HPC: Parallel hardware

LECTURER: UROŠ LOTRIČ

ASSISTANT: DAVOR SLUGA

Architecture

A parallel computer

- Uses multiple processing elements simultaneously in a cooperative manner

Parallel processing

- Techniques and technologies that make computing in parallel possible
- Hardware, networks,
- Operating systems, parallel libraries, languages, compilers, algorithms, …

Parallel computing is an evolution of serial computing

- It is only possible when a problem is inherently concurrent
  - Can be split into tasks which can execute at the same time
- Dependency must be enforced by synchronization

Architecture

Flynn's characterization

- SISD: standard non-parallel (scalar) processor
- SIMD: array processor
  - Array of functional units and a shared controller
- MIMD: separate instructions streams operating on separate data
  - Multi-core, multi-processor, multiple computers, heterogeneous computer
  - Shared and distributed memory and combinations
- MISD: only theoretical
- SIMT
  - Single instruction multiple threads (inside warp)
  - Tiled SIMD: each processor emulates multiple threads using masking, blocks of threads share a control processor, divergent flow is not beneficial
  - Preferred to access the same cache line to improve performance (opposite to classical multi-core systems)

Architecture: von Neumann Architecture

Instructions and data are stored in memory

Machine cycle

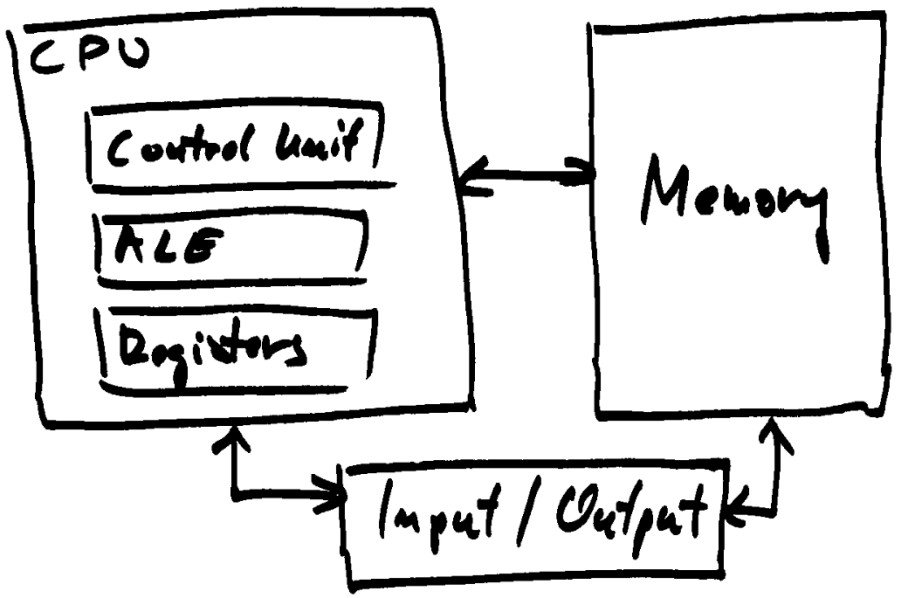
- Fetch
- Decode
- Execute
- Store

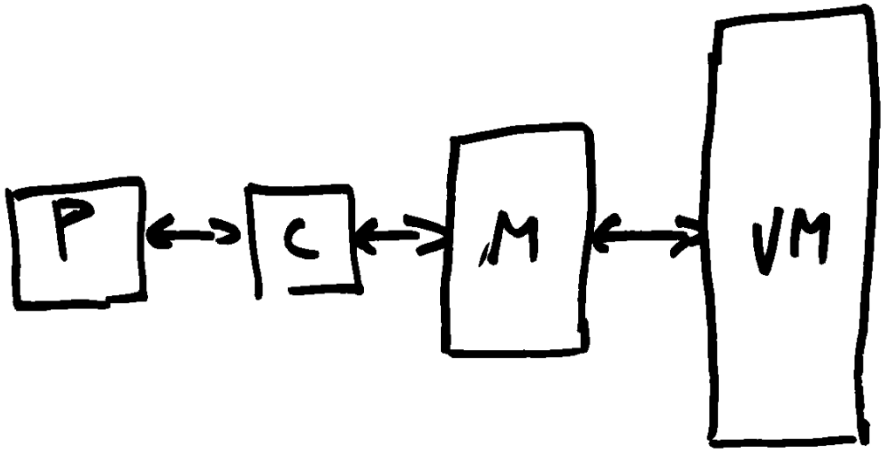
Single instruction, single data

Memory bottleneck

Improvements

- Memory hierarchy
  - Main memory, cache, virtual memory
  - Temporal and spatial locality
- Parallelism in hardware
  - Pipeline, Superscalar execution, Speculative execution,
  - Multiple functional units, Hardware multithreading



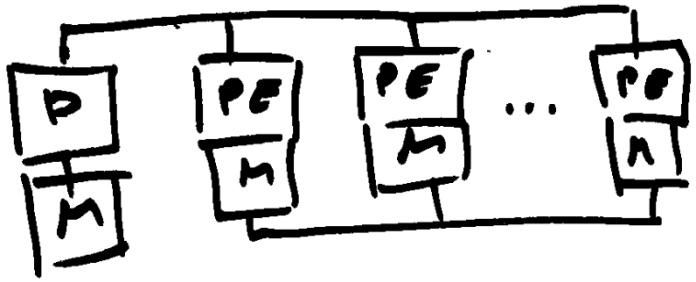


Architecture: vector processor

Single instruction, multiple data

First supercomputers

- Data parallel problems
- Operate on vectors
  - Size of problem must fit to the hardware
- Long and expensive development



Architecture: vector processor

Scalar support

Vector support

- Registers, functional units, instructions
- Operand alignment in memory
- Memory interleaving for faster access

Problems

- Size of problem and number of processing elements must fit
- Branching and masking
  - Slow execution
  - Packing groups threads with same result to avoid masking
- Poor scalability

```
if (a&1)
    a = 3*a + 1;
else
    a = a/2;
```

```
p = (a&1);
t = 3*a + 1;
if (p) a = t;
t = a/2;
if (!p) a = t;
```

Architecture: shared memory systems

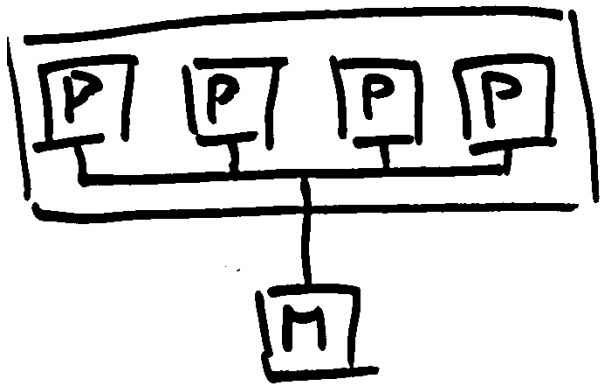
Multiple instructions, multiple data

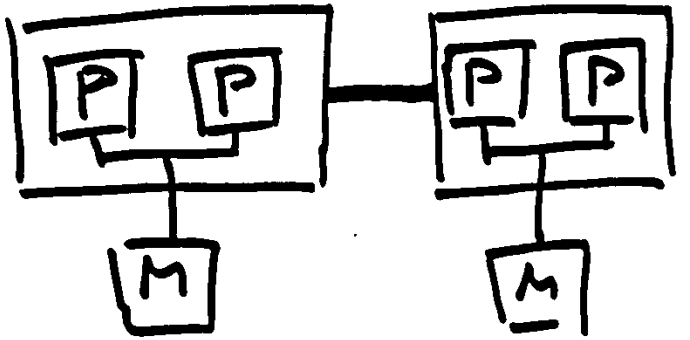
Share common memory

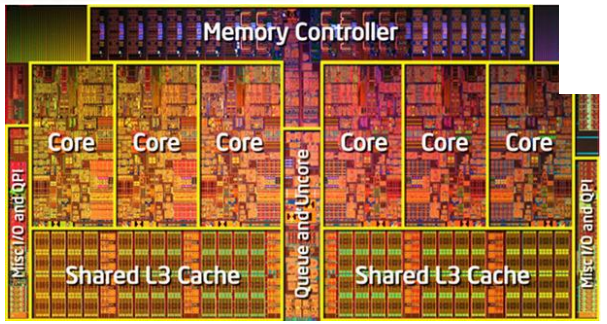
- Processors operate independently
- Processors can access each others' memory
- Changes in memory by one processor are visible to the others

UMA vs. NUMA

- Complexity and scalability
- Memory access times
- Cache coherence and false sharing may be magnified with NUMA
- UMA = SMP (Symmetric Multi Proc.)







Architecture: shared memory systems

Memory access

- One processor to one global memory module
- Memory divided to many modules evenly distributed among processors

Cache

- Each core has its own cache
- Cache coherence must be assured
  - Snooping
  - Directory protocols

Architecture: distributed memory

Multiple instructions, multiple data

Processors have their own memory

Other processors do not see memory changes

Processors exchange data by sending messages

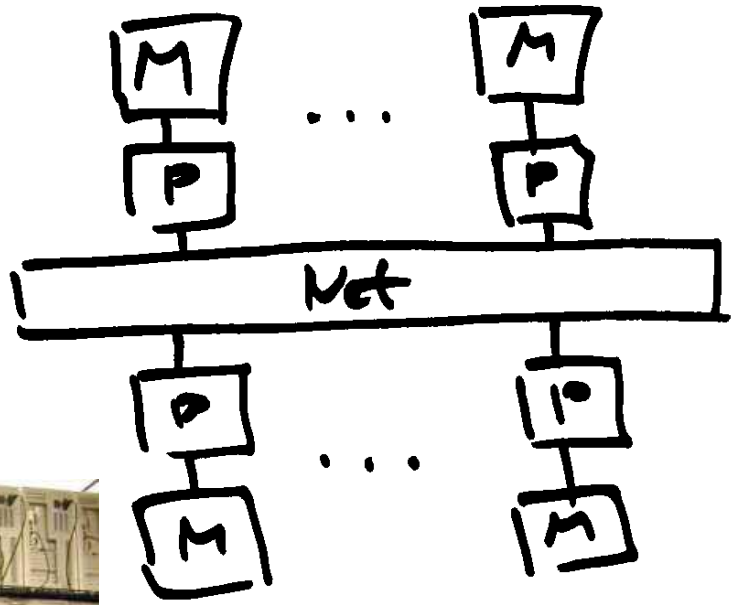
Slower compared to shared memory systems

More scalable

Focus on interconnections

Cost effective

- off-the-shelf hardware components



Architecture: distributed memory

Cluster

- Commodity network components (Ethernet, Infiniband)
- Each node runs its own OS
- Tainhe-2b (all nodes run Kylin Linux)

Massively parallel processors (MPP)

- More tightly integrated
- Can run same instance of OS on all nodes
  - Treats the whole machine as single OS from admin point of view
- More commonly only special interconnects
- TaihuLight (has special architecture, special OS)

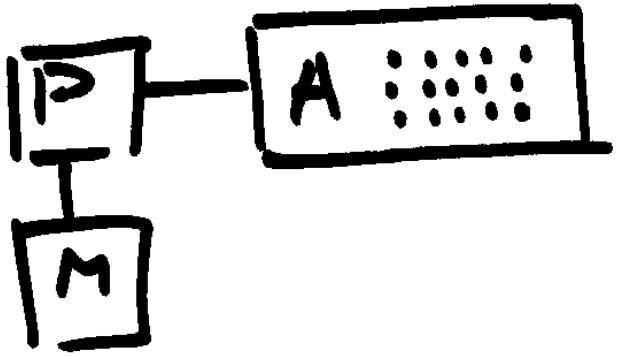
Constellations

- More cores per node than nodes
- Spatial node sharing among users

Architecture: accelerators

Offload processing

- Host sends data to accelerator
- Host triggers computation on an accelerator
- Accelerator performs computation
- Host retrieves results



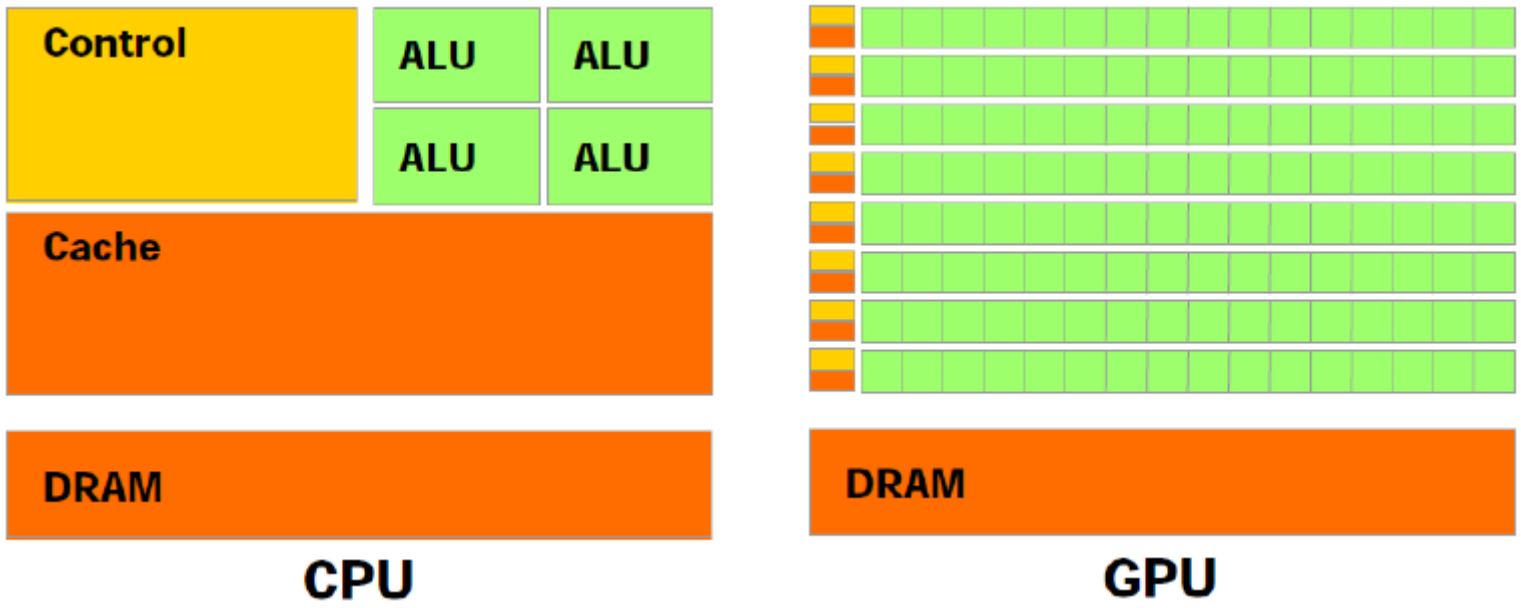
Today extremely popular model

Single instruction, multiple threads

Hierarchical design

Architecture: accelerators

Pros and cons of CPU and GPU



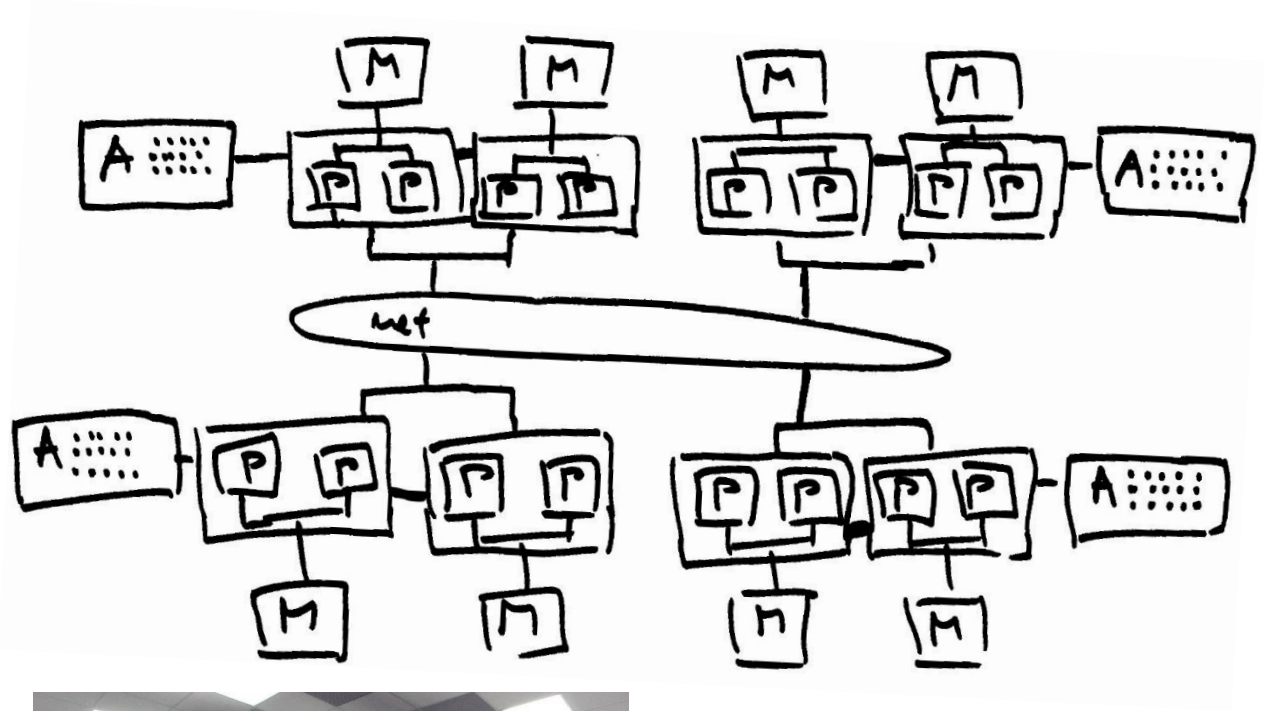
Architecture: modern systems

Modern computing resources

- Hierarchical organization
  - Connects many nodes
  - Shared memory within a node
  - Offload systems on some nodes
  - Message passing between nodes
- Heterogenous systems

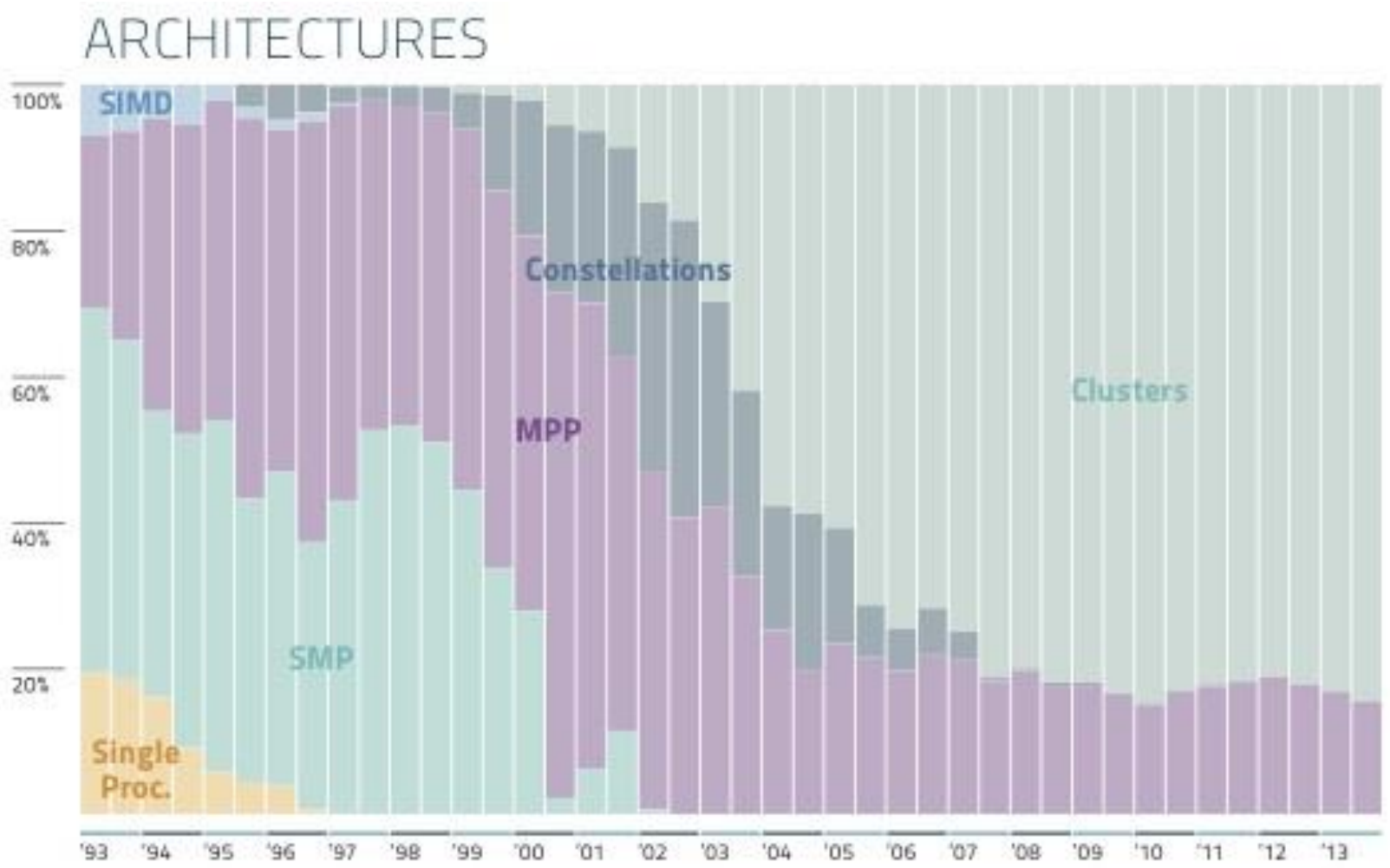
Programming

- Reflects hardware organisation
- Different programming concepts
  - Programming languages
  - Libraries
  - Algorithms

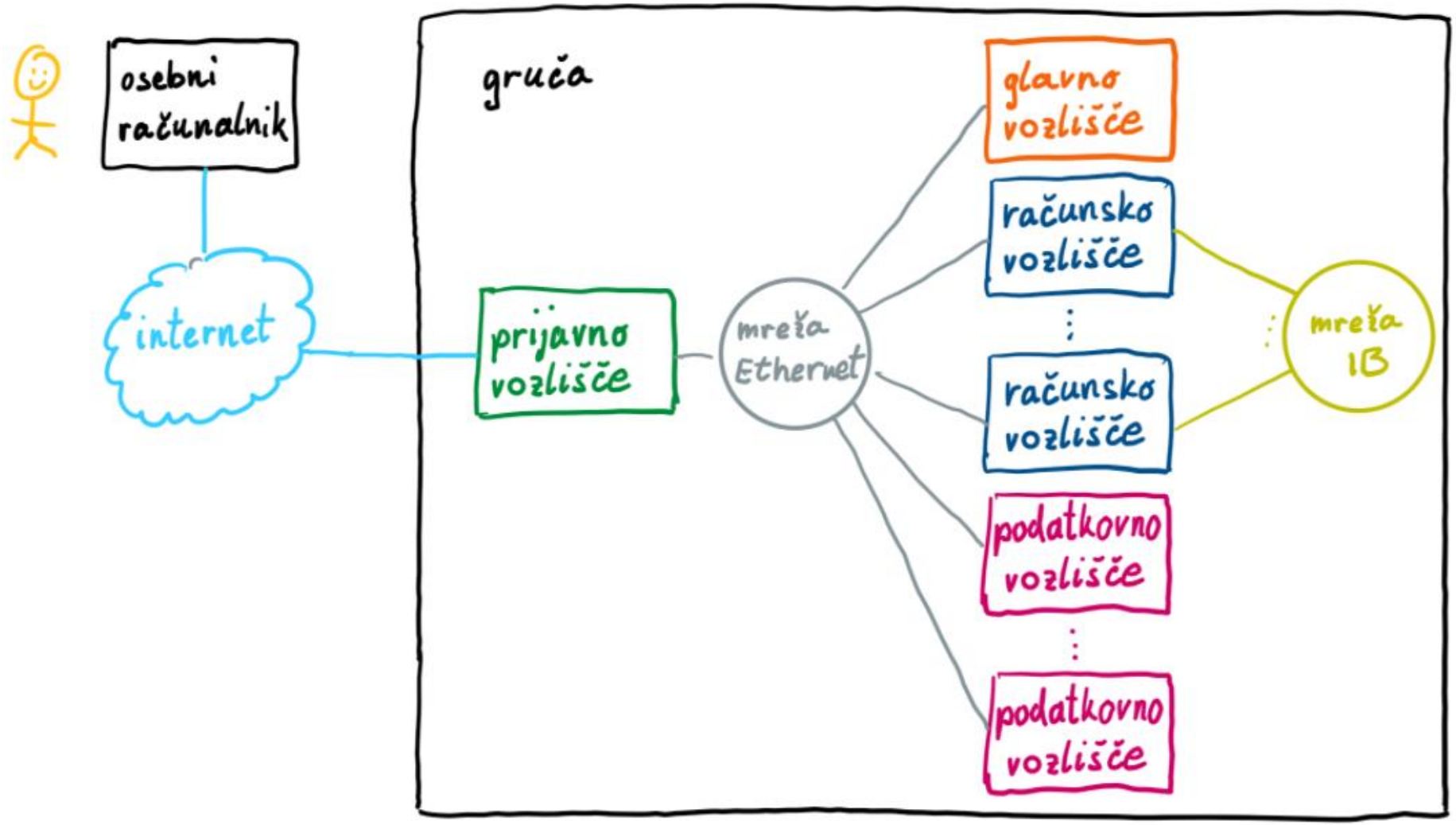


Architecture: overview

Trends



Architecture: NSC cluster



HPC: Middleware

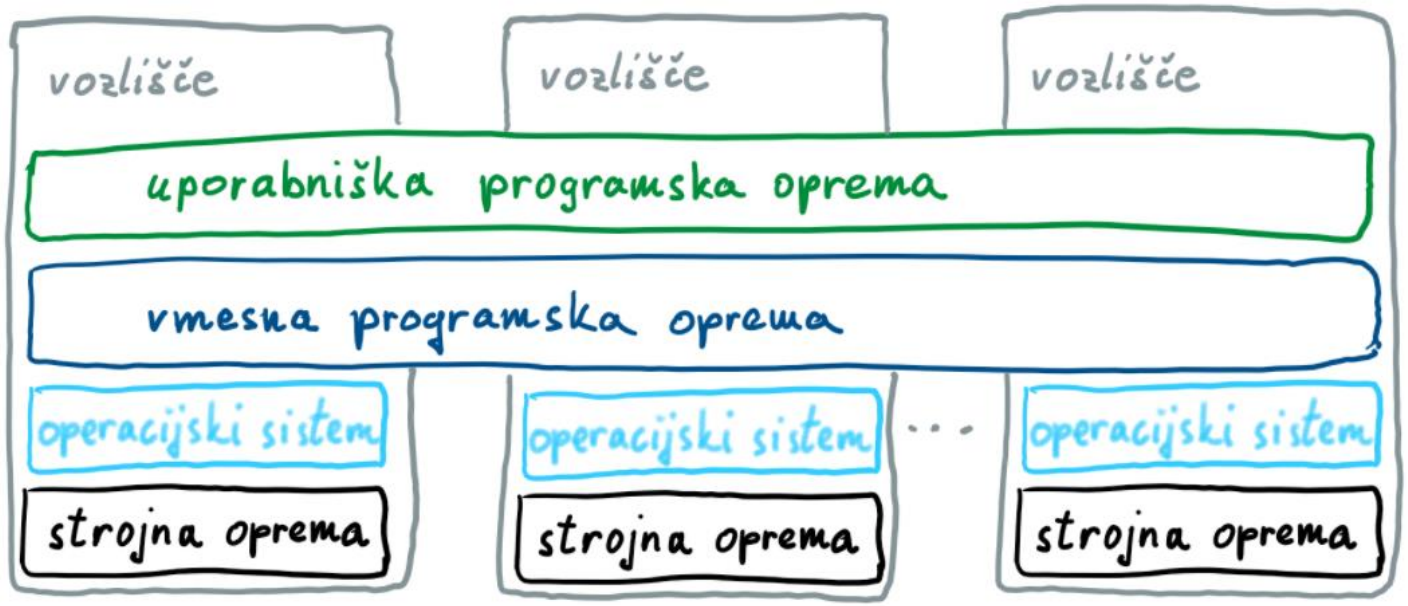
Cluster Software

Focus on Open Source software

- Linux CentOS, Fedora
- Automation through Foreman, Puppet, Ansible
- Administration and metrics:
  ElasticSearch, Syslog, Icinga, Nagios, IPMI
- Storage and data handling:
  dCache, Ceph, Rucio, iRODS, BeeGFS
- Job scheduling: SLURM
- Job submission system: ARC
- Container and virtualization support:
  Singularity, Proxmox, OpenStack


Cluster software

- Job scheduling: SLURM
- Job submission system: ARC



User software

Support for a wide range of applications

- HPC, BigData, AI

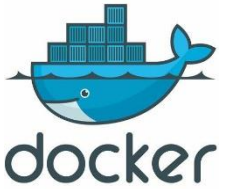
Basic software pre-installed

- GNU toolchain, MPI, Matlab

Environmental modules

Virtualization primarily through containers

- Singularity
- Provide a repository of commonly used images for HPC
- Each user tweaks an existing or builds its own image according to his needs



User software

Containers

SLURM

Simple Linux Utility for Resource Management

Cluster management and job scheduling software

- Local resource management software (LRMS)
- Open source
- Fault tolerant
- Highly scalable
- A lot of plugins (accounting, network, MPI)

Key functions

- Allocates access to resources (compute nodes) to users
- Framework for starting, executing and monitoring work
- Arbitrates contention for resources by managing a queue of pending work

SLURM

Resource manager

- Needed in a parallel computer to execute parallel jobs
- Allocates resources within a cluster
  - Nodes (unique IP)
    - Sockets, cores, threads
    - Memory
    - Interconnect
    - Features (GPUs, bigmem)
  - Licences
- Manages jobs
- When there is more work than resources, the RM manages queue
  - Complex scheduling algorithms
  - Resource time-limit

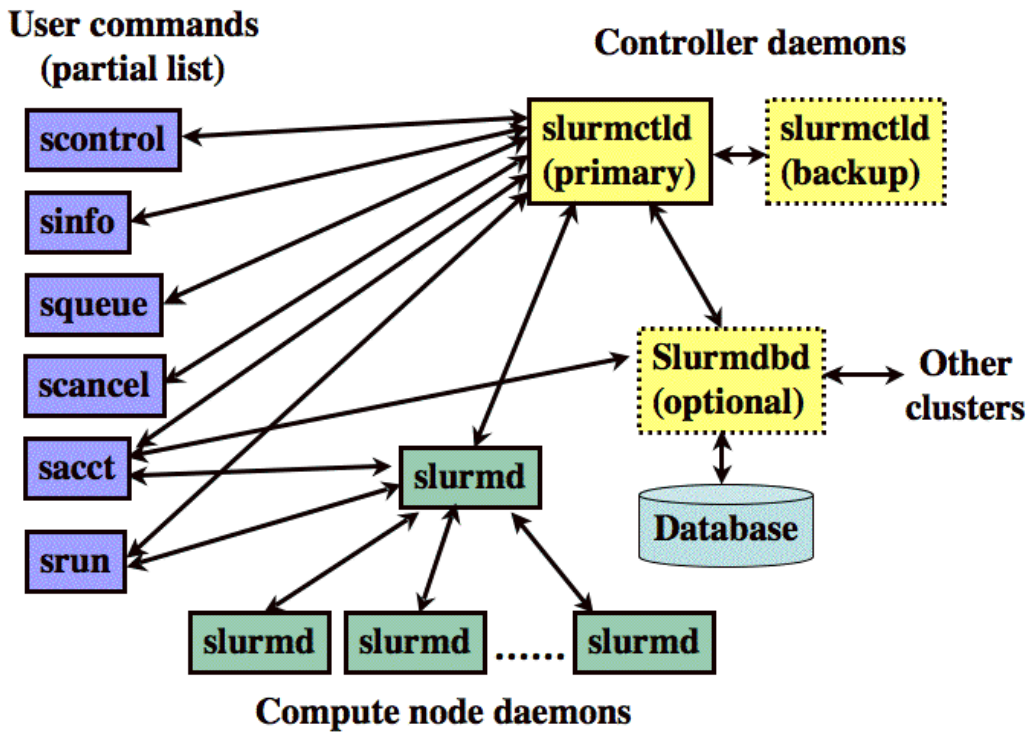
SLURM: architecture

Slurmctld deamon on management node
- Monitors resources
- Manages job queues
- Allocates resources
- Optional fail-over twin

Slurmd deamon runs on each node
- Similar to remote shell
  - waits for work, executes work, reports status
- Hierarchical concept
- Fault-tolerant communication
- Starts Slurmstepd deamon that runs jobs on nodes

Slurmdbs
- Database deamon to record accounting information

Usertools
- control cluster (admin), work with jobs, check queue, …



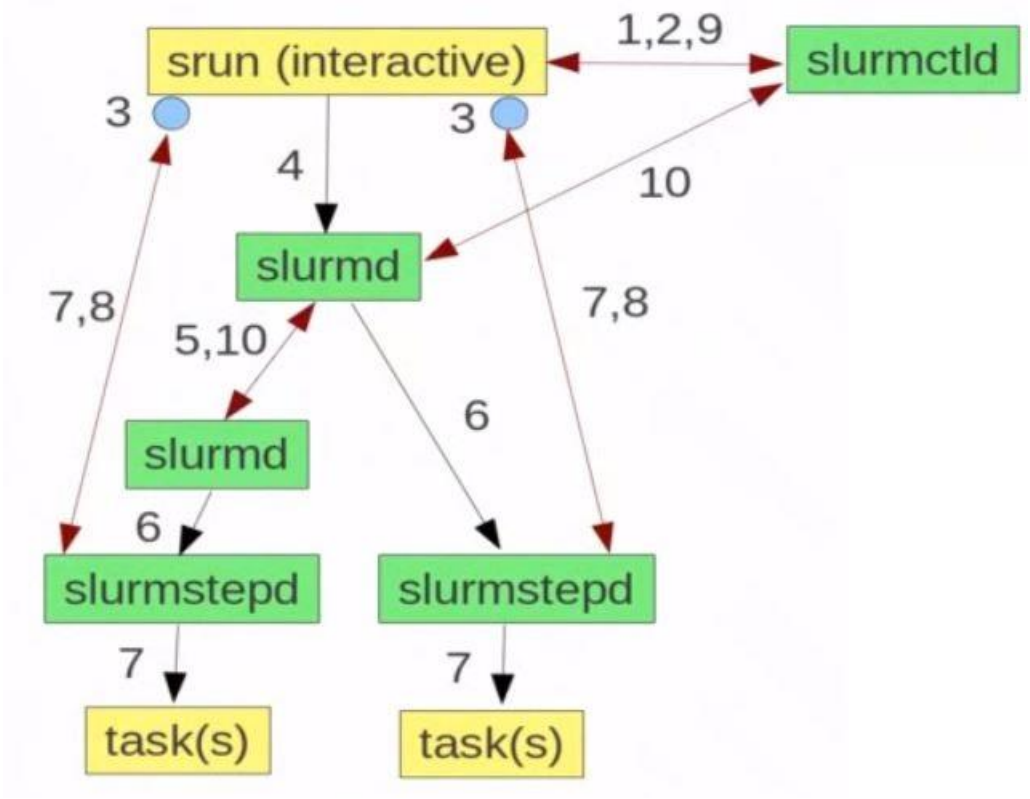
SLURM: jobs

Job description

- ID, name, time limit, size specs, special node features

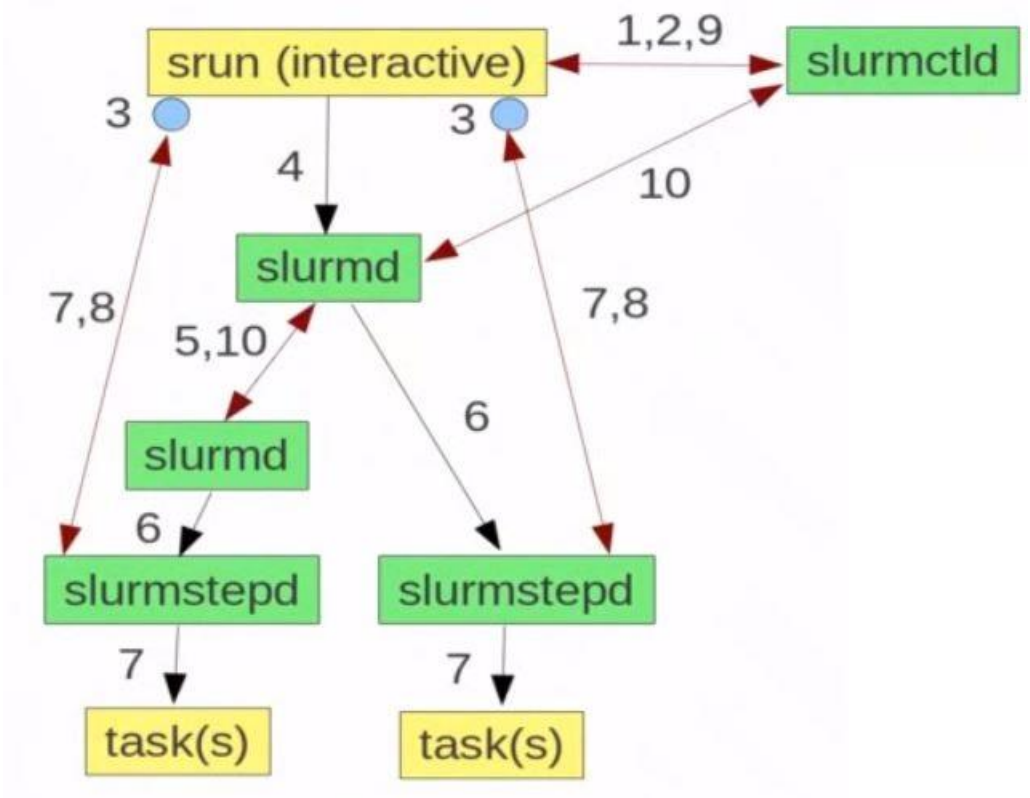
Job execution sequence

1. srun sends job allocation request, slurmctld grants it
2. srun sends job step create request, slurmctld gives credentials
   - credentials include resources that have been allocated
3. srun opens socket for IOs
4. srun sends credentials and task info to slurmd
5. slurmd forwards allocation to required nodes



SLURM: jobs

6. slurmd starts slurmstepd
7. slurmstepd connect IOs to srun IOs and launches tasks
8. slurmstepd notifies srun on task termination
9. srun notifies slurmctld on job termination
10. slurmctld verifies termination via slurmd and releases resources



SLURM: architecture

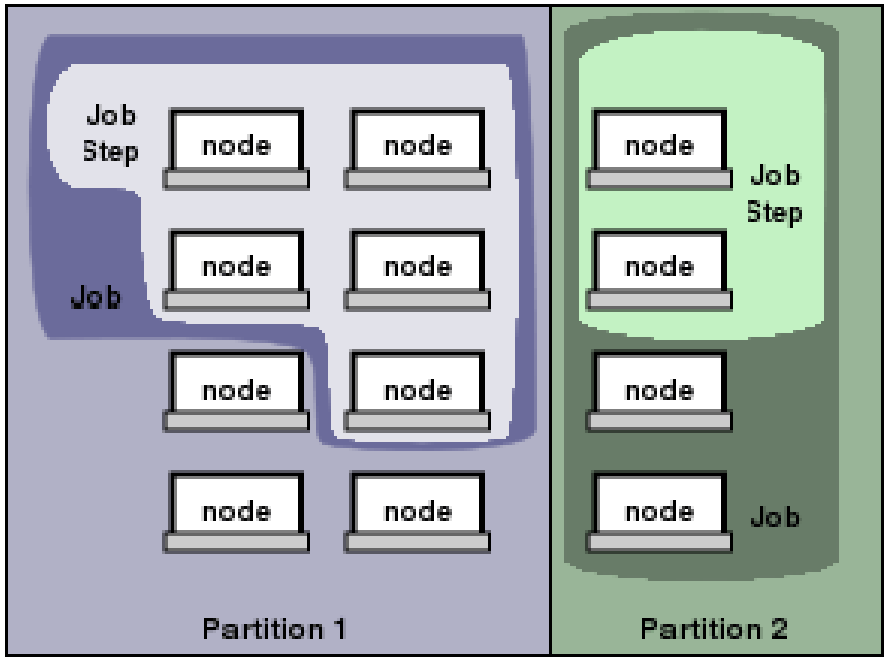
Nodes: compute resources

Partitions: logical groups of nodes

- One node can reside in multiple partitions
- = job queue with limitations (size, time, users, …)

Jobs: allocations of resources assigned to a user for some time

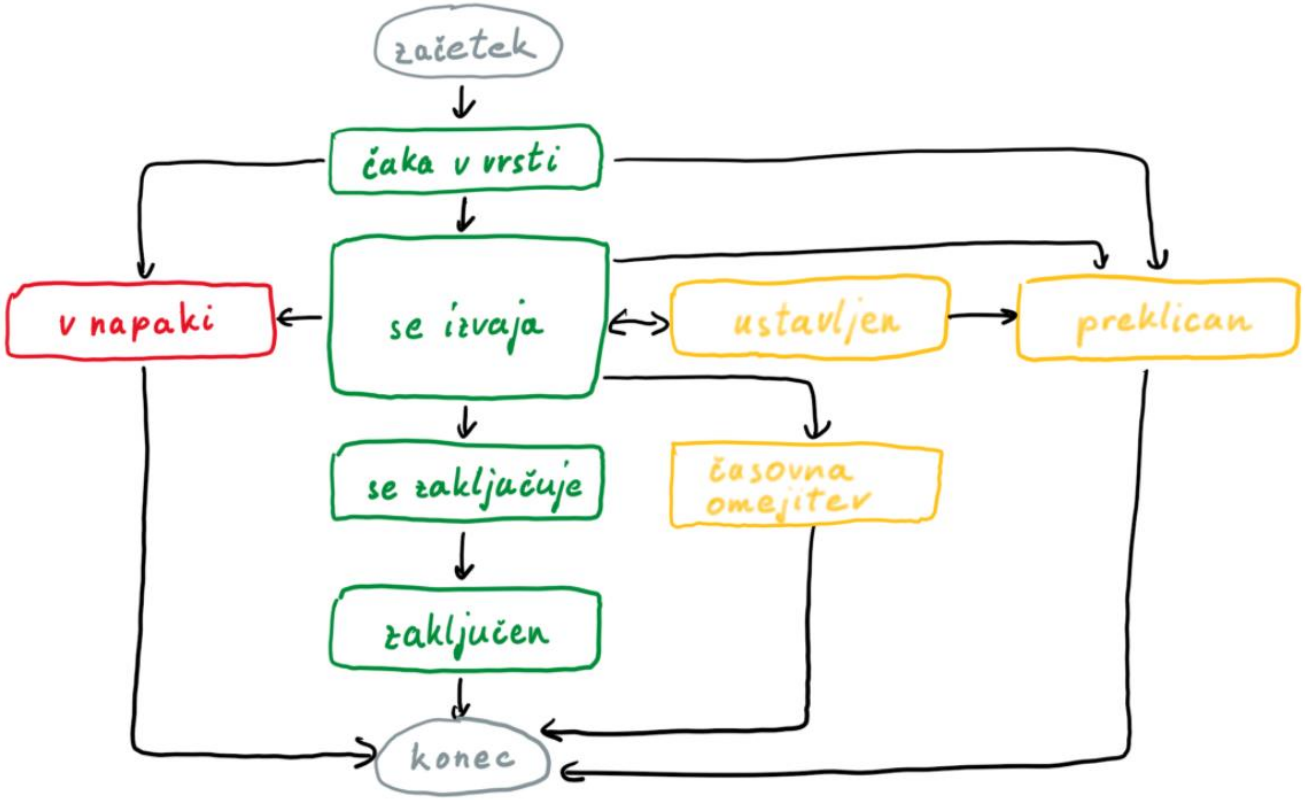
Job steps: sets of (possibly parallel) tasks within a job



SLURM: jobs

squeue

- job states
  - **PD: pending, R: running,**
  - CA: canceled, CF: configuring
  - **CG: completing, CD: completed**
  - F: failed, TO: timeout, NF: node failure,
    RV: revoked, SE: special exit state,
    S: suspended, …
- job reasons
  - Resources: job is waiting for compute nodes to become available
  - Priority: jobs with higher priority are waiting for compute nodes
  - ReqNodeNotAvail: the requested nodes are not available
    (cluster downtime, nodes offline, …)



SLURM: nodes

sinfo (node states)

- down, draining, drained, failing, fail, reboot, maintenance, power…
- **idle**, **allocated**, **mixed**, **completing**, **reserved**
  - idle: all cores are available on the compute node
  - mix: at least one core is available on the compute node
  - alloc: all cores on the compute node are assigned to jobs
  - * node is not responding, will not take new workload

Features/constraints

- Enables users to make specific requests
- sinfo --Format Features
- NSC features: AMD, intel, gpu, k40, bigmem
- -C / --constraint=…

SLURM: important commands

srun: create job allocation and launch job step

- blocks the shell

sbatch: submit script for later execution (batch mode)

- does not block the shell
- many possibilities for job specification
  - job dependence (flag --dependency)
  - run many jobs with different params …

sinfo: status of nodes

squeue: status of jobs and job steps

scontrol: job control (hold, release, show nodes…), system config

SLURM: important commands

salloc: starts shell on the first node that corresponds to given requirements

- interactive work
- resources are specified in the same way as for srun/sbatch
- exit to release resources

sstat: statistics of active job

sacct: statistics of active and finished jobs

SLURM: example, running jobs

```
srun --ntasks=4 hostname
srun --nodes=2 --ntasks=4 hostname

#!/bin/bash
#SBATCH --job-name=ime_mojega_posla
#SBATCH --partition=gridlong
#SBATCH --ntasks=4
#SBATCH --nodes=1
#SBATCH --mem-per-cpu=100MB
#SBATCH --output=moj_posel.out
#SBATCH --time=00:01:00
srun hostname

salloc –reservation=fri –ntasks=2
srun hostname
```

SLURM: example, modules and containers

module load FFmpeg

module list

module spider

srun --reservation=fri ffmpeg –y -i llama.mp4 llama.avi

sstat --job=590860 --format= AveRSS,AveVMSize,MaxRSS,MaxVMSize

sacct --job=590860 --format= cputime,AveRSS,AveVMSize,MaxRSS,MaxVMSize

singularity pull docker://jrottenberg/ffmpeg:alpine

srun --reservation=fri singularity exec ffmpeg_alpine.sif ffmpeg -y -i llama.mp4 llama.avi

ARC (extras)

Advanced Resource Connector

Enables sharing and federation of resources across different domains

Easily integrates existing resources to grid

- No need for reorganization of existing cluster
  - Cluster with batch system LRMS (Local Resource Management System)
  - Installed on top of existing batch processor, interfaces to it
- Standard interface, support for many platforms

Automatically searches for computing resources based on requirements

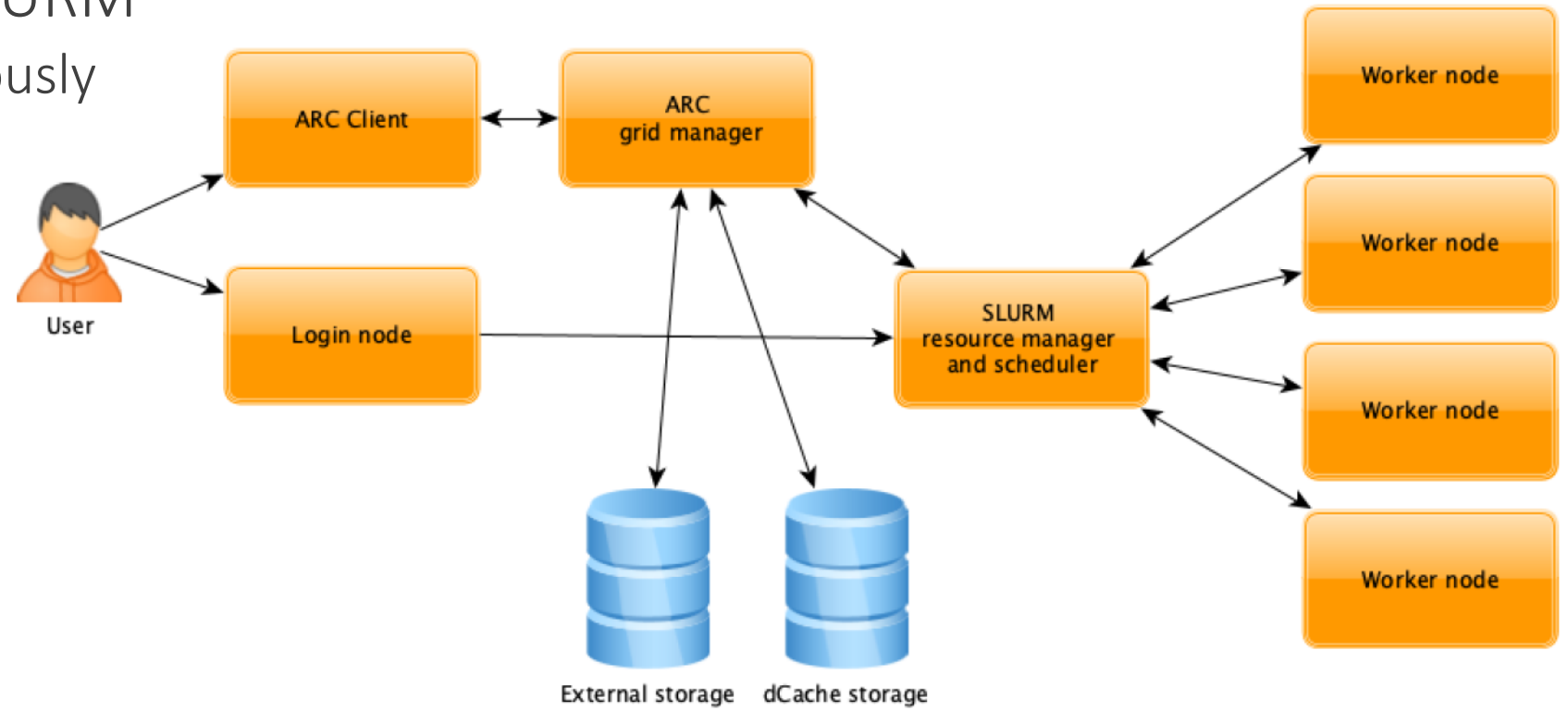
Able to distributes jobs to more groups at once

Administration of domain access

ARC (extras)

Advantages over SLURM

- Run jobs simultaneously on multiple clusters
- Unified runtime environments
- Data management nodes



Disadvantages

- Response
- No possibilities for complex job submission

ARC (extras)

Client side

- Client computer with ARC middleware
- Job submission, inquiry, result retrieval

Server side

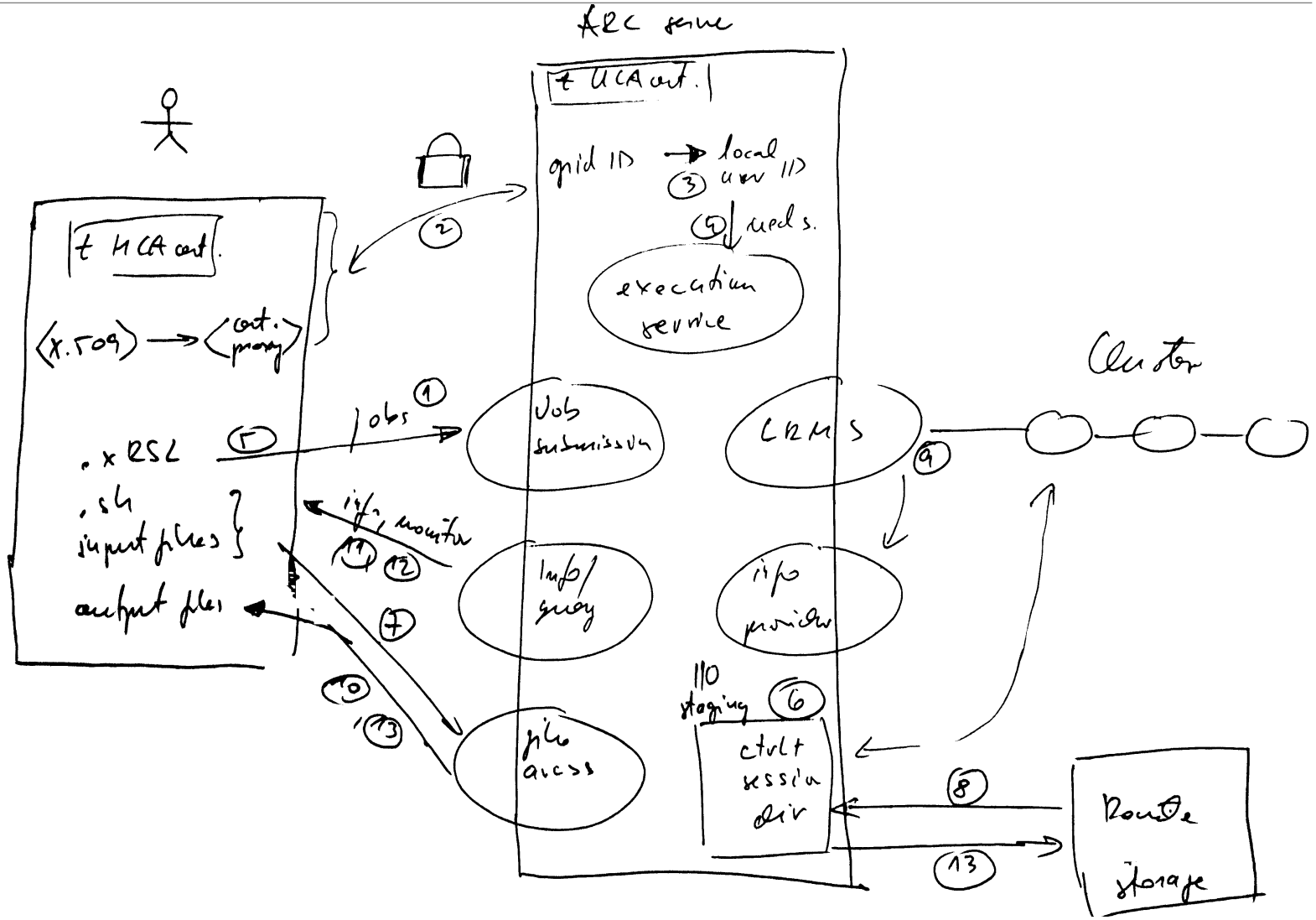
- Offers 3 services: computation, storage, information
- Computing element (CE)
  - Front-end interface to workstation or cluster –
    integrates to LRMS (Local Resource Management System)
  - Control of execution

ARC (extras): CE architecture

Execution service functionality (**A**RC **R**esource-coupled **EX**ecution system)

- Info:
  - Advertise server for the clients to know about it (location, capabilities)
  - Monitor status of jobs (info provider scripts)
- Jobs:
  - Accept jobs for execution through job submission interface
  - Forward jobs to LRMS
    - SLURM - Simple Linux Utility Resource Management
- File access:
  - Accept input files
    - from file access interface or remote storage
    - prevent frequent download by caching
  - Return results
    - file access interface or remote storage

ARC (extras): CE architecture



ARC (extras): Security on the grid

Contradiction

- access is offered to almost everyone but complicated with certificates

Authorization & authentication

- X.509 certificates
- Issued by member of IGTF – international grid trust federation
  SiGNET (http://signet-ca.ijs.si)

Certificates

- Each user has personal certificate
- Each server has host certificate

Users are joined to virtual organization (VO)

- VOMS: virtual organization management service (membership and roles)
- FRI has its own VO: fri.vo.sling.si
- National test VO: gen.vo.sling.si

ARC (extras): Security on the grid

Certificate x.509 requires password

- Automatic execution?

Certificate x.509 + password → short lived certificate proxy

- Basic element for authentication & authorization
- Delegate user‘s rights to jobs or other activities of grid services
- Validity 12 … 24 h

Client CA certificate and Server CA certificate must be present on server and client

Credentials (authorization policy) are based on VO settings

ARC (extras): job description

What software to execute, what data to process, needed run-time environment, hardware requirements

Specified as XRSL – eXtended Resource Specification Language

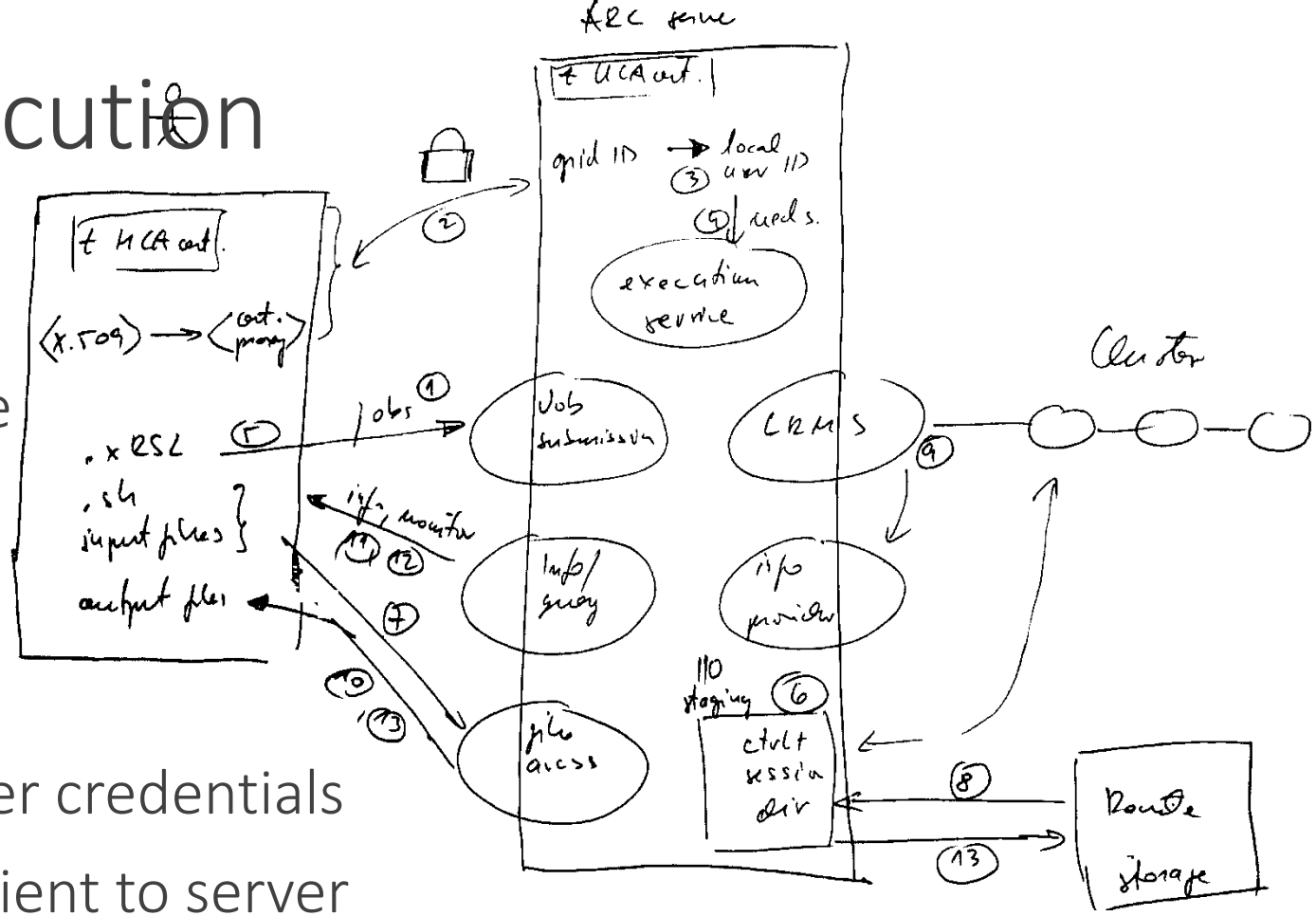
Accompanied with

- Executables (job.sh)
- Input files

Job length: up to 24 hours

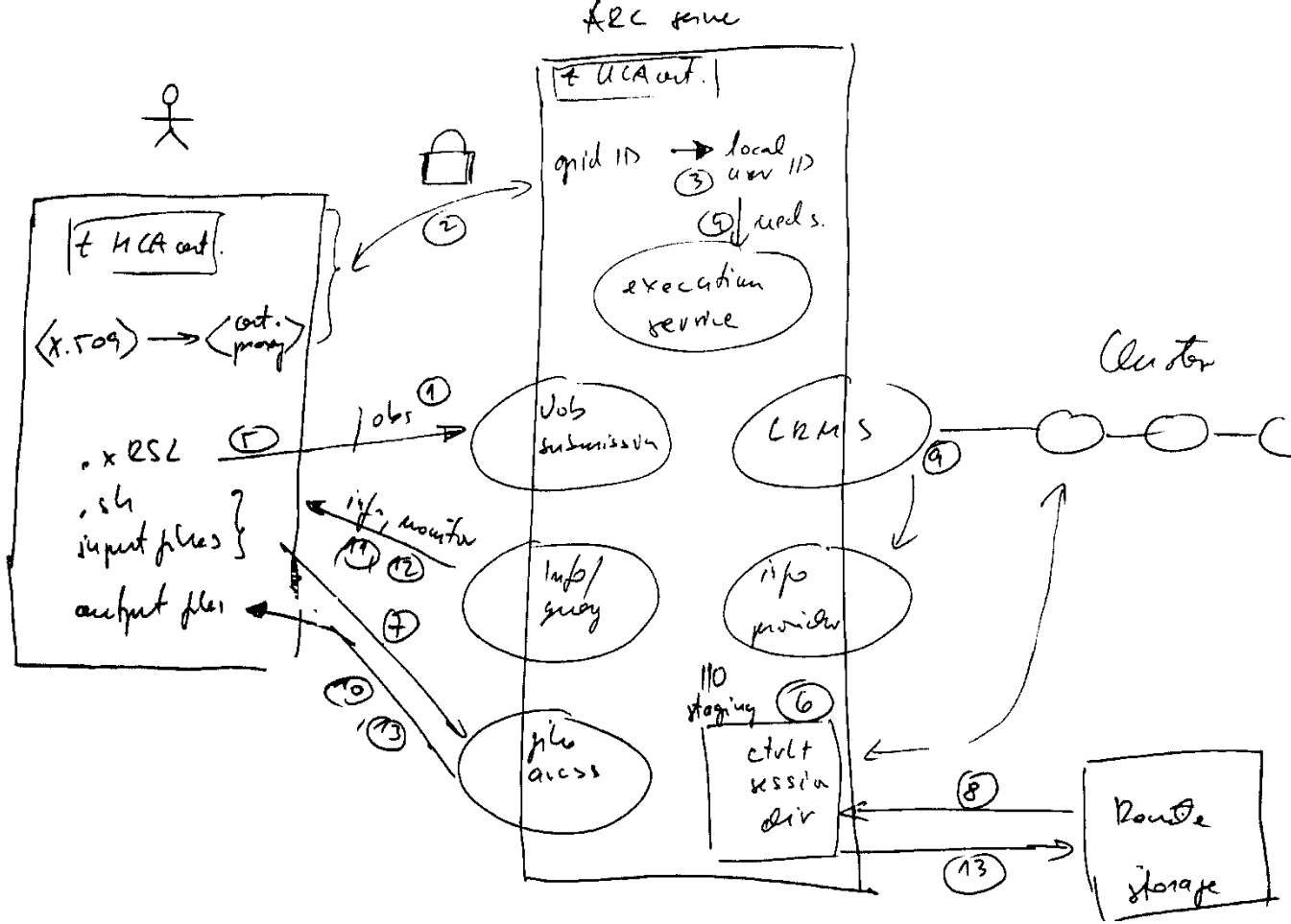
ARC (extras): job execution

1. Client connects to job submission interface
2. Client and server authenticate each other
3. Execution Service authorizes user and maps grid ID to local user
4. Execution service acquires user credentials
5. Job description is sent from client to server
6. Job accepted and session and control directory created
7. Client copies files to session directory
8. Execution service copies files from remote locations



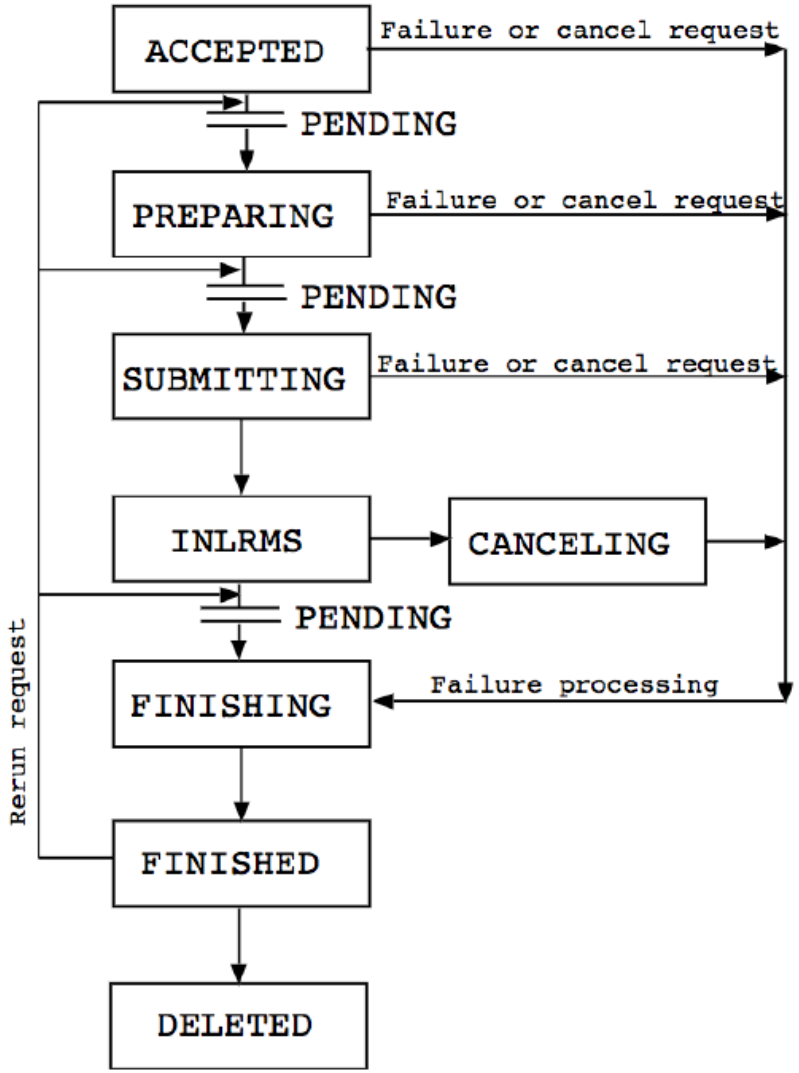
ARC (extras): job exe

9. Job is submitted to LRMS
10. During processing session directory is accessible
11. Information provider monitors the jobs and updates control directory
12. When job is finished, execution system keeps or uploads files
13. Client may download resulting files or put them to Remote storage



ARC (extras): job statuses

Job lifecycle



ARC (extras): Run-time Environments

Each research area demands specific software, libraries, tools

ARC uses the concept of

- run-time environments
  - Interface to software independent of underlying hardware
  - Shell Script
    - installed on local computing resources
    - Initializes variables that point to application software
    - Setup before local job execution
- containers Singularity/Docker

RE configuration

- Basic language: C/C++
- Script languages: perl/bash/python

RE examples:

- MPI, java, python, octave, atlas

ARC (extras): commands

arcsub: job submission
- arcsub -c maister.hpc-rivr.um.si test.xrsl

arcstat: job stats
- arcstat -a

arckill: delete a job
- arckill <jobid>

arcget: to retrieve results
- arcget -a ali arcget <jobid>

arcinfo: cluster info:
- arcinfo nsc.ijs.si

arccat: job output
- arccat <jobid>

ARC (extras): ffmpeg example

XRSL script

ffmpeg-arc.xrsl

arcproxy –l

arcproxy –S fri.vo.sling.si

arcsub -c nsc.ijs.si ffmpeg-arc.xrsl * opcija tudi jost.arnes.si

arcstat –a

arcget -a

SLING

= cocktail: ice, 2 whiskey/gin, ½ lemon juice, ¼ sugar syrup, soda

= weapon

= support for broken arm

= **SL**ovenian **I**nitiative for **N**ational **G**rid

= Slovenian national supercomputing network

SLING

SLING

History (2002)

- Quest for God particle (Highs boson)
- Large Hadron Collider
- Whole capacity
  - 350 centres, 1 mi. cores, 1 EB disk
  - CERN, Atlas experiment used 15 %

Slovenia joined its capacities for international activities

- CERN
- EU projects

SLING

Formed by ARNES & IJS in 2010

Goal: national computer infrastructure for science and research

Tasks

- National grid service
- Coordination, technical development
- Technical support (centres, users)
- Integration to EU and WW infrastructures

Member of EGI (European Grid Infrastructure) as part of NorduGRID

SLING

SLING in numbers

- 9 active centers
- 35k CPU cores
  - Maister ~28%
  - IJS ~55 %
  - Arnes ~12%
- >1 PB disk capacities
- 80 % Slovenian capacity
- 1 – 2 % EGI capacity

SLING

Access grants

- Personal certificate issued by SLING
  - Research institutions, Universities, Companies subject to fees
  - Permits access to other EU HPC systems
- Requests for large amounts of computational time
  - Submit a proposal
  - Special commission decides
  - 1/3 owned by EC, application through EuroHPC
- Part of the system always available for smaller jobs

SLING