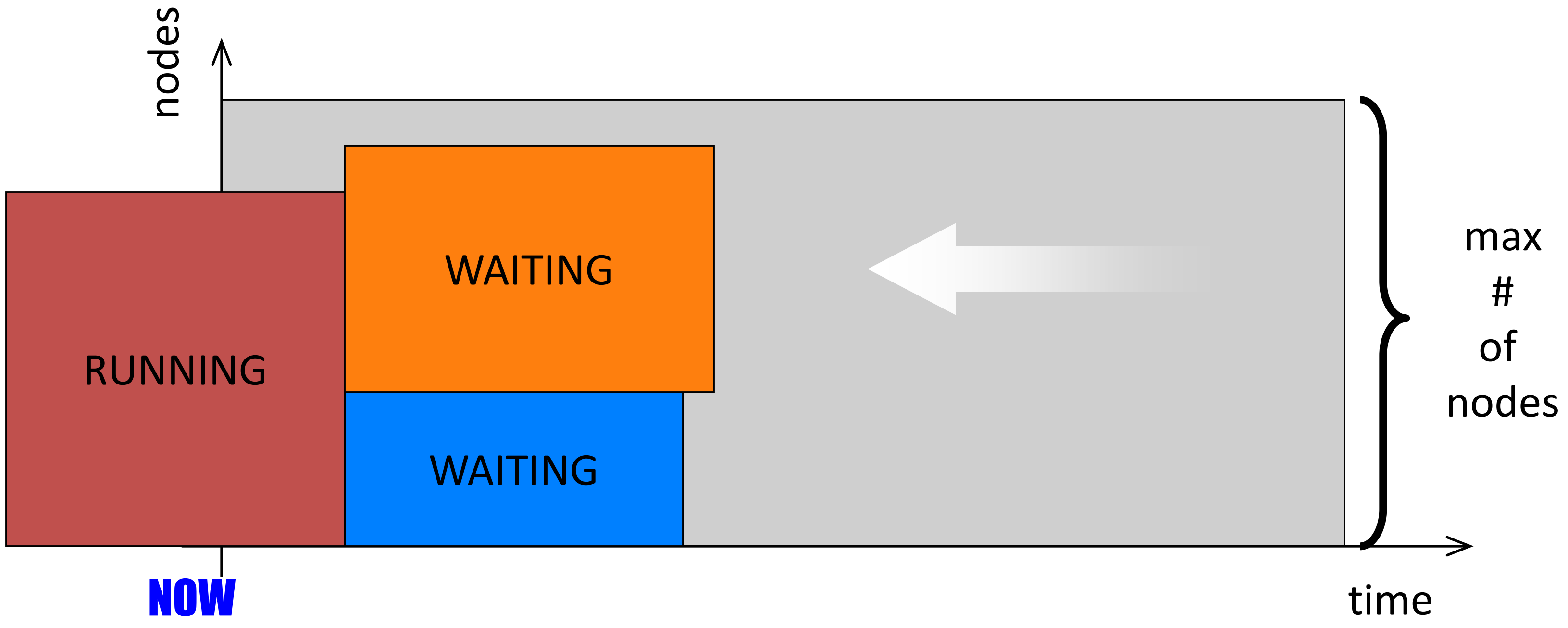
WMS : Accounting & Reporting

- Workload accounting can be defined as the process of collecting resource usage data for the batch jobs that run on the resource. (example % CPU utilization, memory utilization etc.)
- Data obtained from accounting is often used to :
  – Produce weekly/monthly per user usage reports
  – Tuning of scheduling policy
  – Calculating future resource allocations
  – Anticipating future computer component requirements
  – Determining areas for improvement within the system.

Topics

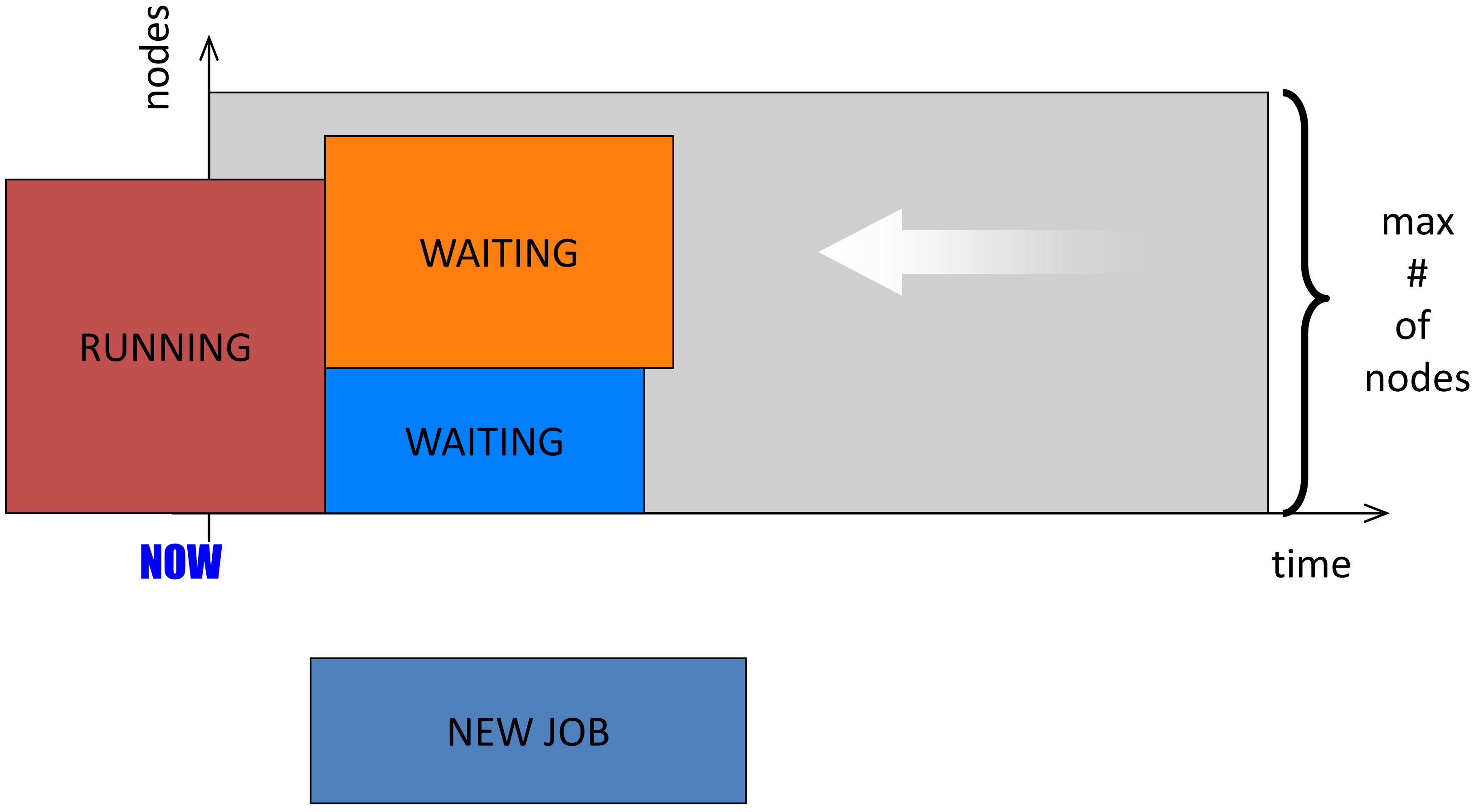
- Introduction
- CPU Scheduling
- Intro to Workload Management Systems
- Scheduling Algorithms
- Portable Batch System : PBS
- LoadLeveler
- Platform LSF
- Summary – Materials for Test

FCFS / FIFO



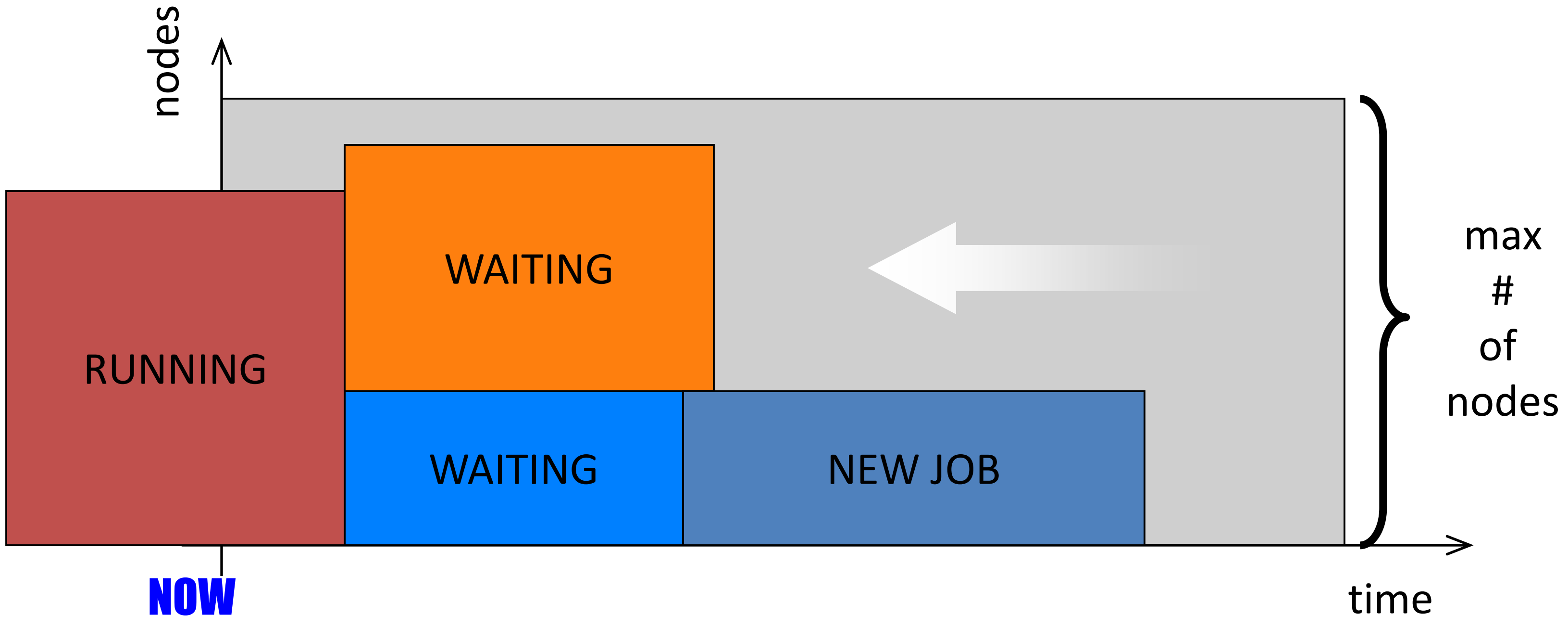
src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

FCFS / FIFO



src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

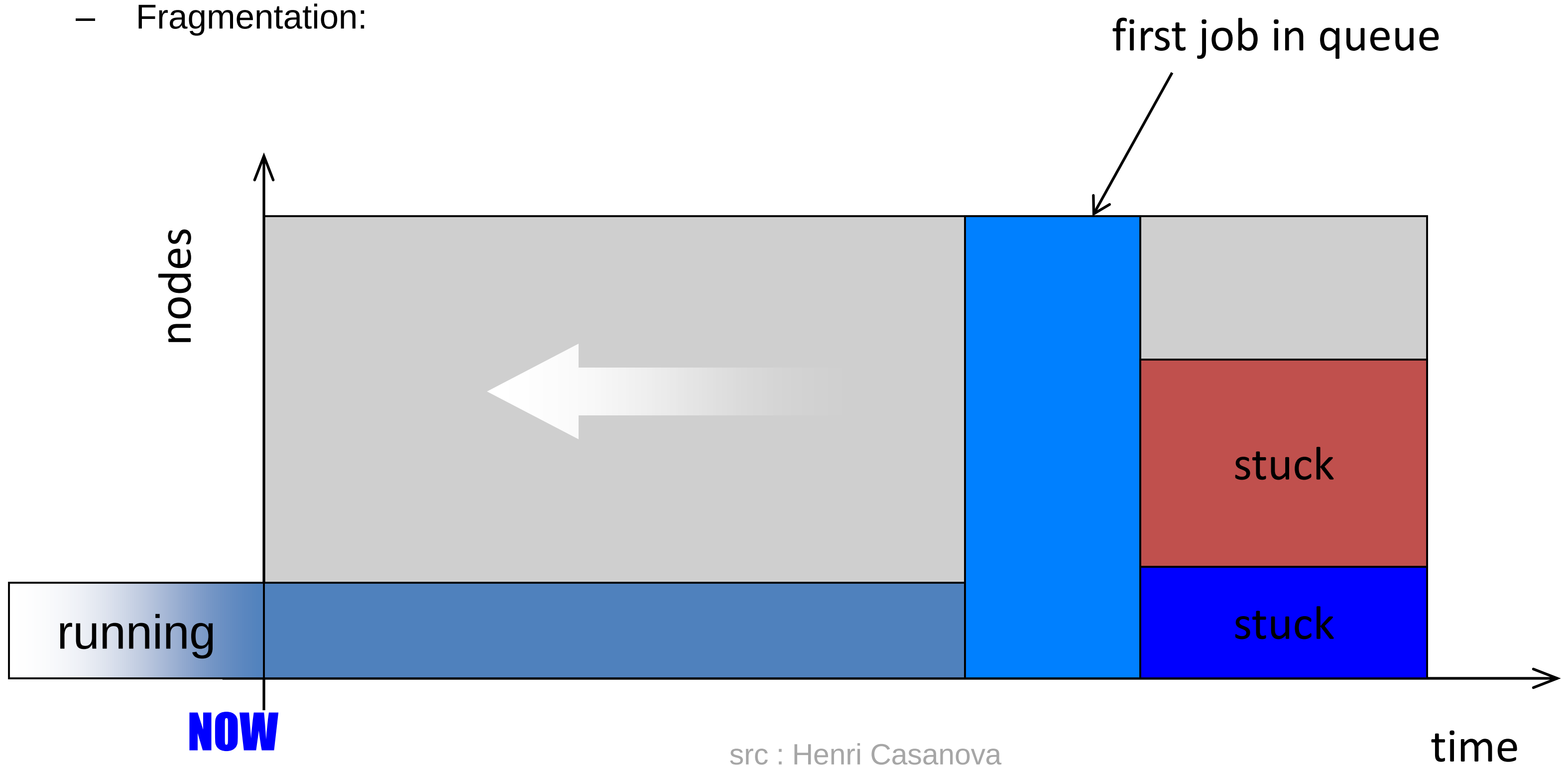
FCFS / FIFO



src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

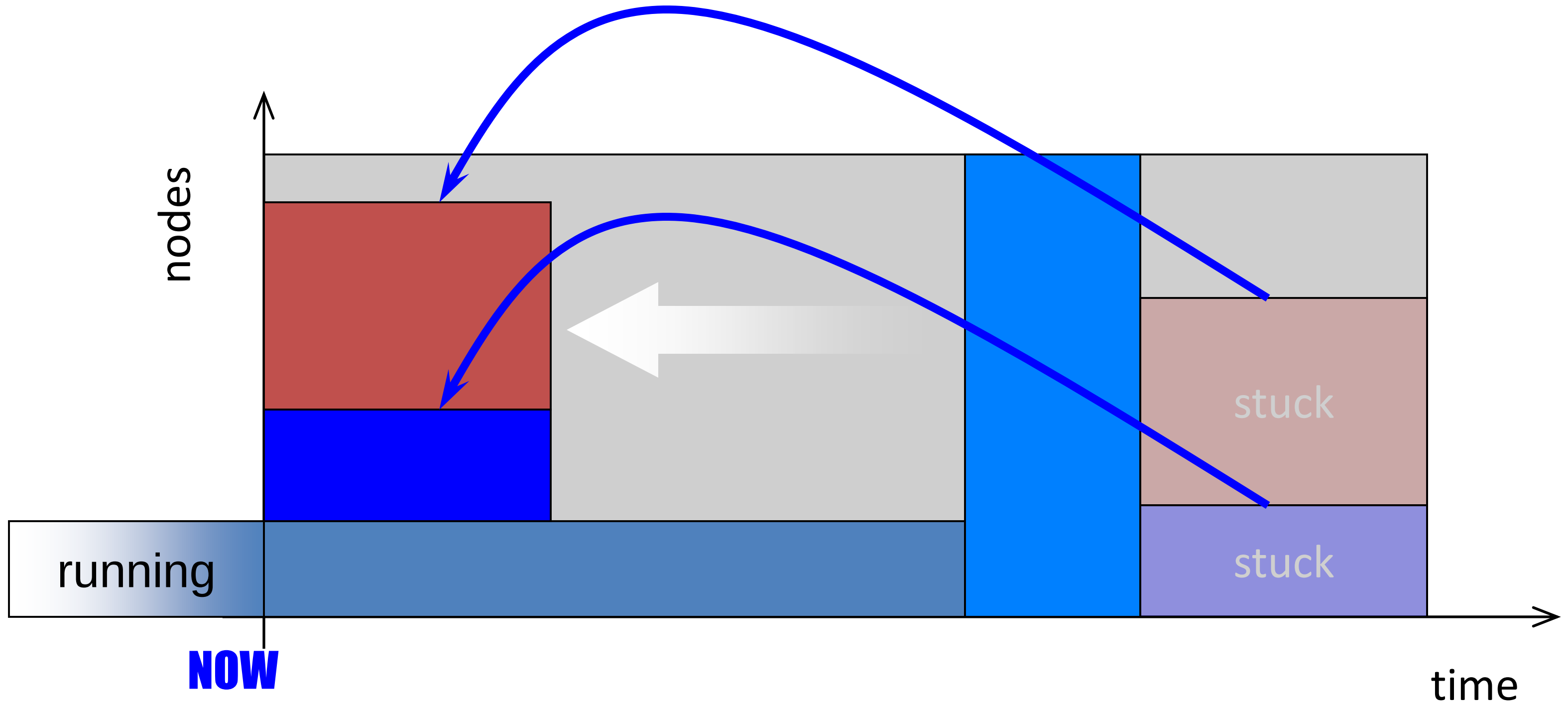
Scheduling FCFS

- Definitions
  - Shadow time: time at which the first job in the queue starts execution
  - Extra nodes: number of nodes idle when the first job in the queue starts execution
- Simplest scheduling option: FCFS
  - First Come First Serve
- Problem:
  - Fragmentation:



src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

FIFO With Backfilling



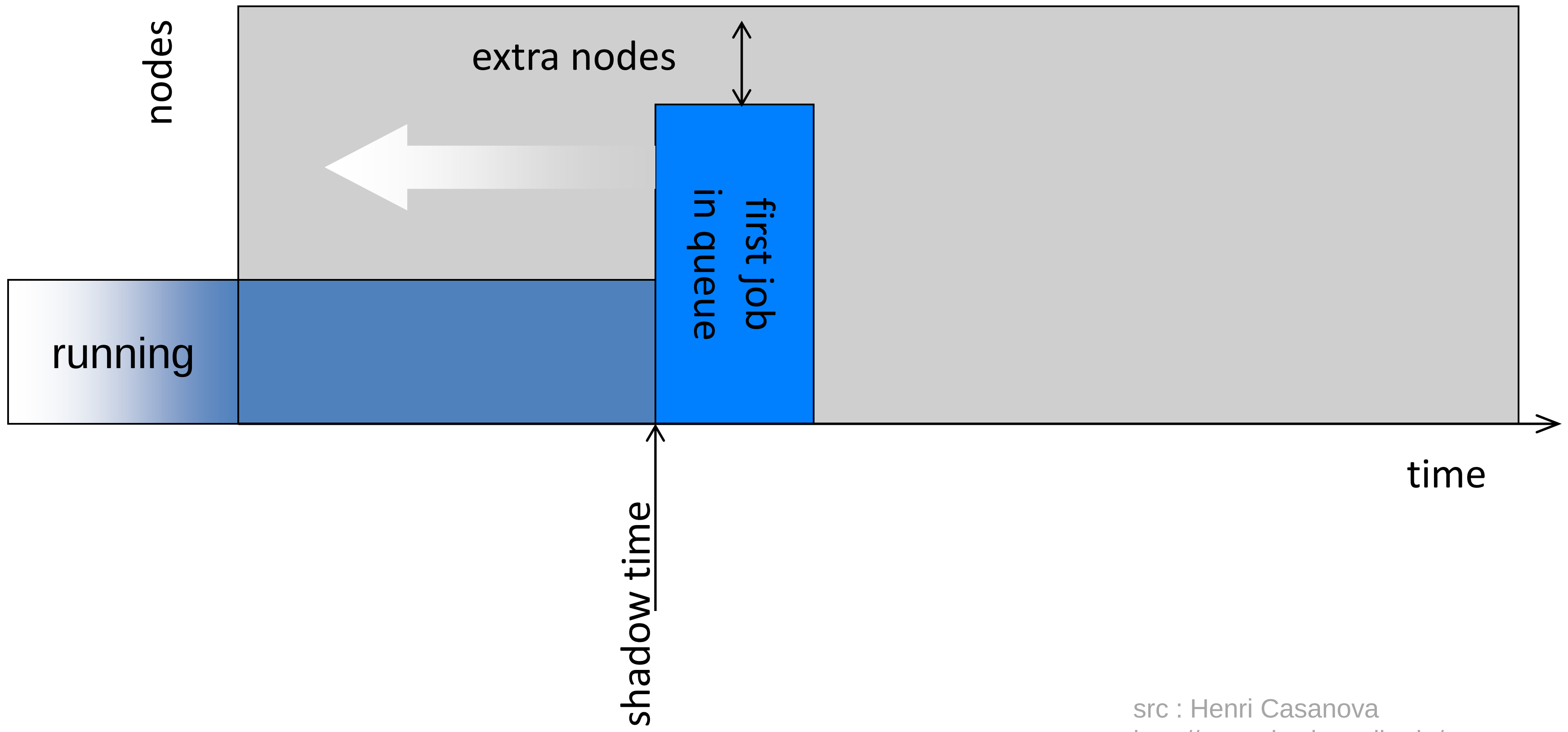
src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

Backfilling & EASY heuristic

- Which job(s) should be picked for promotion through the queue?
- Many heuristics are possible
- Two have been studied in detail
  - EASY
  - Conservative Back Filling (CBF)
- In practice EASY (or variants of it) is used, while CBF is not
- Extensible Argonne Scheduling System
- Maintain only one “reservation”, for the first job in the queue
- Go through the queue in order starting with the 2nd job
- Backfill a job if
  - it will terminate by the shadow time, or
  - it needs less than the extra nodes

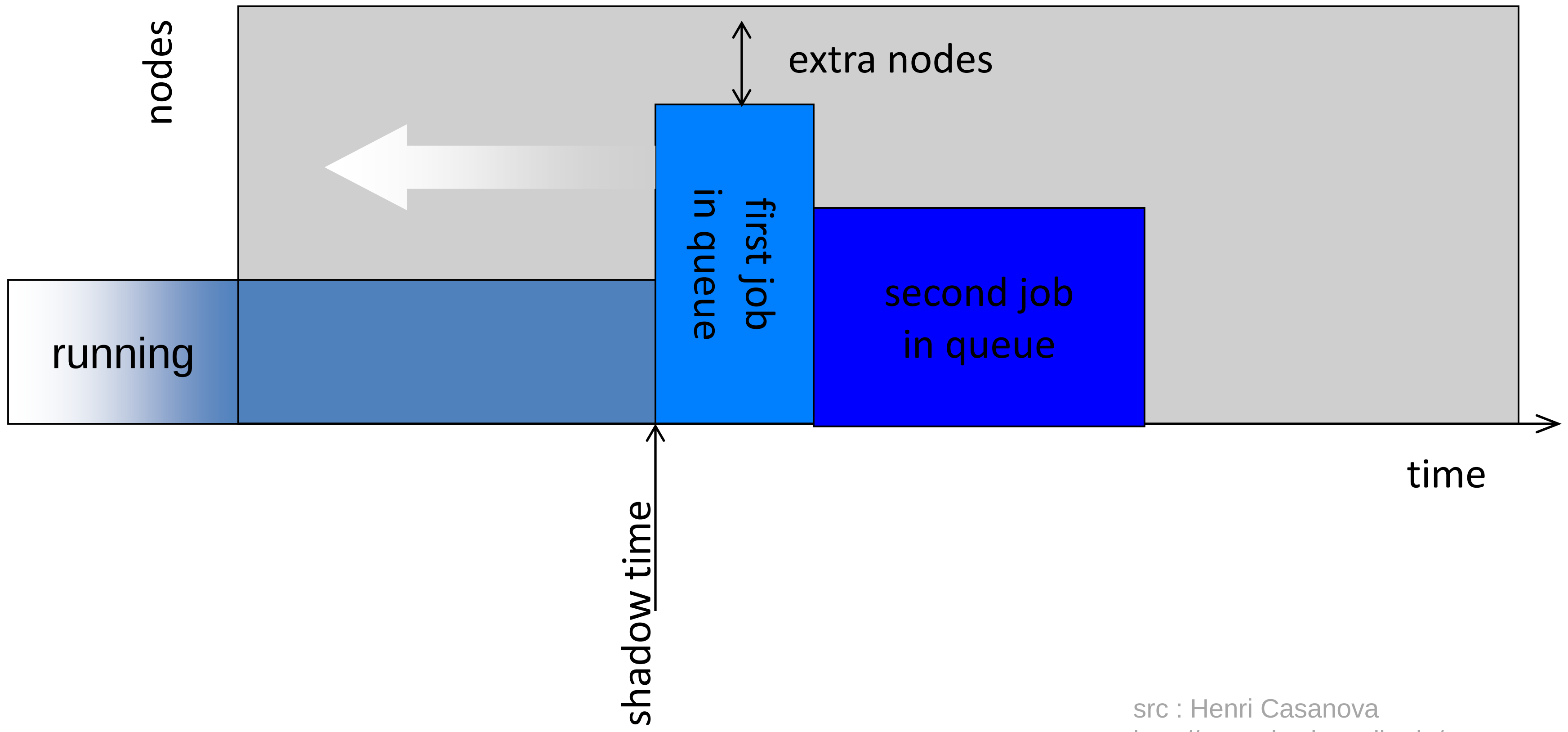
src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

EASY: Example



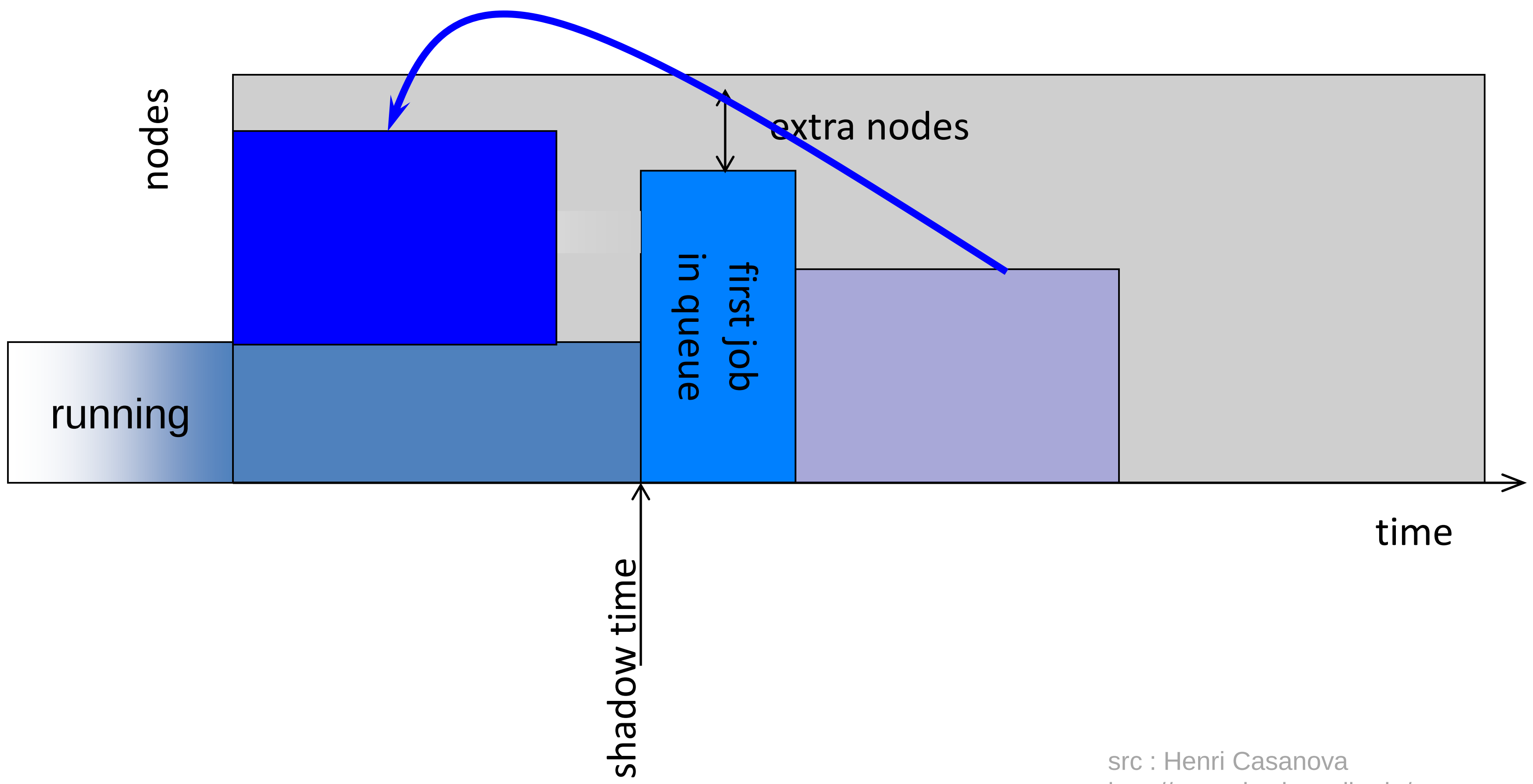
src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

EASY: Example



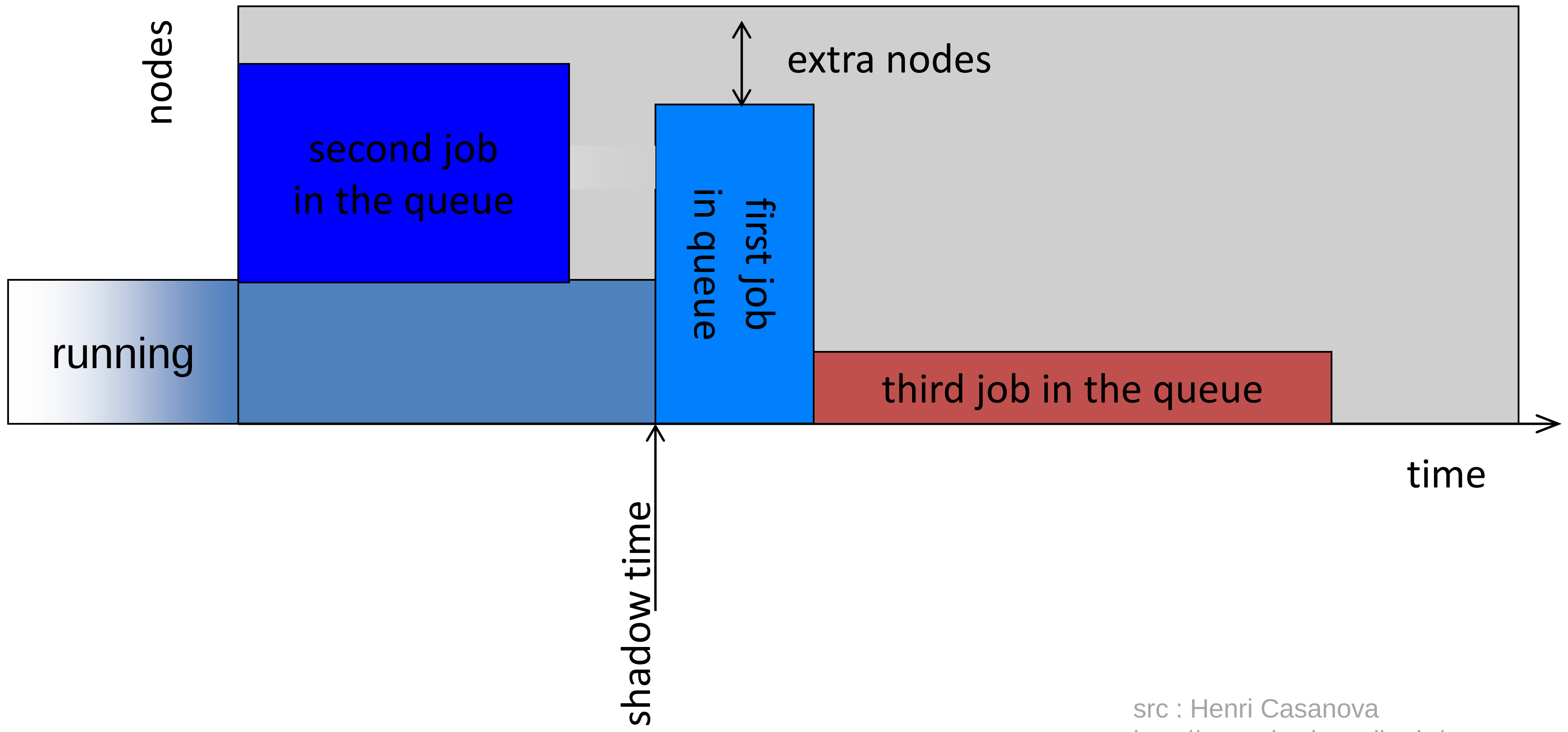
src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

EASY: Example



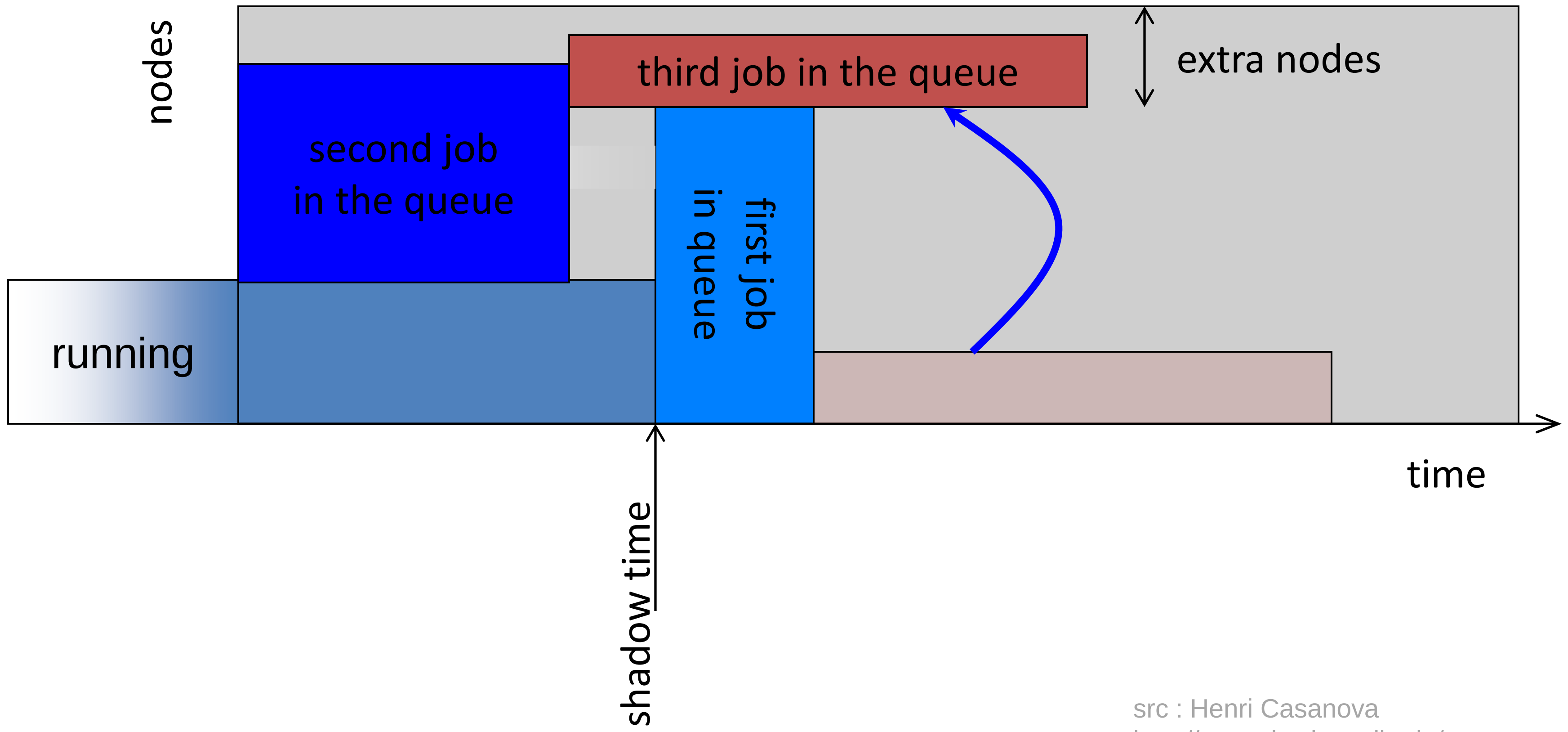
src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

EASY: Example



src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

EASY: Example



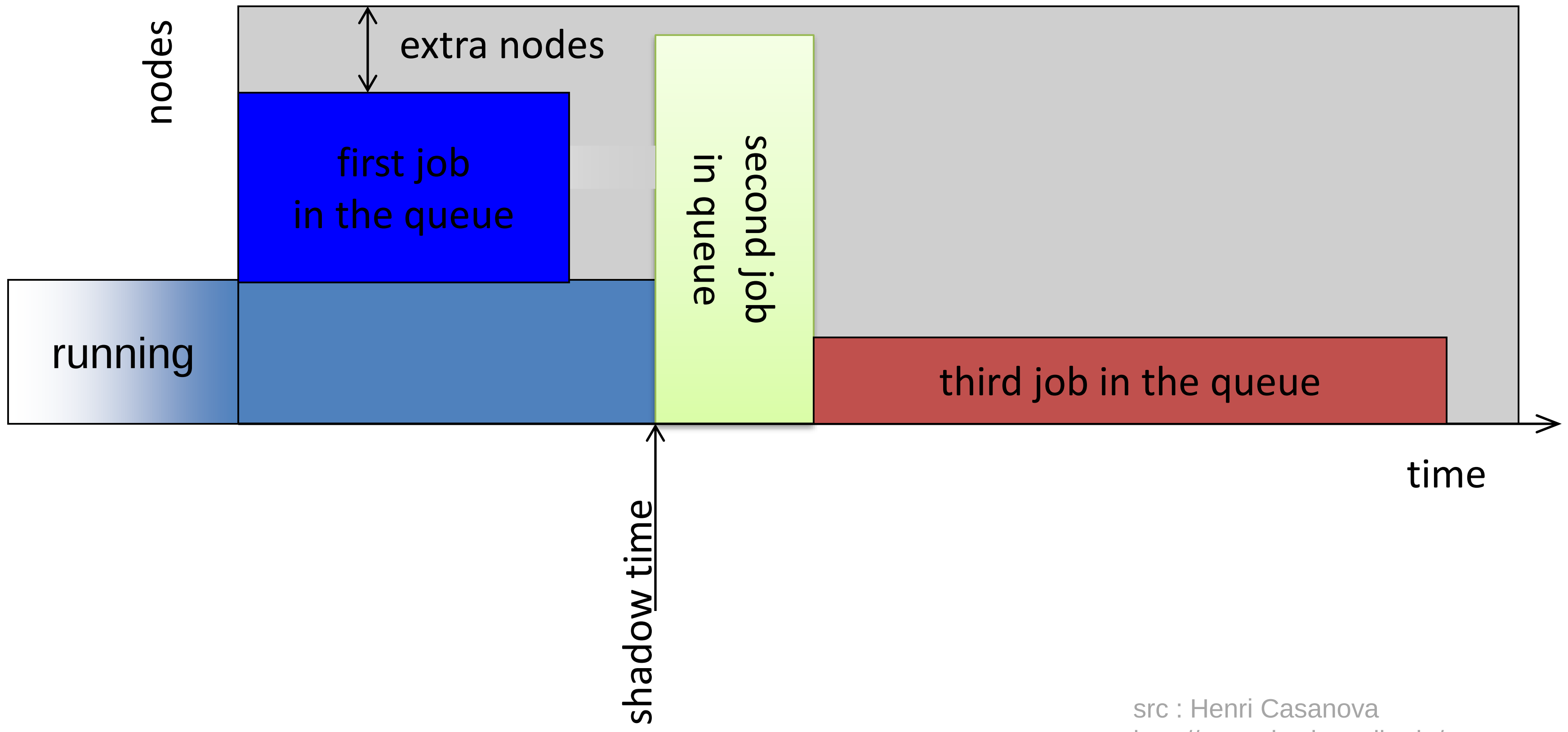
src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

EASY: Properties

- Unbounded Delay
  - The first job in the queue will never be delayed by backfilled jobs
  - BUT, other jobs may be delayed infinitely!

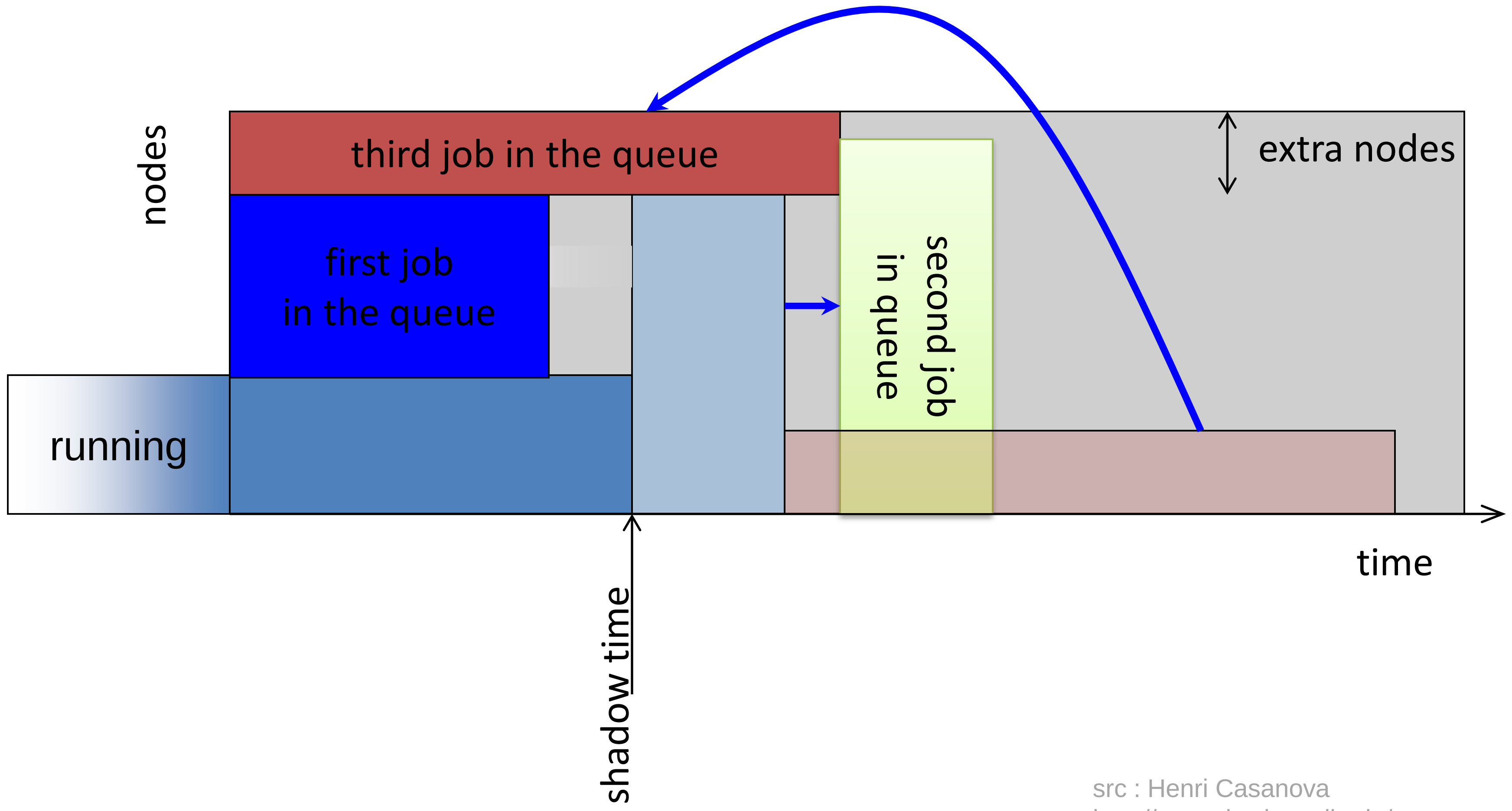
src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

EASY: Unbounded Delay



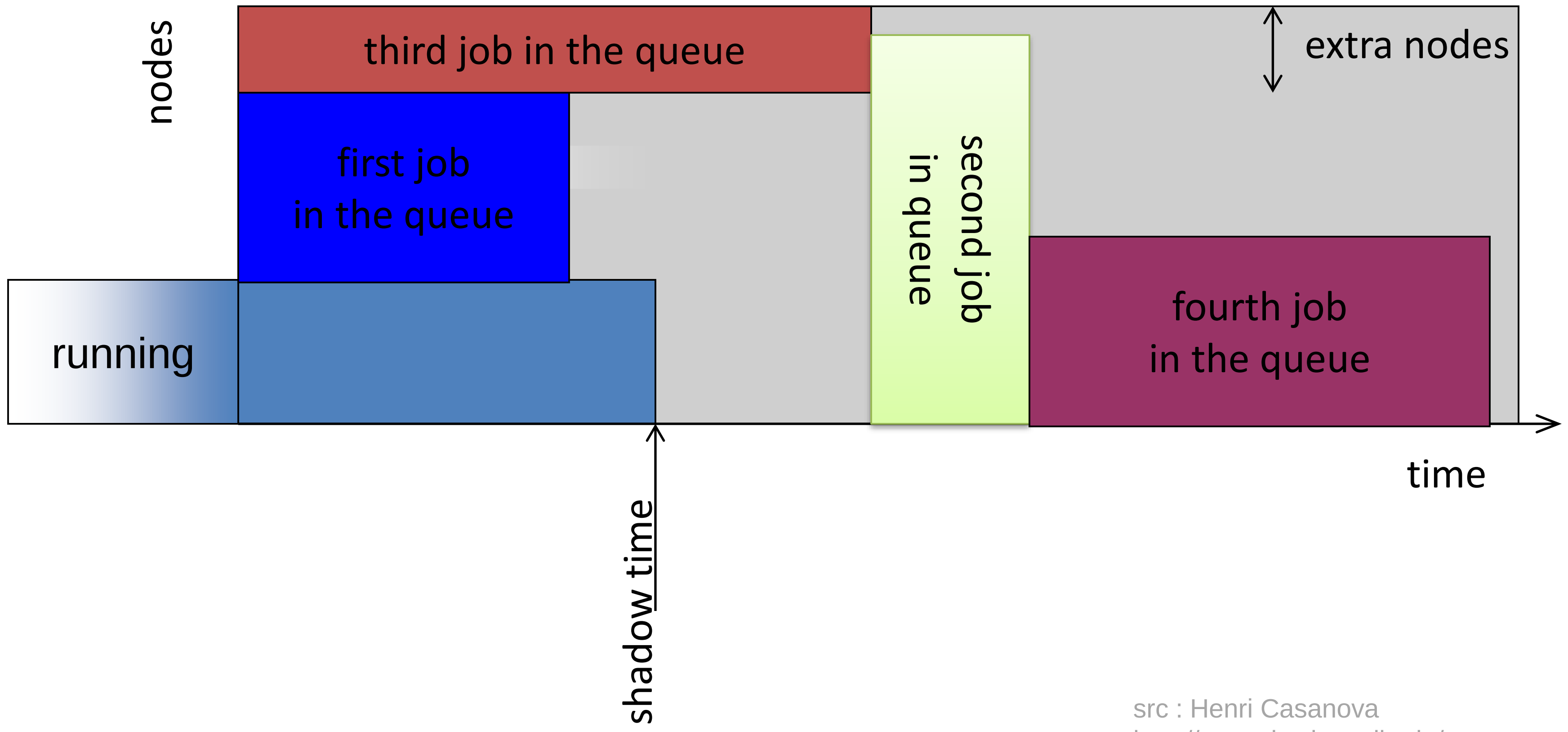
src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

EASY: Unbounded Delay



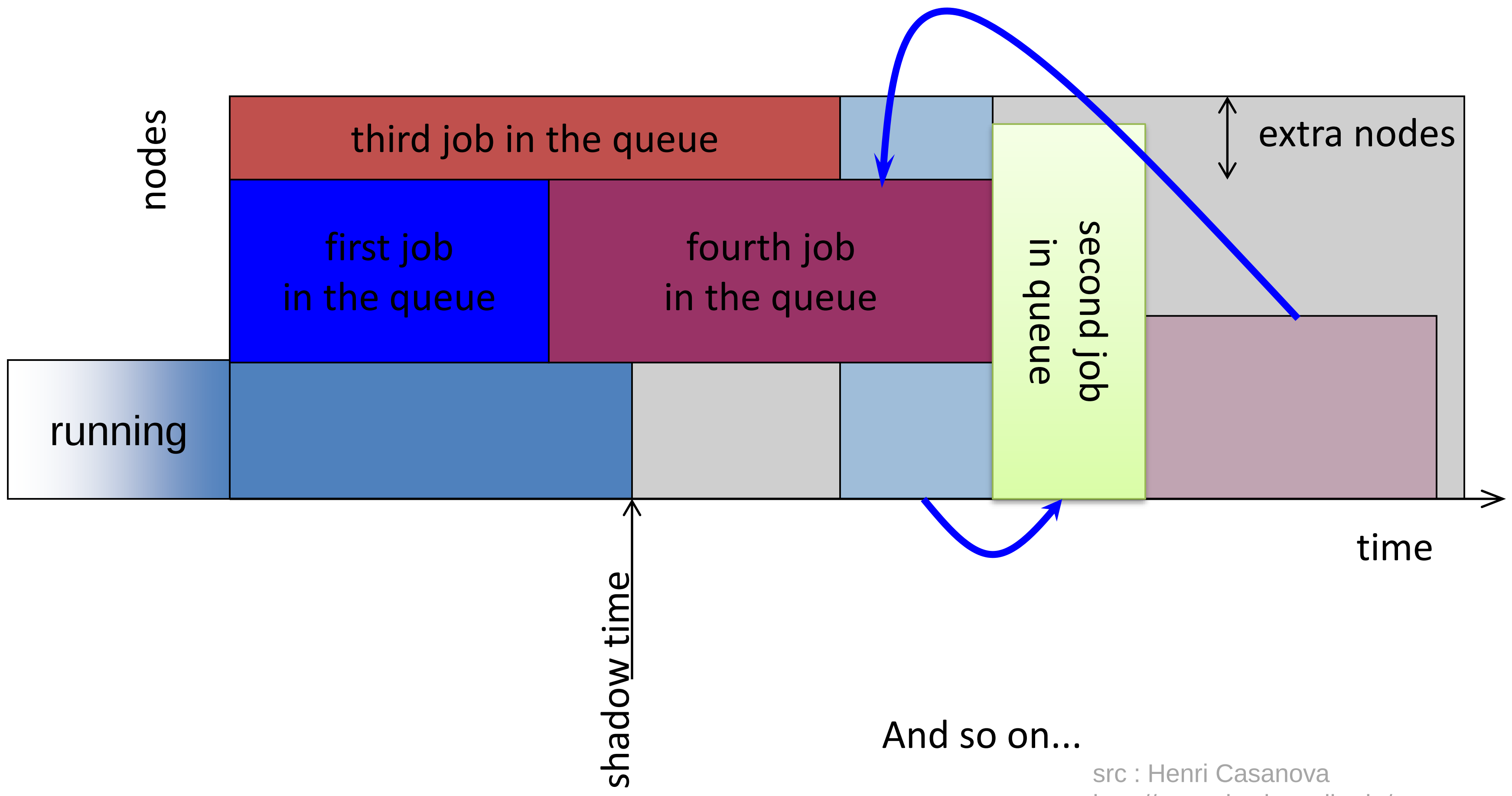
src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

EASY: Unbounded Delay



src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

EASY: Unbounded Delay



And so on...

src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

EASY: Properties

- Unbounded Delay
  - The first job in the queue will never be delayed by backfilled jobs
  - BUT, other jobs may be delayed infinitely!
- No starvation
  - Delay of first job is bounded by runtime of current jobs
  - When the first job runs, the second job becomes the first job in the queue
  - Once it is the first job, it cannot be delayed further
- EASY favors small long jobs
- EASY harms large short jobs

src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

Conservative Backfilling

- EVERY job has a “reservation”
- A job may be backfilled only if it does not delay any other job ahead of it in the queue
- Fixes the unbounded delay problem that EASY has
- More complicated to implement
  - The algorithm must find holes in the schedule

src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

How good is the schedule?

- All of this is great, but how do we know what a “good” schedule is?
  - FCFS, EASY, CFB, Random?
- What we need are metrics to quantify how good a schedule is
  - It has to be an aggregate metric over all jobs
- Metric #1: Turn-around time
  - Also called flow
  - Wait time + Run time
- But:
  - Job #1 needs 1h of compute time and waits 1s
  - Job #2 needs 1s of compute time and waits 1h
- Clearly Job #1 is really happy, and Job #2 is not happy at all

src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

How good is the schedule?

- One question is: How do we come up with a metric that captures the level of “user happiness”?
- Wait time is annoying, so...
- Metric #2: wait time
- But
  - – Job #1 asks for 1 node and waits 1 h
  - – Job #2 asks for 512 nodes and waits 1h
- Again, Job #1 is unhappy while Job #2 is probably sort of happy

src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

How good is the schedule?

- What we really want is a metric that represents happiness for small, large, short, long jobs
- The best we have so far: Slowdown
  - Also called stretch
- Metric #3: “turn-around time” divided by “turn-around time if alone in the system”
- This takes care of the short/long problem
- Doesn’t really take care of the small/large problem
  - Could think of some scaling, but unclear

src : Henri Casanova
http://navet.ics.hawaii.edu/~casanova/

Topics

- Introduction
- CPU Scheduling
- Intro to Workload Management Systems
- Scheduling Algorithms
- Portable Batch System : PBS
- LoadLeveler
- Platform LSF
- Summary – Materials for Test

History : Batch Submission Systems

- Computers used to be completely interactive, Background jobs were processes with input disconnected from the terminal.
- Increase in networked compute servers and workstations led to the requirement of a networked batch scheduling capability.
- Network Queuing System (NQS) – NASA Ames Research System in 1986.
- Distributed parallel systems began to emerge, NQS unable to handle jobs across such resources.
- NASA funded development of a new resource management system : Portable Batch System based on the IEEE POSIX standard.

PBS : Portable Batch System

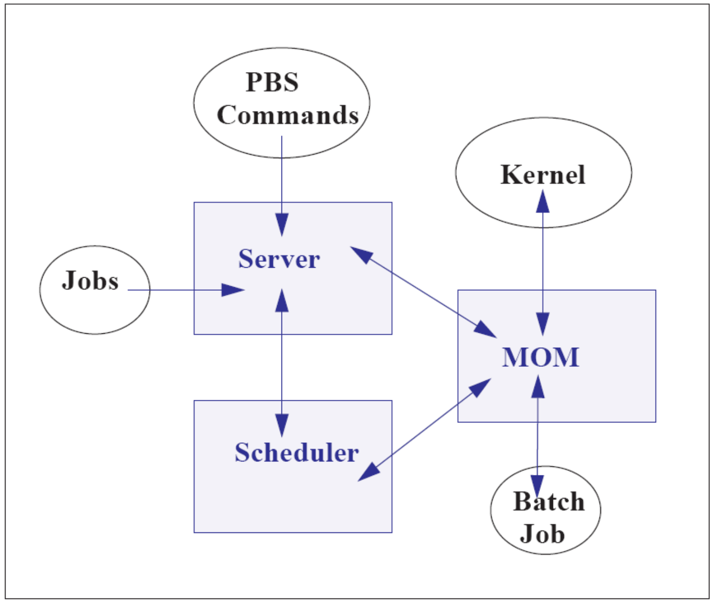
- PBS is a flexible distributed workload management and job scheduling system originally developed to manage aerospace computing resources.
- PBS is one of the common workload management system deployed across
- www.pbspro.org
- www.openpbs.org

PBS : Architecture Overview … 1

- PBS has two major component types:
  - Command-line utilities
  - System processes.

Server :

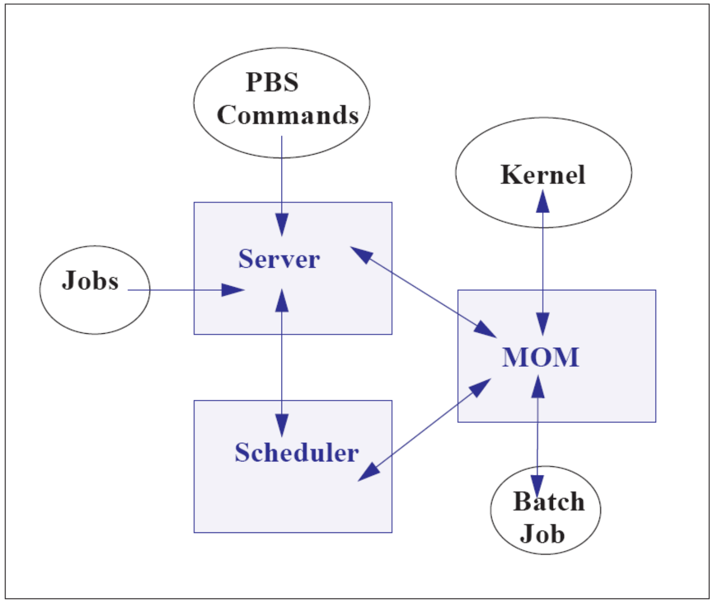
- Server or *pbs_server* is the central focus for a PBS system.
- All communication to the *pbs_server* are through an Internet Protocol network.
- *pbs_server* provides basic services such as:
  - Receiving/ creating a batch job
  - Modifying an existing job
  - Protecting the job against system crashes
  - Running of the job
- Usually there is one server instance managing a pool of resources.



PBS : Architecture Overview … 2

Job Executor :

- Job Executor or the *pbs_mom* is the component that actually places the job into execution.
- *pbs_mom* is informally called MOM (mother of all executing jobs).
- MOM places a job into execution when it receives a copy from a server.
- MOM creates a new session that is as identical to the user session as possible.
- MOM has the responsibility for returning the job's output to the user when directed by the *pbs_server*.
- One MOM runs on each node which will execute PBS jobs.



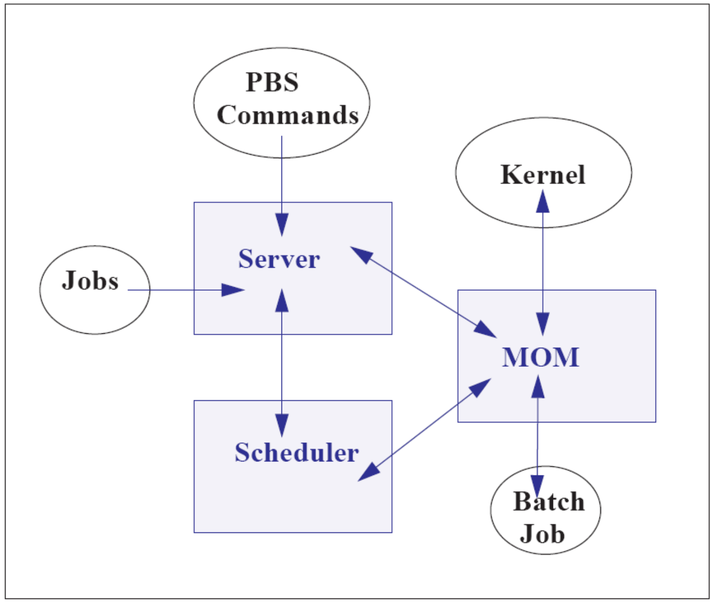
PBS : Architecture Overview … 3

Scheduler :

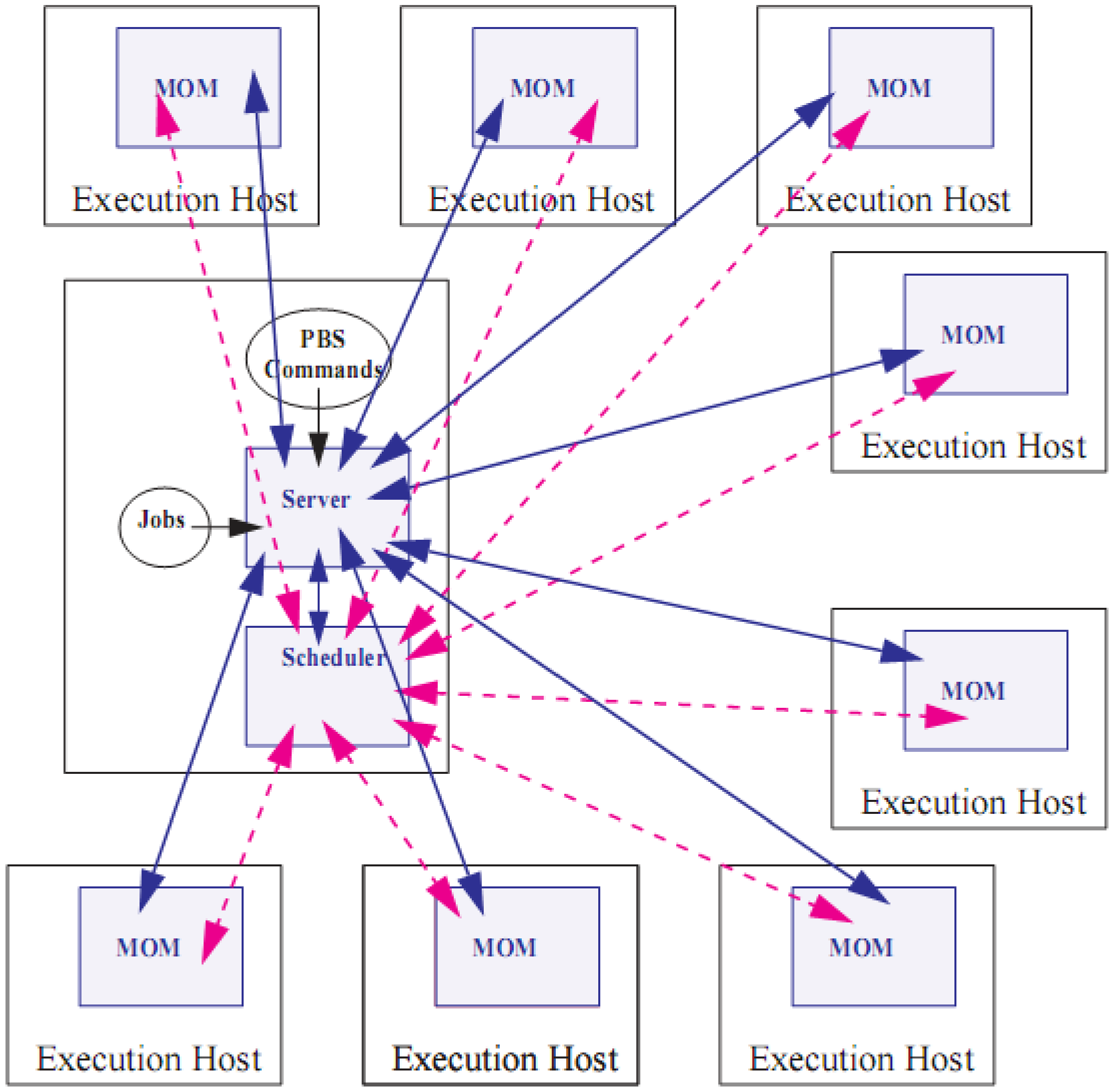
- The scheduler or *pbs_sched*, implements the site's policy controlling when each job is run and on which resources.
- *pbs_sched* communicates with various MOMs to query the state of the system resources and the Server to determine the availability of jobs to execute.
- The interface to the server is through the same API as used by the client commands

Commands :

- User commands that any user can use to submit, delete, query jobs and access resources.,
- Operator commands and Manager commands that provide administrative access to the PBS system.



Usual PBS setup in Clusters



Common PBS User commands

Commands	Purpose
qalter	Alter job(s)
qdel	Delete job(s)
qhold	Hold job(s)
qmsg	Send a message to job(s)
qmov	Move job(s) to another queue
qrls	Release held job(s)
qrerun	Rerun job(s)
qselect	Select a specific subset of job(s)
qsig	Send a signal to job(s)
qstat	Show status of job(s)
qsub	Submit job(s)

Topics

- Introduction
- CPU Scheduling
- Intro to Workload Management Systems
- Scheduling Algorithms
- Portable Batch System : PBS
- LoadLeveler
- Platform LSF
- Summary – Materials for Test

LL : IBM LoadLeveler

- LoadLeveler is a job management system that matches user jobs to available resources while optimizing resource utilization.
- LoadLeveler has a variety of interfaces that enable:
  - users to create, submit and manage jobs
  - allow system administrators to configure the system and control running jobs.
- For LoadLeveler to schedule job on a machine, the machine must be a valid member of a LoadLeveler cluster.
- Each machine in the LoadLeveler cluster performs one or more role in the scheduling of jobs.

LL Cluster : Machine Roles

Submitting Machine:
- Participates in the LL Cluster on a limited basis, and has the basic client command-line utilities installed
- Users can use this resource to create, submit, and manage their jobs.

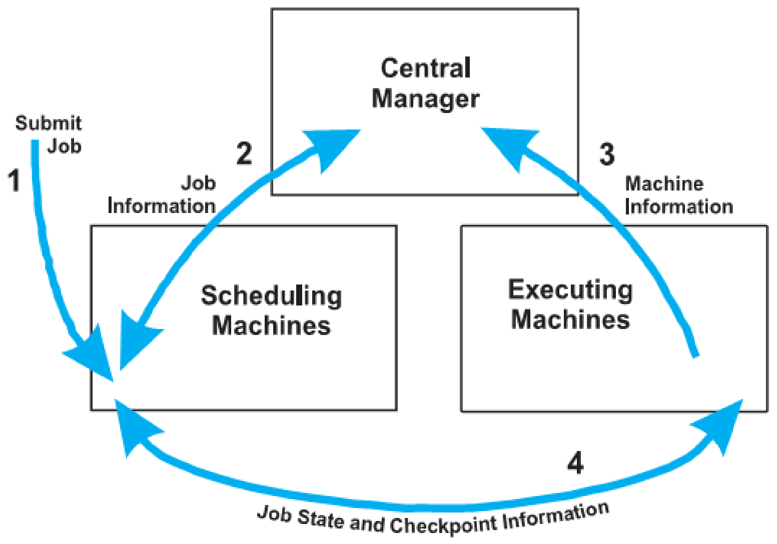
Scheduling Machine:
- When a job is submitted, it is placed in a queue managed by the scheduling machine.
- The scheduling machine contacts the Central manager machine to find a suitable machine that can run the job and retains the persistent information about the job.

Central Manager Machine :
- Examines the job requirements and finds one or more machines in the LL cluster that will run the job.

Executing Machine :
- The machine that runs the job is known as the executing machine.

LL : Process Overview

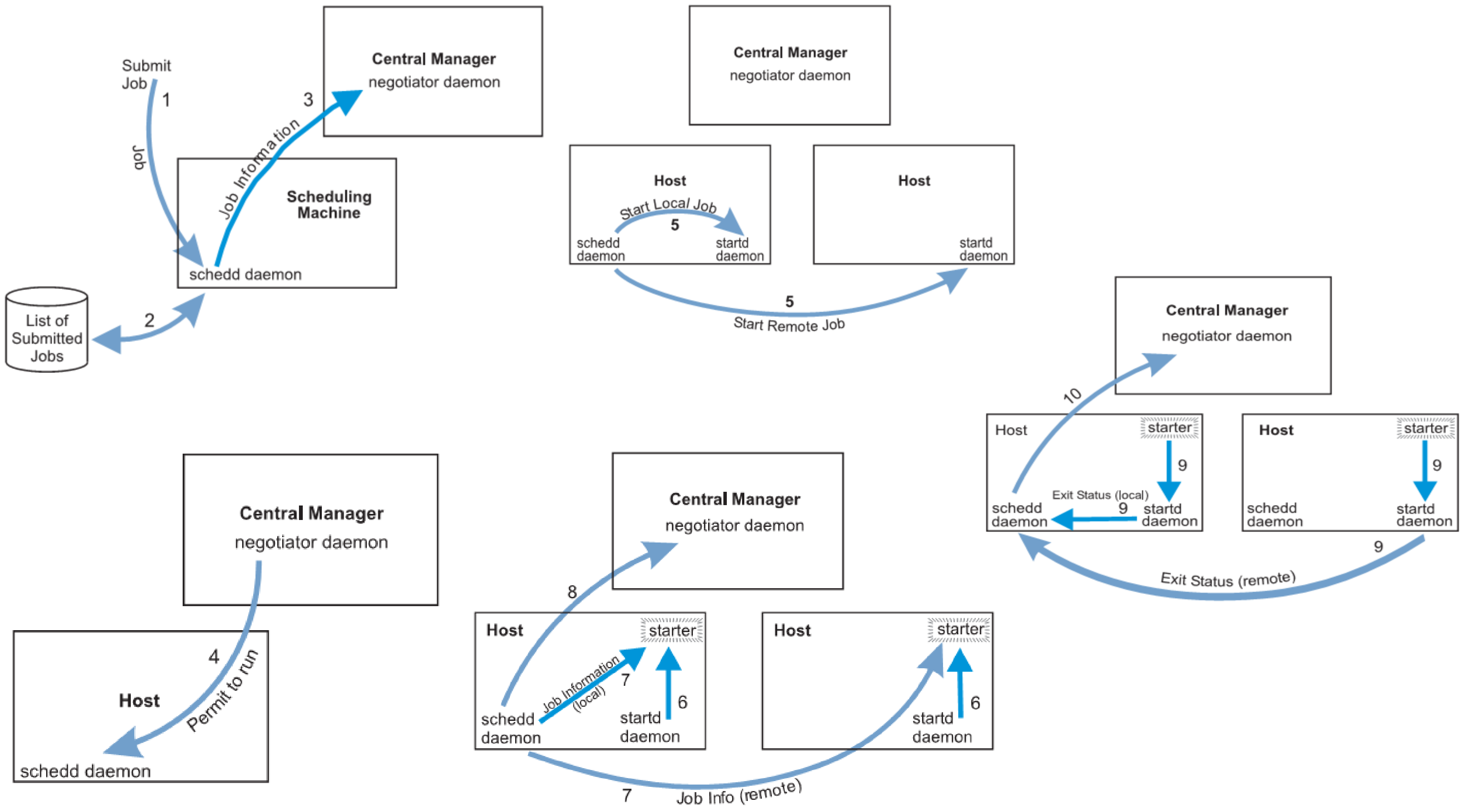
- The user submits the job from the submitting machine to the scheduling machine
- The job is usually characterized by the at least 2 main kinds of information, the resource requirements, and the job description (executable, precedence, arguments etc.)
- The Scheduling Machine adds the job to a job queue where a list of jobs are waiting to be executed.
- The scheduling machine contacts the central manager to determine a corresponding match
- When a match is found the Central manager sends the information to the scheduling machine.
- The job is sent to the executing machine(s) for further processing



LL : System Processes

daemon	Description
LoadL_master	The master daemon that runs on all the machines in the cluster whose primary responsibility is to manage other LL daemons.
LoadL_schedd	The scheduler daemon that receives jobs from the llsubmit command and manages them on the resource selected by the negotiator daemon.
LoadL_startd	Monitors Job and resource status and forwards information to the negotiator daemon. This daemon spawns a starter process that manages a job on the executing machine.
LoadL_negotiator	Monitors status of each job and resource in the cluster. Responds to queries llstatus and llq. Runs on the central manager
LoadL_kbdd	Keyboard daemon that monitors keyboard and mouse activity.
LoadL_Gsmonitor	Monitors for down machines based on the heartbeat responses.

LoadLeveler Job Lifecycle



Common LL User Commands

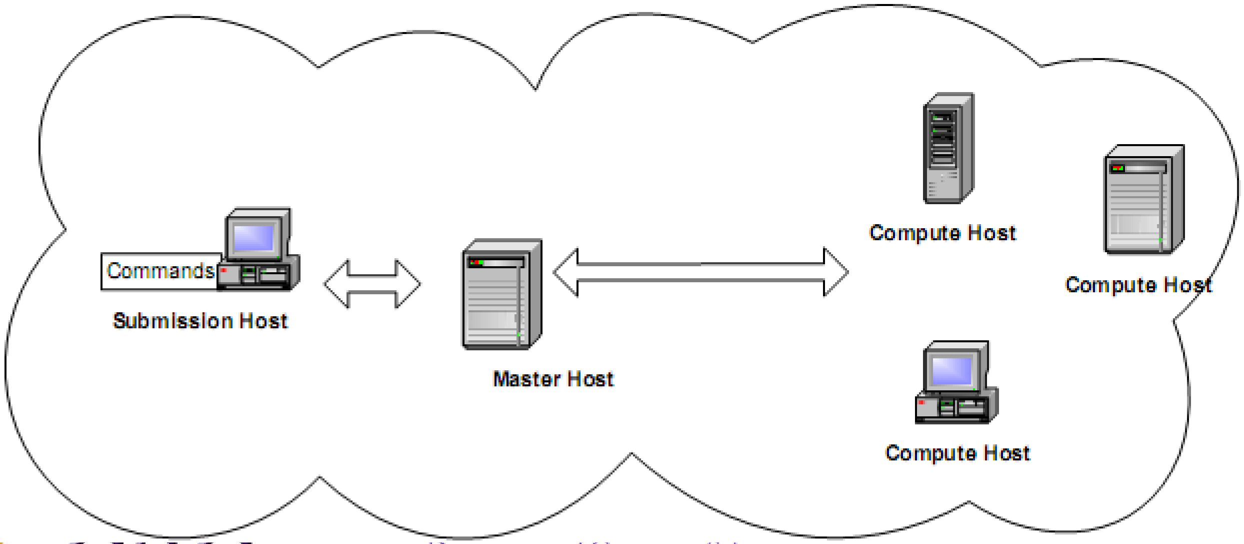
Commands	Purpose
llbind	Bind job steps to a reservation
llcancel	Cancel a submitted job
llchres	Change attributes of a reservation
llchkpt	Checkpoint a running jobstep
llhold	Hold or release a submitted job
llmkres	Make a reservation
llmodify	Change attributes of a submitted job
llpreempt	Preempt submitted jobstep
llq	Query Job Status
llstatus	Query machine status
llsubmit	Submit a Job
llsummary	Return job resource information for accounting

Topics

- Introduction
- CPU Scheduling
- Intro to Workload Management Systems
- Scheduling Algorithms
- Portable Batch System : PBS
- LoadLeveler
- Platform LSF
- Summary – Materials for Test

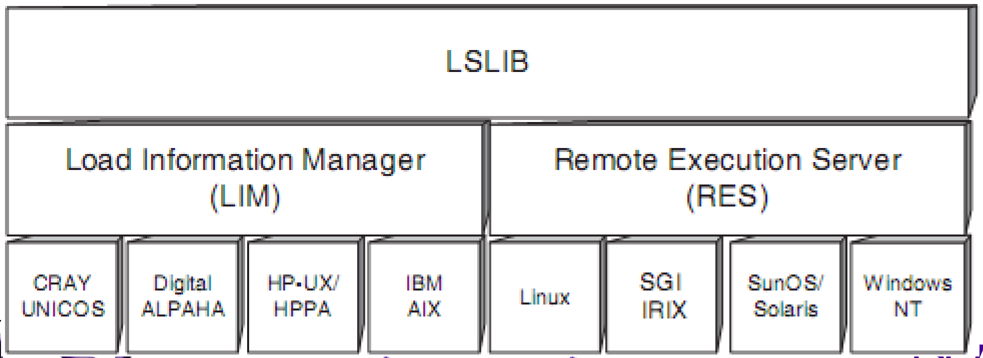
Platform LSF

- Platform LSF is a workload management system that forms a layer on top of the underlying operating systems like Windows and Linux in a uniform manner so all the computing resources on a network can be efficiently managed and used.
- From a user perspective, using the LSF system consists of the same steps, i.e. using a submission host to submit a batch job to a scheduling host that manages the execution of the job on the compute resources.



LSF Overview

- From a system perspective, LSF Base forms the core of the system providing resource management, host selection, job placement, transparent remote execution and file staging.
- LSF Base includes :
  - Load Information Manager (LIM) : Monitors and reports load information from each host to the master LIM. Master LIM provides this information to all other applications.
  - Process Information Manager (PIM)
  - Remote Execution Server (RES) : Resident on each host the RES accepts remote execution requests and provides fast, transparent and secure remote execution of tasks.
  - LSF Base API , and lstools, lstcsh, lsmake
- LSF Batch extends the LSF Base to provide batch services such as load balancing and policy driven resource allocation control.

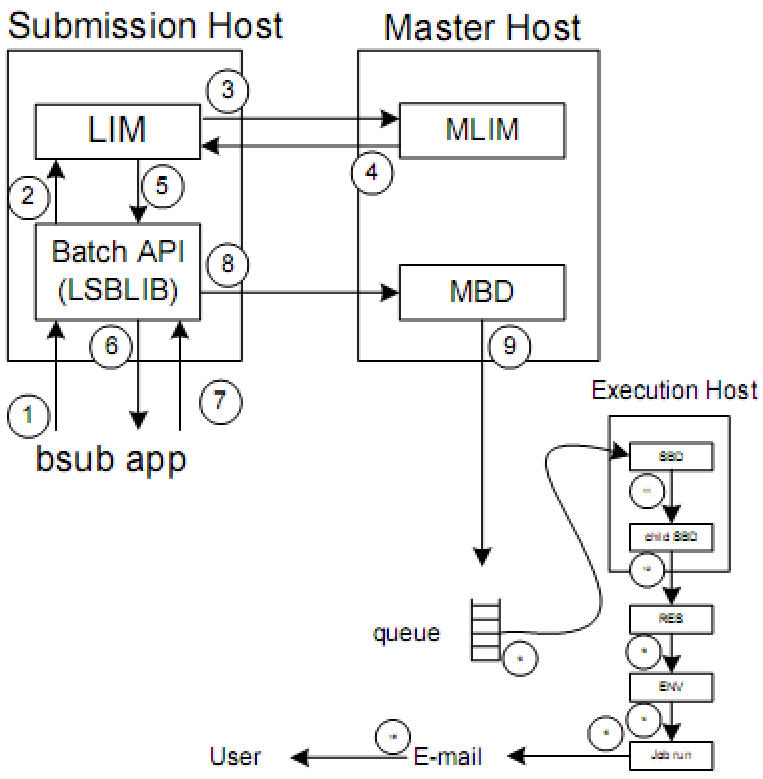


LSF System Processes

Daemons	Description
mbatchd	Master batch daemon, running on the master host, is responsible for overall state of the jobs in the system. Receives job submission and information query requests, manages jobs held in queues, and dispatches jobs to hosts as determined by *mbsched*
mbschd	Master batch scheduler daemon running on the master host works with the *mbatchd* to make scheduling decisions based on job requirements and policies.
sbatchd	Slave batch daemon running on each host receives run requests from *mbatchd* and manages local execution of job. Responsible for enforcing local policies and maintaining the state of the jobs on the host.
res	Remote execution server running on each host accepts remote execution requests to provide transparent and secure remote execution of jobs and tasks
lim	Load information manager running on each host collects host load and configuration information and forwards it to the master LIM running on the master host.
pim	Process information manager running on each host provides information such as CPU and Memory utilization by the job and reports the information to *sbatchd*

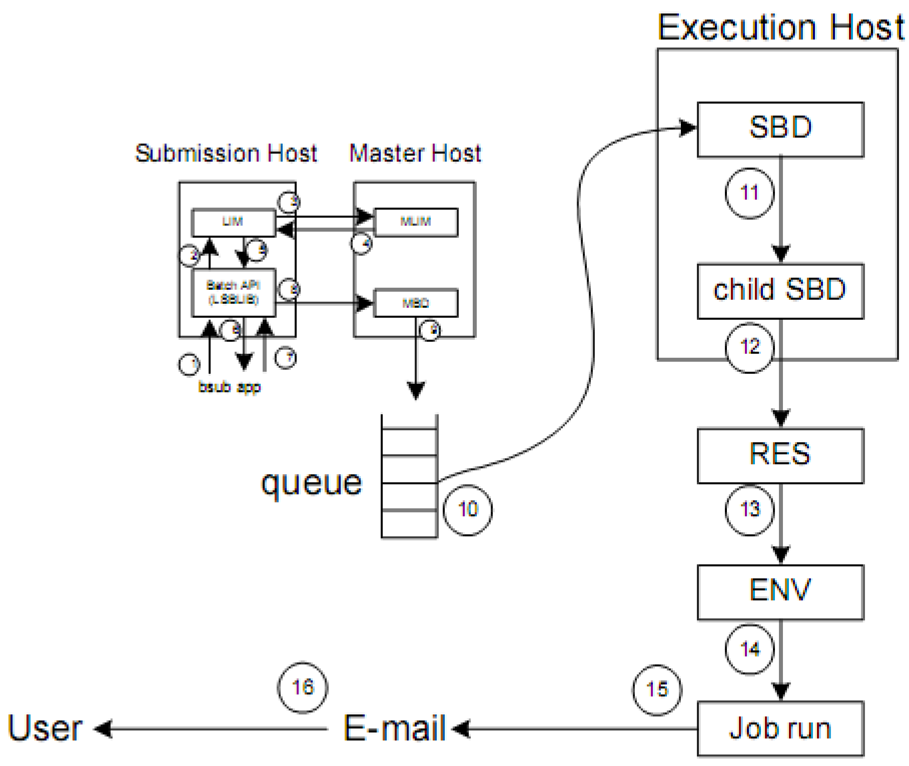
LSF Process Overview … 1

1. *bsub* or *lsb_submit()* submits job to LSF for execution
2. After basic steps such as LSF syntax checks etc the Job is parsed and forwarded to the submit host's LIM .
3. The submit host LIM communicates the job's information to the cluster's master LIM (managed by *mbatchd*).
4. The master LIM determines the best host to run the task and sends this information back to the submission host's LIM.
5. Information about the chosen execution host is passed through the LSF Base Library
6. Information about the host to execute the task is passed back to *bsub or lsb_submit()*.



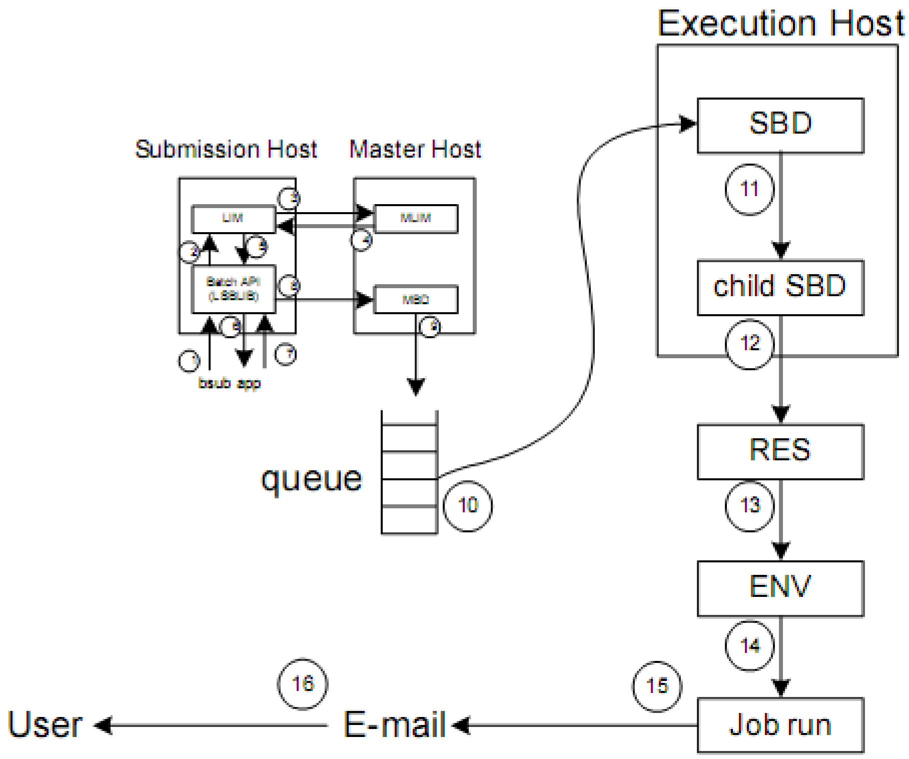
LSF Process Overview … 2

7. To enter the batch system, *bsub* or *lsb_submit()* sends the job to *LSBLIB*
8. Using *LSBLIB*, the job is sent to the *mbatchd* running on the cluster's master host
9. The mbatchd puts the job in a queue and waits for an appropriate time to dispatch the job.
10. *Mbatch* dispatches the job when its corresponding resource match is met, when more than one resource is available the best host is chosen. The sbatchd on the execution host recieves this job request.
11. Once the sbatchd receives the jon, it controls the jobs execution and reports job status to the mbatchd. The sbatchd creates a child sbatchd to handle the job.



LSF Process Overview … 3

12. The child sbatchd sends the job to the RES
13. Child RES creates an execution environment to run the job.
14. Job is run in the execution environment.
15. The child sbatchd sends completed job information back to the sbatchd.
16. The output is sent from the sbatchd to the user. The child sbatchd and the execution environment are destroyed.



Common LSF User Commands

Commands	Purpose
bsub	Submitting Jobs
bmod	Modifying submitted jobs
bkill	Killing running/submitted jobs
bstop/bresume	Suspending and Resuming Jobs
bbot	Moving a job to the bottom of the queue
btop	Moving a job to the top of the queue
bjobs	Display the status of jobs in the LSF systems
bhist	Displays history of a job from the moment it was submitted.
bpeek	Allows user to look at the output that the job has produced so far.
blimits	Displays user level accounting information.

Topics

- Introduction
- CPU Scheduling
- Intro to Workload Management Systems
- Scheduling Algorithms
- Portable Batch System : PBS
- LoadLeveler
- Platform LSF
- Summary – Materials for Test

Summary – Material for the Test

- Intro. Concepts – Slides 4,5
- CPU Scheduling – Slides 7,8
- CPU Scheduling Algorithms – Slides 9 -17
- Workload Management Systems - Slides 22-27
- Scheduling Algorithms – Slides 29 - 49

Key Sources :

– PBSPro 8.0 Manual
– LSF 6.2 Manual
– LoadLeveler Manual V3 release 3
– Job Scheduling on Parallel Systems – Jonathan Weinberg (http://www.cs.ucsd.edu/users/j1weinberg/papers/weinberg_schedulingSurvey.pdf)
– Henri Casanova (http://navet.ics.hawaii.edu/~casanova/)
– CPU Scheduling : Operating System Concepts - Silberschatz, Galvin, Gagne