Parallel Programming on EGEE: Best practices

Gergely Sipos
MTA SZTAKI

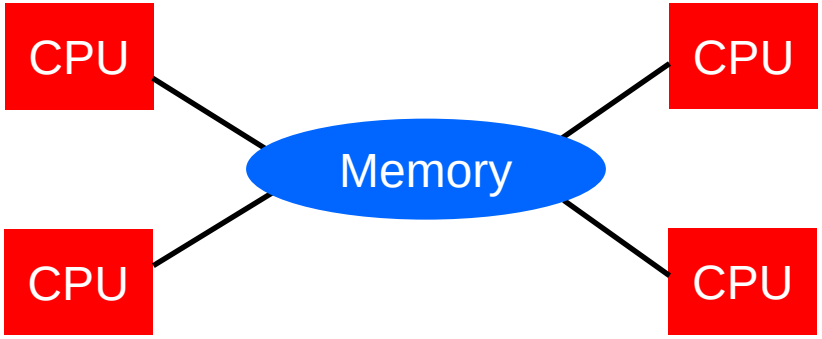
www.eu-egee.org

Information Society and Media

Outline

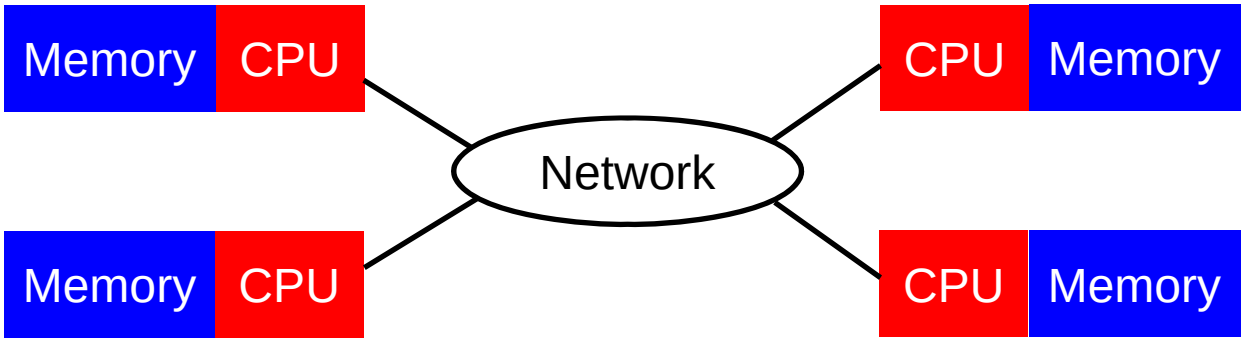
- **Parallel computing architectures**
  - Supercomputers, clusters, EGEE grid
- **Functional vs data parallelism**
- **Patterns and best practices for data parallelism**
  - From jobs to master slave to workflow

Traditional distributed architectures: Shared memory architecture



- **Multiple processors operate independently but share the same memory resources**
- **Only one processor can access the shared memory location at a time**
  - Mutual exclusion provided by at system level
- **Synchronization achieved by controlling tasks' reading from and writing to the shared memory**
- **Typical architecture of supercomputers**

Traditional distributed architectures: Distributed memory architecture

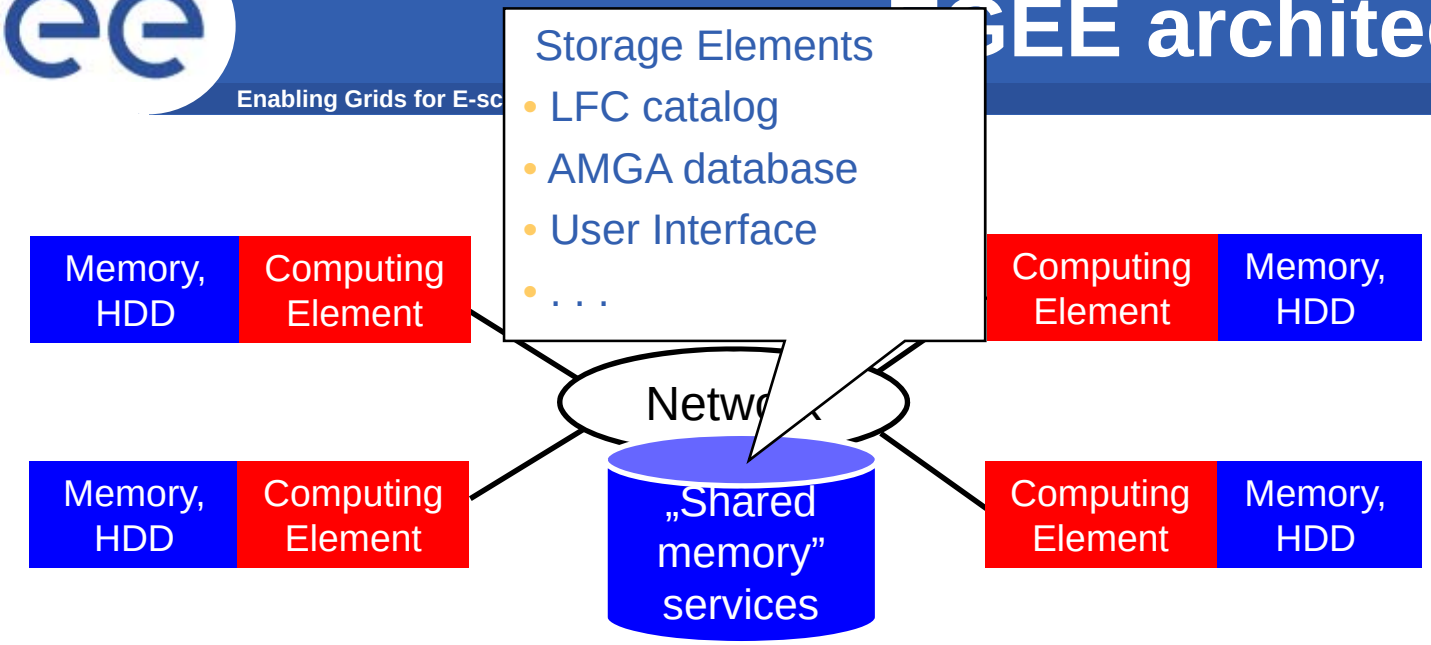


- **Multiple processors operate independently, each has its own private memory**
- **Data is shared across a network using message passing**
  - User responsible for synchronization using message passing
- **Typical architecture of clusters**

References

- SP Parallel Programming Workshop – Parallel Programming Introduction: http://www.mhpcc.edu/training/workshop/parallel_intro/MAIN.html

EGEE architecture



- **Some kind of distributed memory system:**
  - Multiple processors operate independently, each has its own private memory (HDD & memory in a Computing element)
  - Direct communication between CEs is not possible (such as MPICH)
- **Some kind of shared memory system:**
  - Central services to share data between CEs (between jobs) (e.g. Storage elements)
  - Communicating through central services must be handled at user level
    - No mutual exclusion, locking, etc.

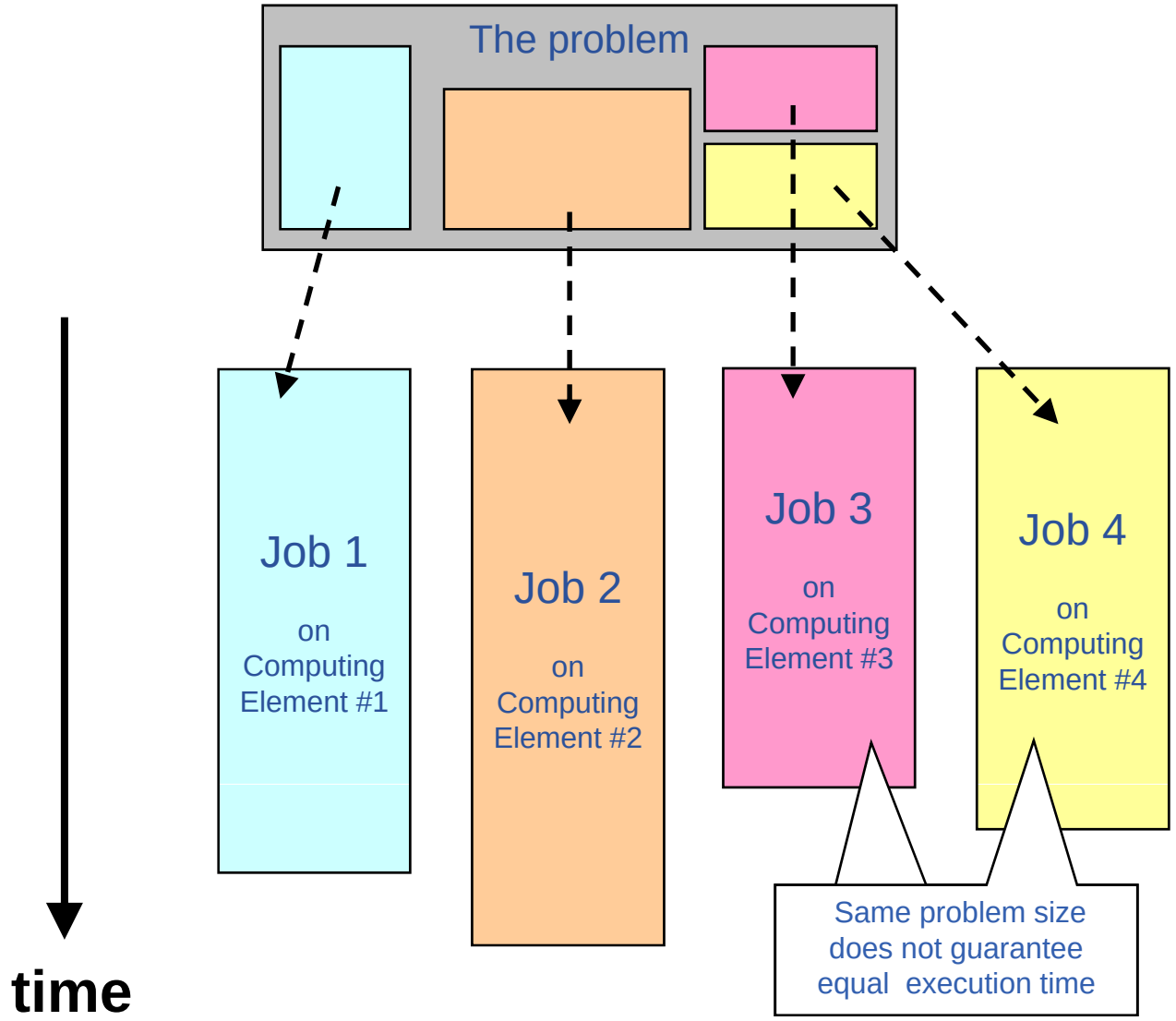
Functional Vs Data parallelism

- **Functional Decomposition (Functional Parallelism)**
  - Decomposing the problem into different jobs which can be distributed to multiple CEs for simultaneous execution
  - Different code run on different CEs
  - Good to use when there is not static structure or fixed determination of number of calculations to be performed

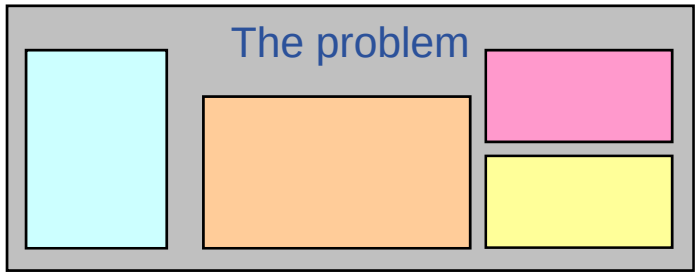
- **Domain Decomposition (Data Parallelism)**
  - Partitioning the problem's data domain and distributing portions to multiple instances of the same job for simultaneous execution
  - Same code runs on different CEs processing different data
  - Good to use for problems where:
    - data is static (e.g. factoring, solving large matrix or finite difference calculations, parameter studies)
    - dynamic data structure tied to single entity where entity can be subsetted (large multi-body problems)
    - domain is fixed but computation within various regions of the domain is dynamic (fluid vortices models)

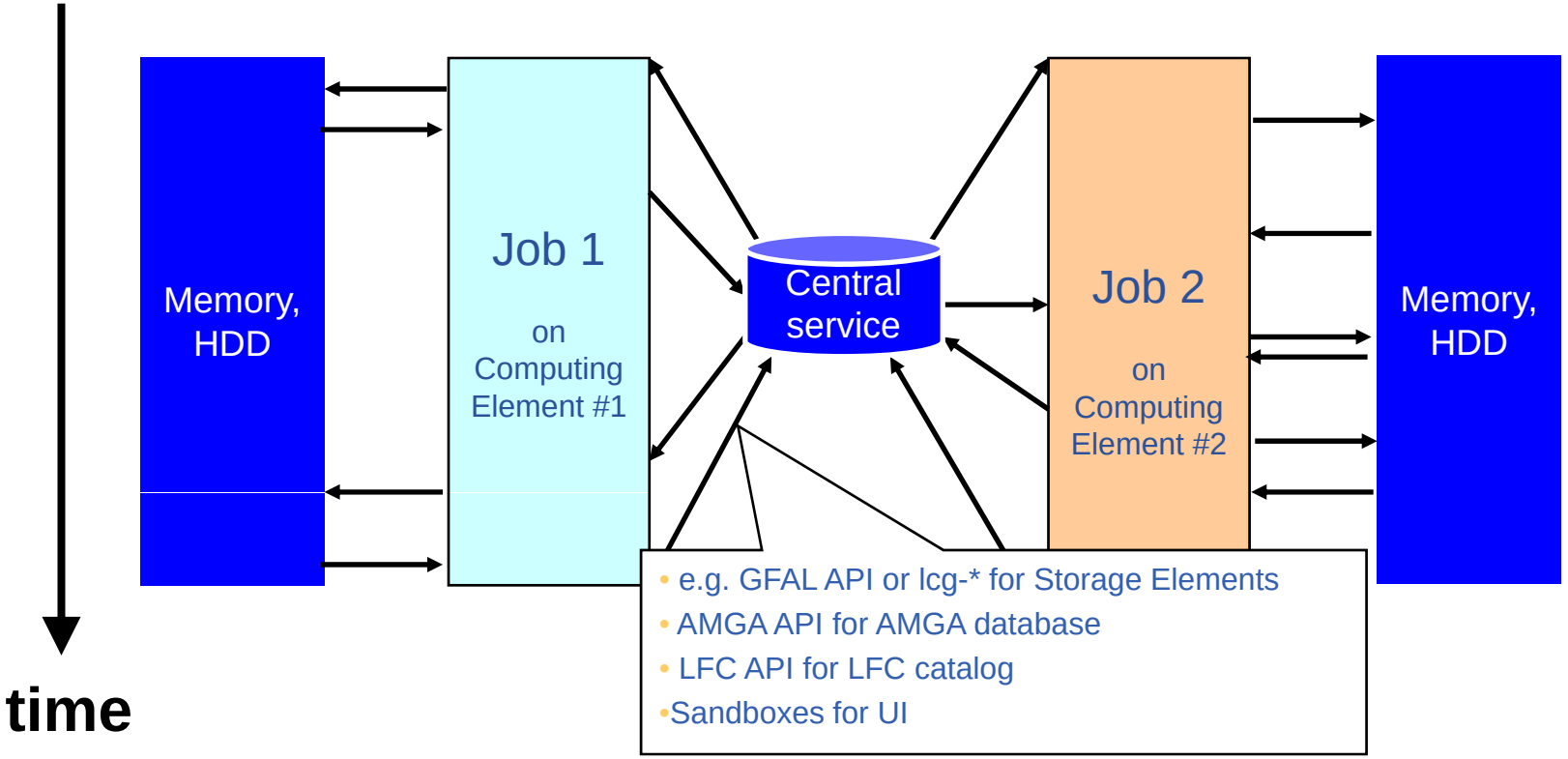
- **> 90% of grid applications employ data parallelism (parameter study)**

Functional parallelism

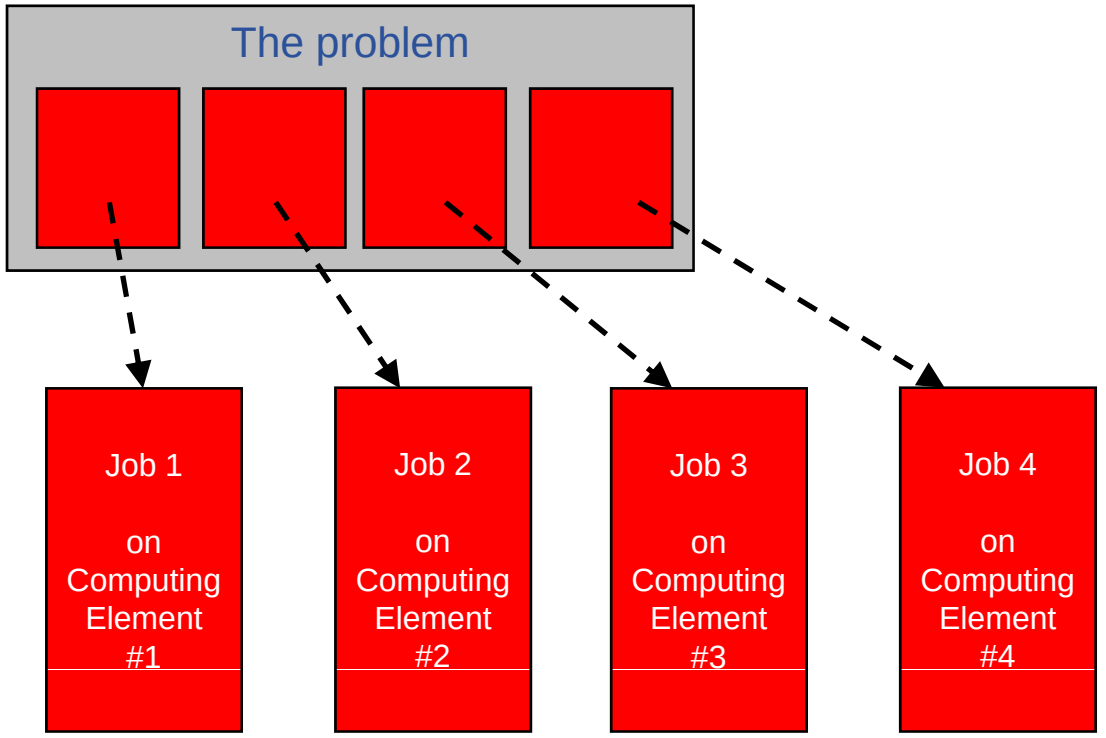


Intra-job communication





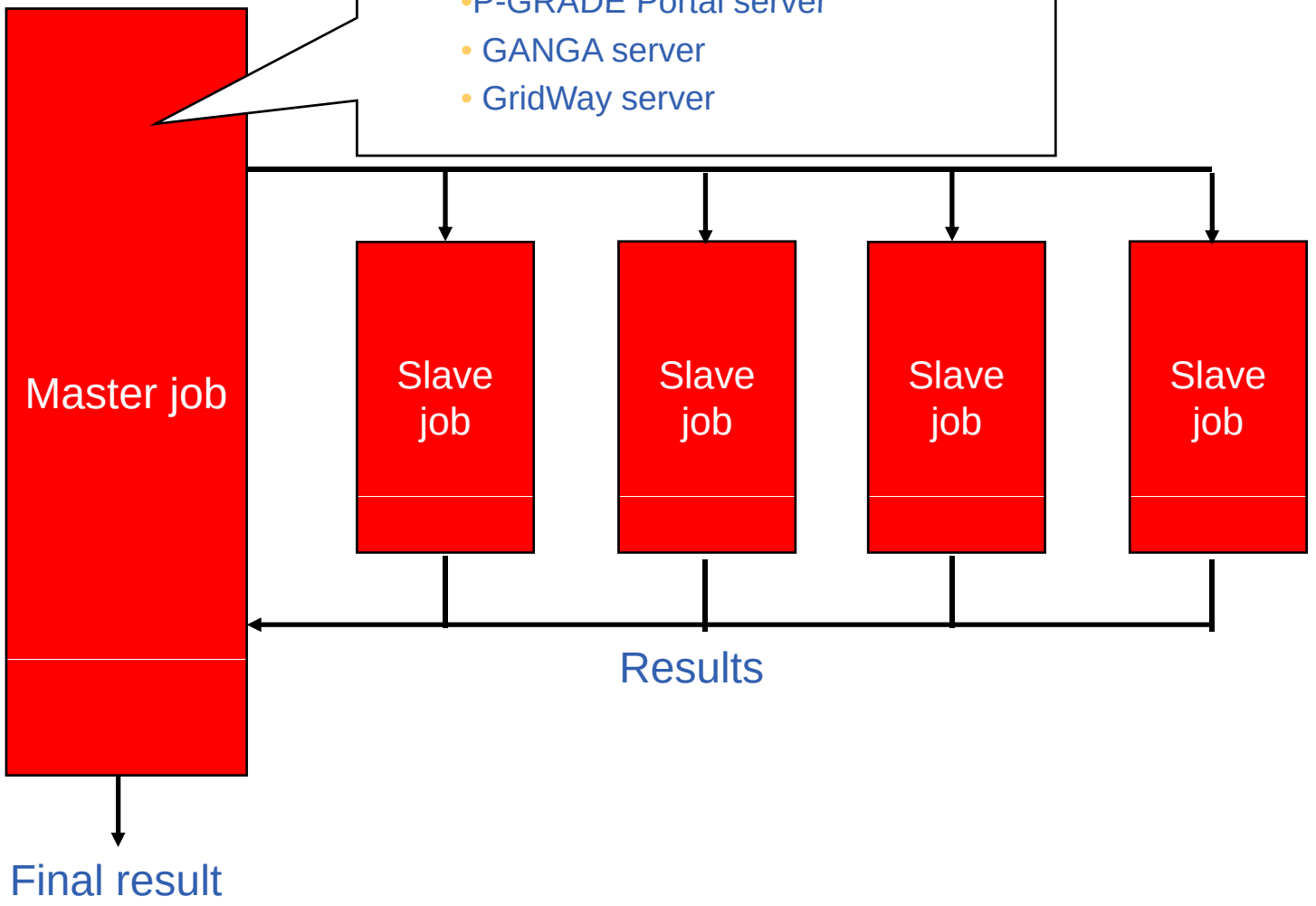
Data parallelism



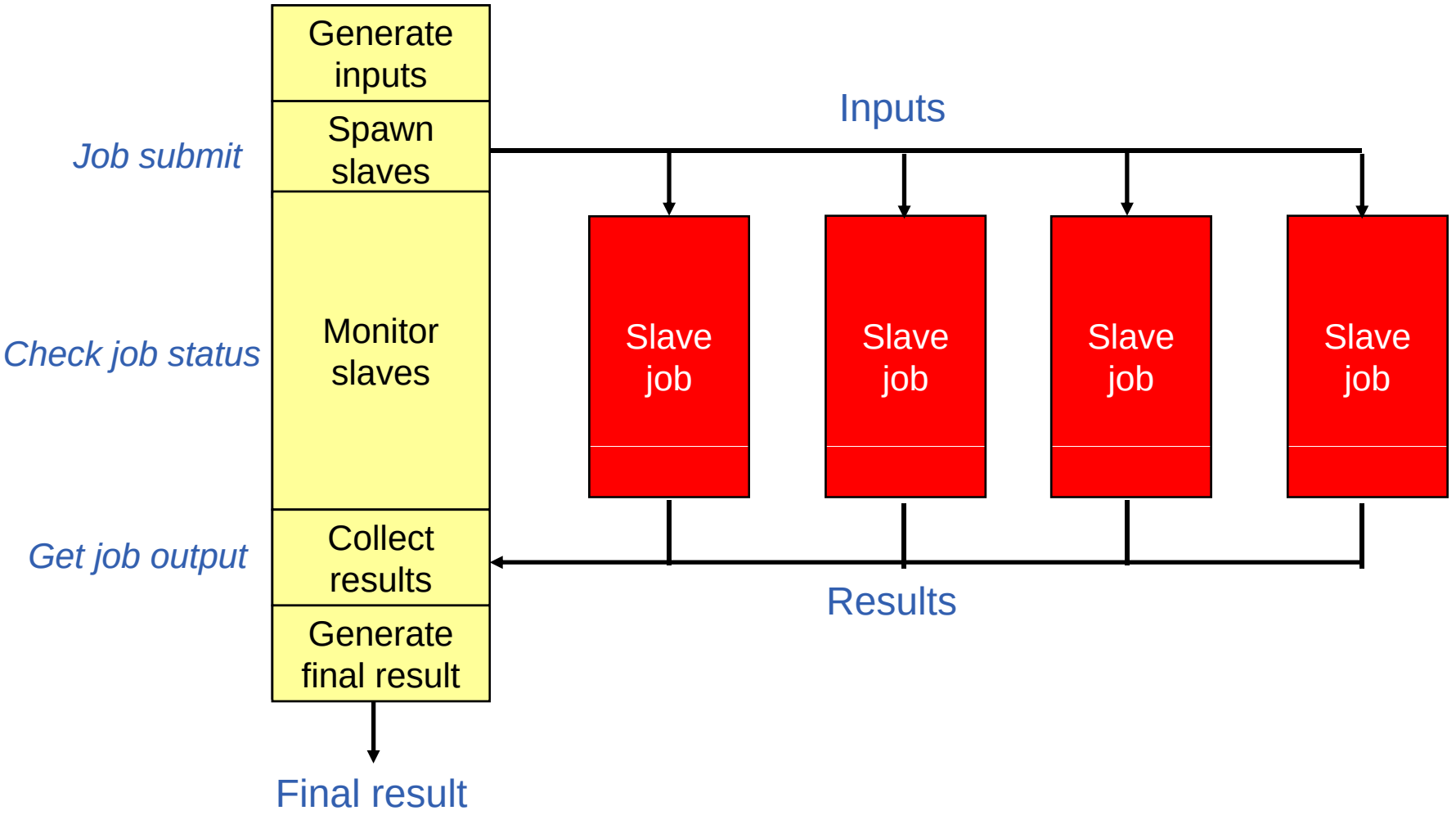
Same problem size does not guarantee equal execution time on the Grid

Data parallelism: Master-slave paradigm

- User process running on the UI or
- on a central server such as
  - WMS
  - P-GRADE Portal server
  - GANGA server
  - GridWay server



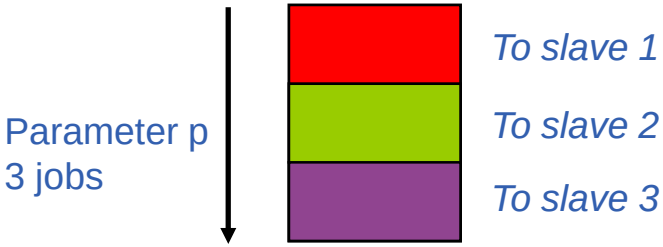
Structure of the master



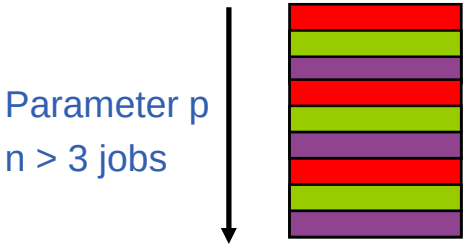
Data distibution techniques

- **One Dimensional Data Distribution**

Block Distribution

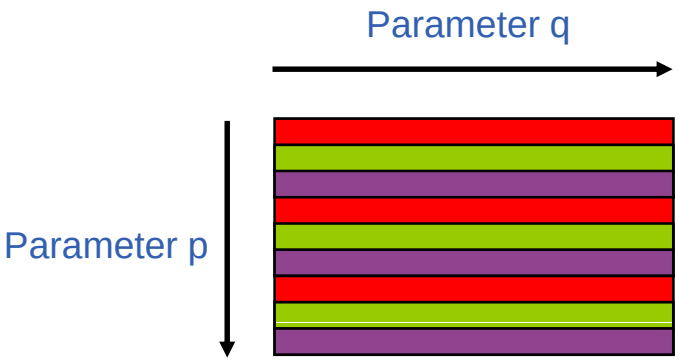


Cyclic Distribution

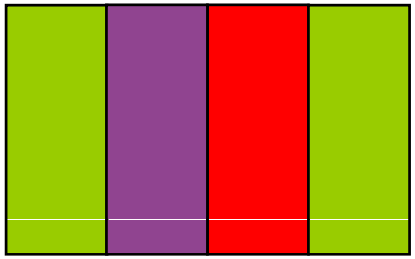


- **Two Dimensional Data Distribution**

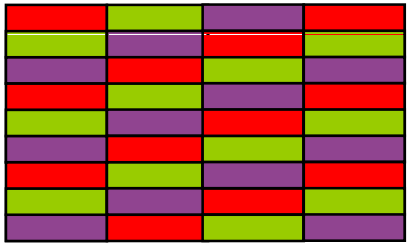
Cyclic block



Block Cyclic



Block Block



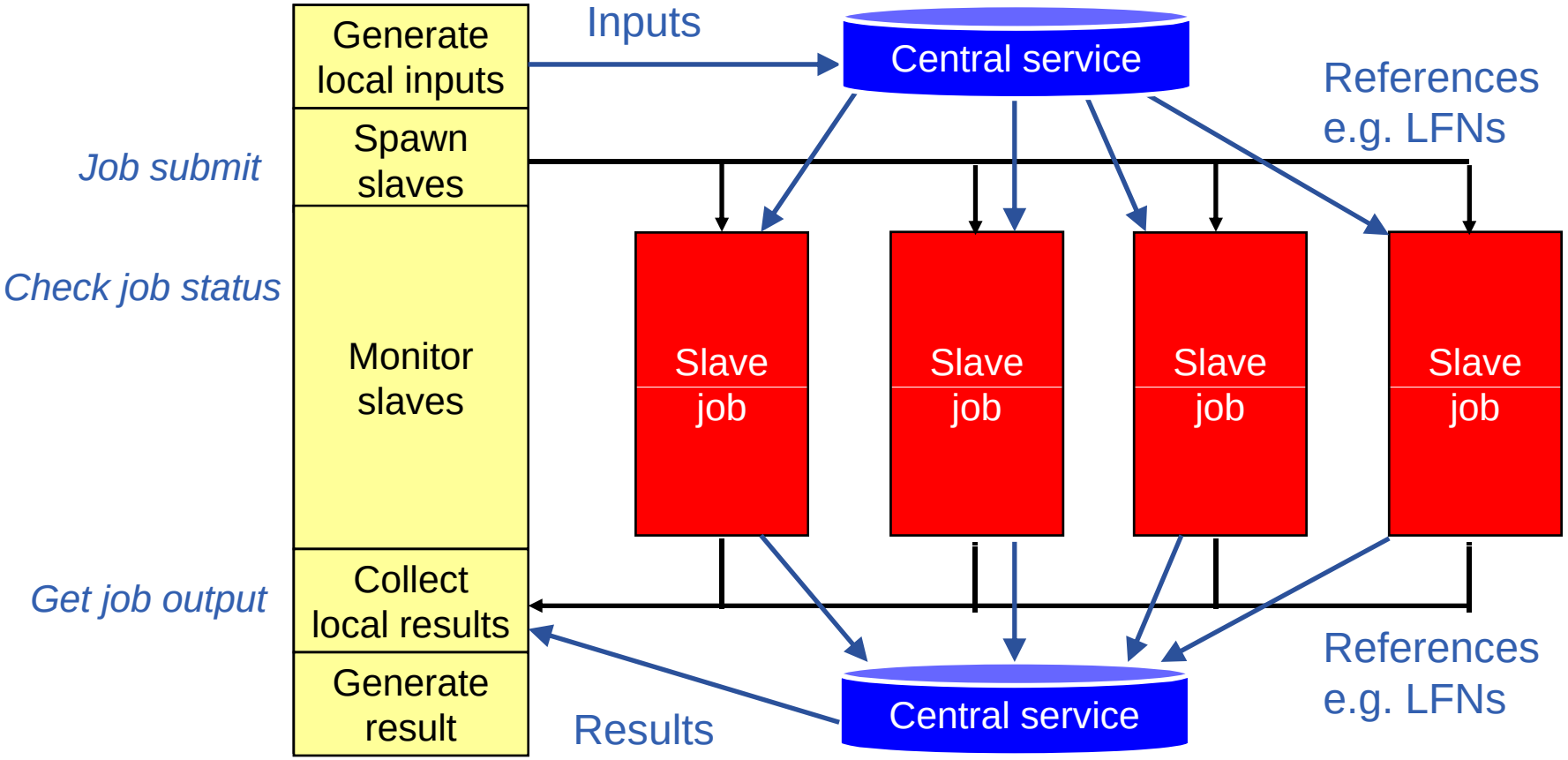
Choose job number carefully

- **Less jobs → long jobs:**
  - Smaller submission overhead
    - Middleware overhead 5-10 minutes / job
    - Waiting queue overhead 0-X minutes / job → depends on the VO
  - Unequal utilization of resources
    - Slow and fast resources must do the same amount of work

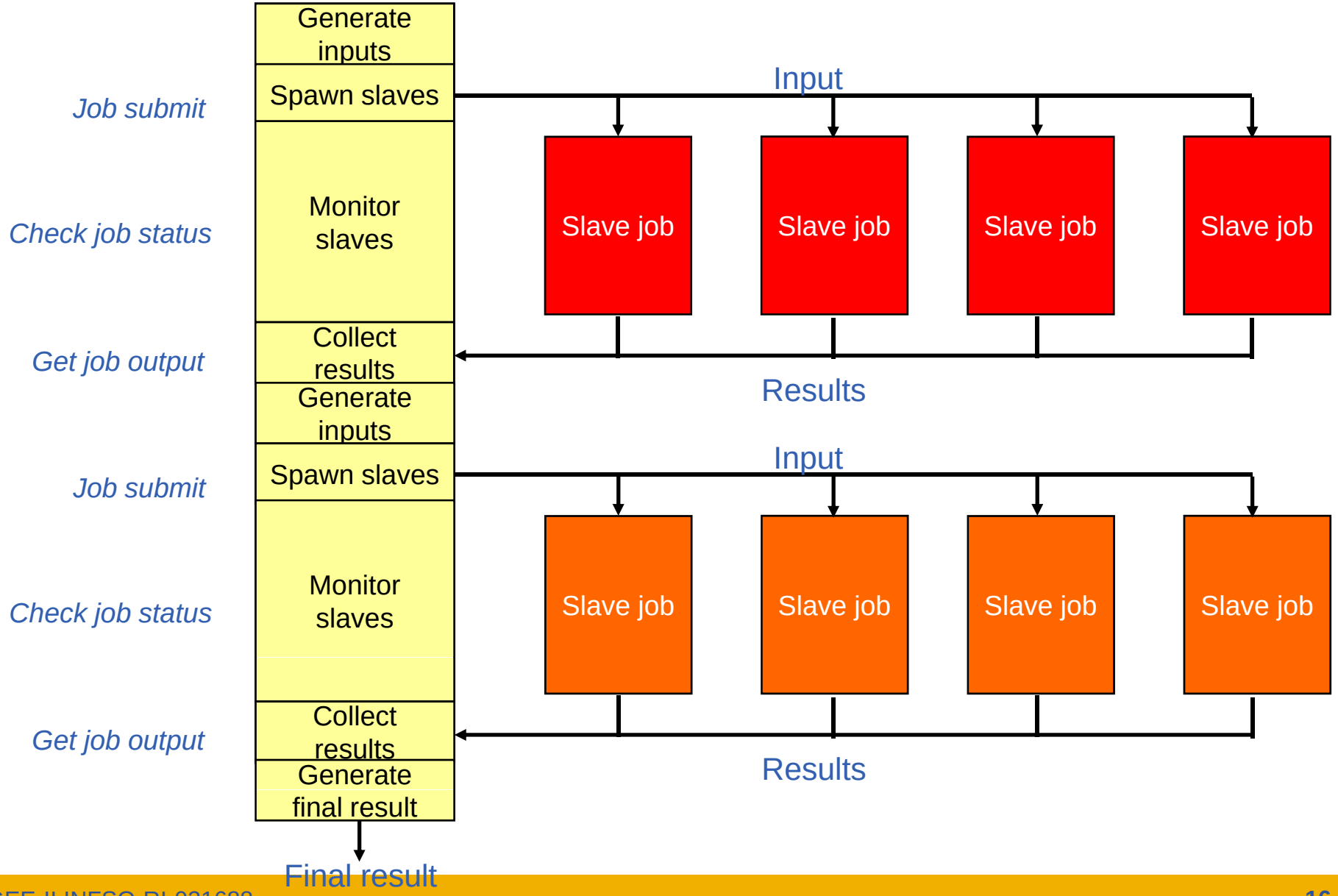
- **More jobs → short jobs:**
  - Better load balancing
    - Faster machines do more
    - Overall execution time can be shorter
  - Submission overhead is bigger

Distribution of large data sets

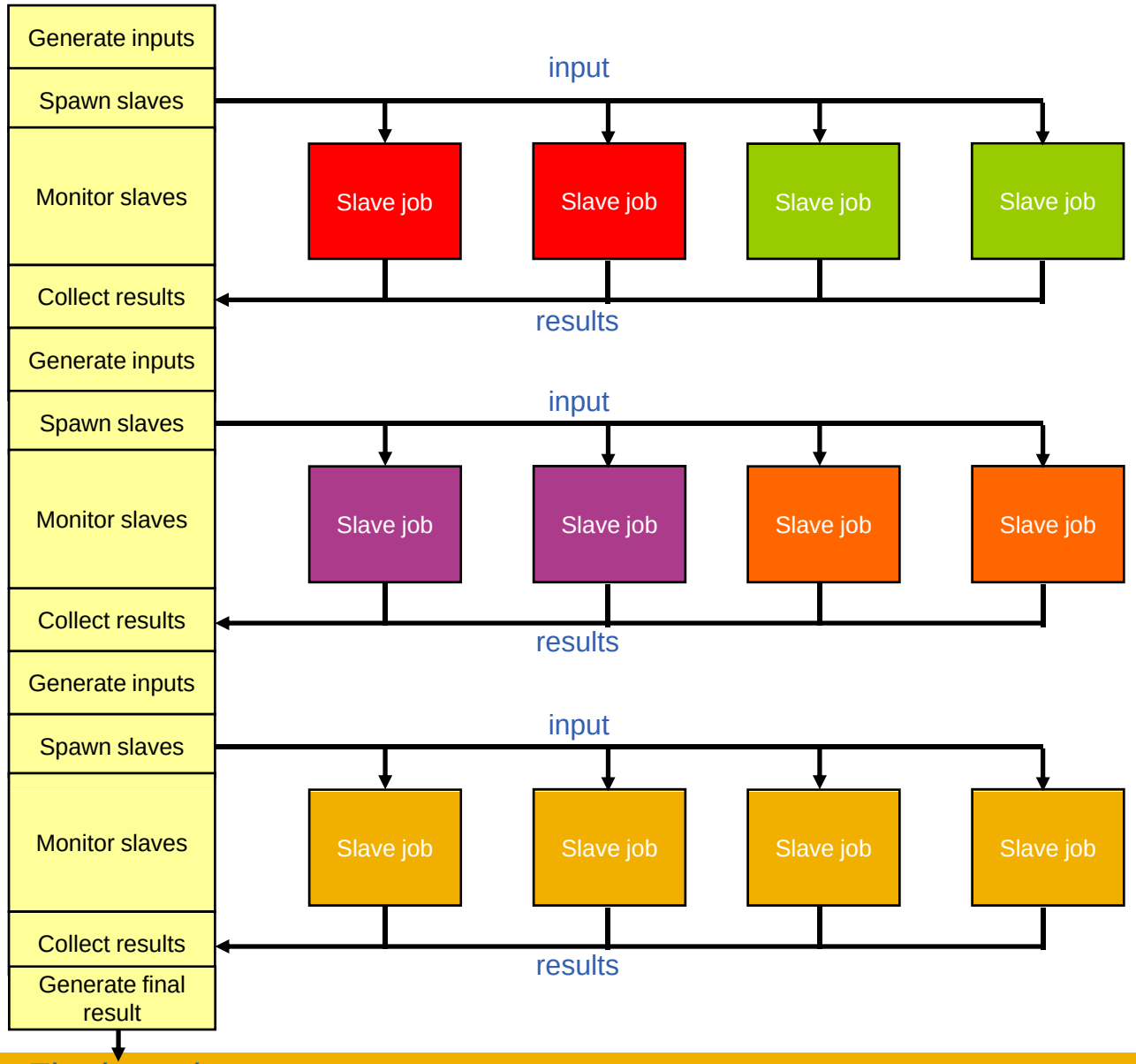
- **Slaves receives only data reference from master and download real data from Storage Element, AMGA, etc.**
- **Slaves put results into Storage Elements, AMGA, etc. and return references**



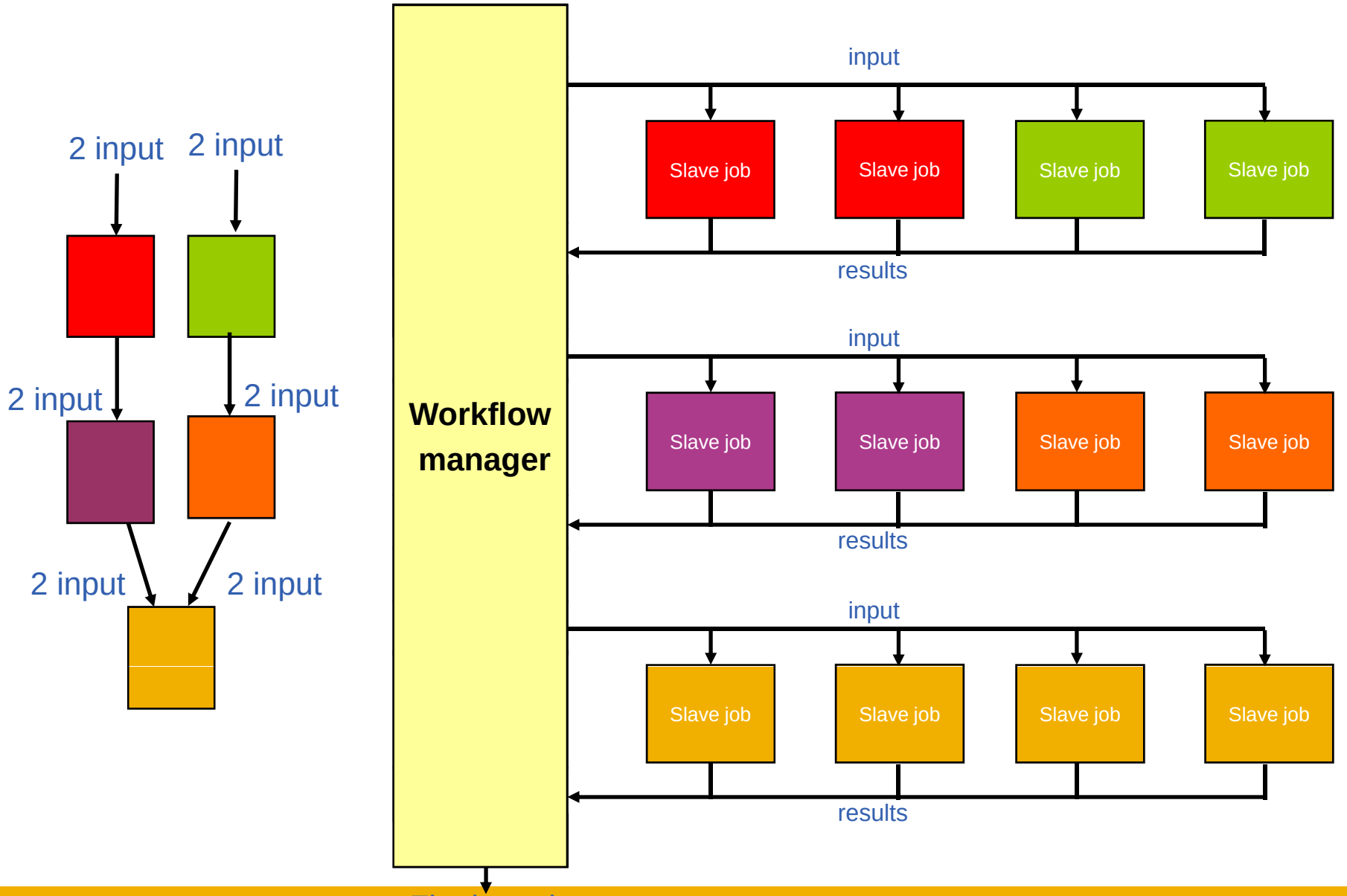
Multi-level master-slave



Complex master-slave



Complex master-slave = workflow



Workflow managers

- **Mechanisms to tie pieces of application together in standard ways**
- **Better than doing it yourself**
  - workflow systems handle many of the gritty details
    - you could implement them yourself
    - you would do it very badly (trust me)
  - useful 'additional' functionality beyond basic plumbing such as
    - Failure management
    - Resubmission
    - Data conversion
- **Different requirements per scientific discipline or by application**
  - Support for multiple levels of parallelization
  - Data semantics and / or Control flow semantics
  - Monitoring (especially for long-running workflows)
  - . . .

Many workflow systems for different grid middleware

- Askalon
- Bigbross Bossa
- Bea's WLI
- BioPipe
- BizTalk
- BPWS4J
- Breeze
- Carnot
- Con:cern
- DAGMan
- DiscoveryNet
- Dralasoft
- Enhydra Shark
- Filenet
- Fujitsu's i-Flow
- GridAnt
- Grid Job Handler
- GRMS (GridLab Resource Management System)
- GWFE
- GWES
- IBM's holosofx tool
- IT Innovation Enactment Engine
- ICENI
- Inforsense
- Intalio
- jBpm
- JIGSA
- JOpera
- Kepler
- Karajan
- Lombardi
- Microsoft WWF
- Microsoft WWF
- Moteur
- NetWeaver
- Oakgrove's reactor
- ObjectWeb Bonita
- OFBiz
- OMII-BPEL
- Open Business Engine
- Oracle's integration platform
- OSIRIS
- OSWorkflow
- OpenWFE
- Q-Link
- Pegasus
- Pipeline Pilot
- Platform Process Manager
- P-GRADE
- PowerFolder
- PtolemyII
- Savvion
- Seebeyond
- Sonic's orchestration server
- Staffware
- ScyFLOW
- SDSC Matrix
- SHOP2
- Swift
- Taverna
- Triana
- Twister
- Ultimus
- Versata
- WebMethod's process modeling
- wftk
- XFlow
- YAWL Engine
- WebAndFlo
- Wildfire
- Werkflow
- wfmOpen
- WFEE
- ZBuilder ……

Many workflow systems for different grid middleware

EGEE Biomed community

gLite WMS

EGEE related DILIGENT project

- Askalon
- Bigbross Bossa
- Bea's WLI
- BioPipe
- BizTalk
- BPWS4J
- Breeze
- Carnot
- Con:cern
- DAGMan
- DiscoveryNet
- Dralasoft
- Enhydra Shark
- Filenet
- Fujitsu's i-Flow
- GridAnt
- Grid Job Handler
- GRMS (GridLab Resource Management System)
- GWFE
- GWES
- IBM's holosofx tool
- IT Innovation
- Inforsense
- Intalio
- jBpm
- JIGSA
- JOpera
- Kepler
- Karajan
- Lombardi
- Microsoft WWF
- Microsoft WW
- Moteur
- NetWeaver
- Oakgrove's reactor
- ObjectWeb Bonita
- OFBiz
- OMII-BPEL
- Open Business Engine
- Oracle's integration platform
- OSIRIS
- OSWorkflow
- OpenWFE
- Q-Link
- Pegasus
- Pipeline P
- Platform Proc Manager
- P-GRADE
- PowerFolder
- PtolemyII
- Savvion
- Seebeyond
- Sonic's orchestration server
- Staffware
- ScyFLOW
- SDSC Matrix
- SHOP2
- Swift
- Taverna
- Triana
- Twister
- Ultimus
- Versata
- YAWL Engine
- WebAndFlo
- Wildfire
- Werkflow
- wfmOpen
- WFEE
- ZBuilder ……

During the course

- **gLite WMS**
  - Parametric jobs (master-slave)
  - DAG (workflow)
- **GANGA**
  - Parameter studies (master-slave)
- **P-GRADE Portal**
  - Workflows
  - Parameter studies (master-slave)
  - Workflow based parameter studies

Thank you

Questions?

www.eu-egee.org

Information Society and Media

Case studies

Not related to EGEE at all. Case studies in general

- **Numeric Weather Prediction Model, developed by Glenn Wightwick of IBM Australia Science & Technology**
  - 2D mesh
  - http://www.mhpcc.edu/training/workshop/parallel_intro/nwp_case_study.html
- **Monte Carlo Cellular Microphysiology**
  - Parameter study
  - http://whitepapers.techrepublic.com.com/casestudy.aspx?docid=109125